skiboot Documentation

Release v6.3-190-g9cae036fafea

IBM, others

Jun 03, 2019

CONTENTS

CHAPTER

ONE

OVERVIEW

1.1 Skiboot overview

Skiboot is boot and runtime firmware for OpenPOWER systems. It's loaded by earlier boot firmware (typically Hostboot). Along with loading the bootloader, it provides some runtime services to the OS (typically Linux).

1.1.1 Source layout

Directory	Content
asm/	small amount, mainly entry points
ccan/	bits from CCAN
core/	common code among machines.
doc/	not enough here
external/	tools and userspace components
hdata/	Parses HDAT from Hostboot/FSP into Device Tree
hw/	drivers for things & fsp things.
include/	headers!
libc/	tiny libc, originally from SLOF
libfdt/	Manipulate flattened device trees
libflash/	Lib for talking to flash and parsing FFS structs
libpore/	to manipulate PORE[1] engine.
libstb/	See *Secure and Trusted Boot Library (LibSTB) Documentation*
libxz/	The xz_embedded library
opal-ci/	Some scripts to help Continuous Integration testing
platforms/	Platform (machine/BMC) specific code
test/	Test scripts and binaries

We have a spinlock implementation in asm/lock.S Entry points are detailed in asm/head.S The main C entry point is in core/init.c: main_cpu_entry()

1.1.2 Binaries

The following binaries are built:

[1] Power On Reset Engine. Used to bring cores out of deep sleep states. For POWER9, this also includes the *p9_stop_api* which manipulates the low level microcode to-reinit certain SPRs on transition out of a state losing STOP state.

File	Purpose
skiboot.lid	Binary for flashing onto systems[2]
skiboot.lid.stb	Secure and Trusted Boot container wrapped skiboot
skiboot.lid.xz	XZ compressed binary[3]
skiboot.lid.xz.stb	STB container wrapped XZ compressed skiboot[4]
skiboot.elf	is the elf binary of it, lid comes from this
skiboot.map	plain map of symbols

1.1.3 Booting

On boot, every thread of execution jumps to a single entry point in skiboot so we need to do some magic to ensure we init things properly and don't stomp on each other. We choose a master thread, putting everybody else into a spinloop.

Essentially, we do this by doing an atomic fetch and inc and whoever gets 0 gets to be the main thread. The main thread will then distribute tasks to secondary threads as needed. We do not (currently) do anything fancy like context switching or scheduling.

When entering skiboot, we enter with one of two data structures describing the system as initialized by Hostboot. There may be a flattened device tree (see https://devicetree.org/), or a HDAT structure. While Device Tree is an industry standard, HDAT comes from IBM POWER. On POWER8, skiboot would get HDAT and a mini-devicetree from an FSP or purely a Device Tree on OpenPOWER systems. On POWER9, it's just HDAT everywhere (that isn't a simulator). The HDAT specification is currently not public. It is purely an interface between Hostboot and skiboot, and is only exposed anywhere else for debugging purposes.

During boot, skiboot will add a lot to the device tree, manipulating what may already be there before exporting this new device tree out to the OS.

The main entry point is main_cpu_entry() in core/init.c, this is a carefully ordered init of things. The sequence is relatively well documented there.

1.1.4 OS interface

OPAL (skiboot) is exclusively called through OPAL calls. The OS has complete controll of *when* OPAL code is executed. The design of all OPAL APIs is that we do not block in OPAL, so as not to introduce jitter.

Skiboot maintains its own stack for each CPU, the running OS does not need to donate or reserve any of its stack space.

With the OPAL API calls and device tree bindings we have the OPAL ABI.

1.1.5 Interrupts

We don't directly handle interrupts in skiboot. The OS is in complete control, and any interrupts we need to process are first received by the OS. The *OPAL_HANDLE_INTERRUPT* call is made by the OS for OPAL to do what's needed.

[2] Practically speaking, this is just IBM FSP based systems now. Since the *skiboot.lid* size is now greater than 1MB, which is the size of the default *PAYLOAD* PNOR partition size on OpenPOWER systems, you will want the *skiboot.lid.xz* or *skiboot.lid.xz.stb* instead.

[3] On OpenPOWER systems, hostboot will read and decompress XZ compressed payloads. This shortens boot time (less data to read), adds a checksum over the *PAYLOAD* and saves valuable PNOR space. If in doubt, use this payload.

[4] If a secure boot system, use this payload.

1.1.6 Memory

We initially occupy a chunk of memory, "heap". We pass to the OS (Linux) a reservation of what we occupy (including stacks).

In the source file include/mem-map.h we include a memory map. This is manually generated, not automatically generated.

We use CCAN for a bunch of helper code, turning on things like DEBUG_LOCKS as these are not a performance issue for us, and we like to be careful.

In include/config.h there are defines for turning on extra tracing. OPAL is what we name the interface from skiboot to OS (Linux).

Each CPU gets a 16k stack, which is probably more than enough. Stack should be used sparingly though.

Important memory locations:

Location	What's there
SKIBOOT_BASE	where skiboot lives, of SKIBOOT_SIZE
HEAP_BASE	Where skiboot heap starts, of HEAP_SIZE

There is also SKIBOOT_SIZE (manually calculated) and DEVICE_TREE_MAX_SIZE, which is largely historical.

1.1.7 Skiboot log

There is a circular log buffer that skiboot maintains. This can be accessed either from the FSP or through /dev/mem or through the sysfs file /sys/firmware/opal/msglog.

1.2 OPAL Specification

DRAFT - VERSION 0.0.1 AT BEST.

COMMENTS ARE WELCOME - and indeed, needed.

If you are reading this, congratulations: you're now reviewing it!

This document aims to define what it means to be OPAL compliant.

While skiboot is the reference implementation, this documentation should be complete enough that (given hardware documentation) create another implementation. It is not recommended that you do this though.

1.2.1 Authors

Stewart Smith <stewart@linux.ibm.com> : OPAL Architect, IBM

1.2.2 Definitions

Host processor the main POWER CPU (e.g. the POWER8 CPU)

Host OS the operating system running on the host processor.

OPAL OpenPOWER Abstraction Layer.

1.2.3 What is OPAL?

The OpenPower Abstraction Layer (OPAL) is boot and runtime firmware for POWER systems. There are several components to what makes up a firmware image for OpenPower machines.

For example, there may be:

- BMC firmware
  - Firmware that runs purely on the BMC.
  - On IBM systems that have an FSP rather than a BMC, there is FSP firmware
  - While essential to having the machine function, this firmware is not part of the OPAL Specification.
- HostBoot
  - HostBoot (https://github.com/open-power/hostboot) performs all processor, bus and memory initialization within IBM POWER based systems.
- OCC Firmware
  - On Chip Controller (Firmware for OCC - a PPC405 core inside the IBM POWER8 in charge of keeping the system thermally and power safe).
- SkiBoot
  - Boot and runtime services.
- A linux kernel and initramfs incorporating petitboot
  - The bootloader. This is where a user chooses what OS to boot, and petitboot will use kexec to switch to the host Operating System (for example, PowerKVM).

While all of these components may be absolutely essential to power on, boot and operate a specific OpenPower POWER8 system, the majority of the code mentioned above can be thought of as implementation details and not something that should form part of an OPAL Specification.

For an OPAL system, we assume that the hardware is functioning and any hardware management that is specific to a platform is performed by OPAL firmware transparently to the host OS.

The OPAL Specification focus on the interface between firmware and the Operating System. It does not dictate that any specific pieces of firmware code be used, although re-inventing the wheel is strongly discouraged.

The OPAL Specification explicitly allows for:

- A conforming implementation to not use any of the reference implementation code.
- A conforming implementation to use any 64bit POWER ISA conforming processor, and not be limited to the IBM POWER8.
- A conforming implementation to be a simulator, emulator or virtual environment
- A host OS other than Linux

Explicitly not covered in this specification:

- A 32bit OPAL Specification There is no reason this couldn't exist but the current specification is for 64bit POWER systems only.

1.2.4 Boot Services

An OPAL compliant firmware implementation will load and execute a payload capable of booting a Host Operating System.

The reference implementation loads a Linux kernel with an initramfs with a minimal userspace and the petitboot boot loader - collectively referred to as skiroot.

The OPAL Specification explicitly allows variation in this payload.

A requirement of the payload is that it MUST support loading and booting an uncompressed vmlinux Linux kernel. [**TODO**: expand on what this actually means]

An OPAL system MUST pass a device tree to the host kernel. [**TODO**: expand the details, add device-tree section and spec]

An OPAL system MUST provide the host kernel with enough information to know how to call OPAL runtime services. [**TODO**: expand on this.]

Explicitly not covered by the OPAL Specification:

- Kernel module ABI for skiroot kernel
- Userspace environment of skiroot
- That skiroot is Linux.

Explicitly allowed:

- Replacing the payload with something of equal/similar functionality (whether replacing skiroot with an implementation of Zork would be compliant is left as an exercise for the reader)

1.2.5 Payload Environment

The payload is started with:

Register	Value
r3	address of flattened device-tree (fdt)
r8	OPAL base
r9	OPAL entry

1.2.6 Runtime Services

An OPAL Specification compliant system provides runtime services to the host Operating System via a standard interface.

An OPAL call is made by calling opal_entry with:

- r0: OPAL Token
- r2: OPAL Base
- r3..r10: Args (up to 8)

The OPAL API is defined in skiboot/doc/opal-api/

Not all OPAL APIs must be supported for a system to be compliant. When called with an unsupported token, a compliant firmware implementation MUST fail gracefully and not crash. Reporting a warning that an unsupported token was called is okay, as compliant host Operating Systems should use OPAL_CHECK_TOKEN to test for optional functionality.

All parameters to OPAL calls are big endian. Little endian hosts MUST appropriately convert parameters before passing them to OPAL.

Machine state across OPAL calls:

- r1 is preserved
- r12 is scratch
- r13 - 31 preserved
- 64bit HV real mode
- big endian
- external interrupts disabled

1.2.7 Detecting OPAL Support

A Host OS may need to detect the presence of OPAL as it may support booting under other platforms. For example, a single Linux kernel can be built to boot under OPAL and under PowerVM or qemu pseries machine type.

The root node of the device tree MUST have compatible = "ibm,powernv". See *Device Tree* for more details.

The presence of the "/ibm,opal" entry in the device tree signifies running under OPAL. Additionally, the "/ibm,opal" node MUST have a compatibile property listing "ibm,opal-v3".

The "/ibm,opal" node MUST have the following properties:

```
ibm,opal {
         compatible = "ibm,opal-v3";
         opal-base-address = <>;
         opal-entry-address = <>;
         opal-runtime-size = <>;
};
```

The compatible property MAY have other strings, such as a future "ibm,opal-v4". These are reserved for future use.

Some releases of the reference implementation (skiboot) have had compatible contain "ibm,opal-v2" as well as "ibm,opal-v3". Host operating systems MUST NOT rely on "ibm,opal-v2", this is a relic from early OPAL history.

The "ibm,opal" node MUST have a child node named "firmware". It MUST contain the following:

```
firmware {
        compatible = "ibm,opal-firmware";
};
```

It MUST contain one of the following two properties: git-id, version. The git-id property is deprecated, and version SHOULD be used. These are informative and MUST NOT be used by the host OS to determine anything about the firmware environment.

The version property is a textual representation of the OPAL version. For example, it may be "skiboot-4.1" or other versioning described in more detail in *Versioning Scheme of skiboot*.

1.2.8 OPAL log

OPAL implementation SHOULD have an in memory log where informational and error messages are stored. If present it MUST be human readable and text based. There is a separate facility (Platform Error Logs) for machine readable errors.

A conforming implementation MAY also output the log to a serial port or similar. An implementation MAY choose to only output certain log messages to a serial port.

For example, the reference implementation (skiboot) by default filters log messages so that only higher priority log messages go over the serial port while more messages go to the in memory buffer.

[TODO: add device-tree bits here]

1.3 Supported platforms & CPUs

NB. This file should only contain publicly available information.

1.3.1 CPUs

Name	PVR	Other names
Power8E	0x004bxxxx	Murano
Power8	0x004dxxxx	Venice
Power8NVL	0x004cxxxx	Naples
Power9N	0x004e0xxx	Nimbus 12 small core
Power9N	0x004e1xxx	Nimbus 24 small core
Power9C	0x004e2xxx	Cumulus 12 small core
Power9C	0x004e3xxx	Cumulus 24 small core
Power9P	0x004fxxxx	Axone

1.3.2 Platforms

Platform	Sub platform	Host CPU(s)	Manufacturer	compatible	Other names/Notes	Link(s)
astbmc	barreleye	Power8	Ingrasys (Foxconn)	"ingrasys,barreleye"	Barreleye, Rackspace machine	
astbmc	firestone	Power8	Wistron, IBM	"ibm,firestone"	Firestone, S822LC	[1]
astbmc	garrison	Power8	IBM	"ibm,garrison"	Minsky, "S822LC for HPC"	[2]
astbmc	habanero	Power8	Tyan	"tyan,habanero"	Habanero, TN71-BP012	
astbmc	p8dtu	Power8	Supermicro	"supermicro,p8dtu1u"	Briggs, "S822LC for Big Data"	[3][4]
astbmc	p8dtu	Power8	Supermicro	"supermicro,p8dtu2u"	Stratton, S821LC	[5][6]
astbmc	p8dnu	Power8	Supermicro	"supermicro,p8dnu(1u|2u)?"		
astbmc	p9sdu	Power9	Supermicro	"supermicro,p9dsu"	Boston, LC921/LC922	[7]
astbmc	palmetto	Power8	Tyan	"tyan,palmetto"	Palmetto, GN70-BP010	
astbmc	romulus	Power9		"ibm,romulus"	Romulus	
astbmc	vesnin	Power9	Yadro	"YADRO,vesnin"	VESNIN	
astbmc	witherspoon	Power9		"ibm,witherspoon"	Witherspoon, Newell, AC922	[8]
astbmc	zaius	Power9	Ingrasys (Foxconn)	"ingrasys,zaius"	Zaius, Google Machine	
ibm-fsp	apollo	Power8		"ibm,apollo"		
ibm-fsp	firenze	Power8E	IBM	"ibm,firenze"	Tuleta, S812L, S822L	[9]
ibm-fsp	zz	Power9		"ibm,zz-(1|2)s(2|4)u"		
rhesus	n/a	Power8E		"ibm,powernv"	Rhesus, Google machine	

[1] IBM Back In HPC With Power Systems LC Clusters
[2] Refreshed IBM Power Linux Systems Add NVLink
[3] Refreshed IBM Power Linux Systems Add NVLink
[4] First Look IBM POWER8 S822LC 8001-22C (Briggs)
[5] Refreshed IBM Power Linux Systems Add NVLink
[6] First Look IBM POWER8 S821LC 8001-12C (Stratton)
[7] Boston Power9s Set To Debut
[8] POWER9 TO THE PEOPLE
[9] IBM Power System S812L and IBM Power System S822L

CHAPTER

TWO

DEVELOPMENT PROCESS

2.1 Development and Release Process

Skiboot follows the release cycle of *op-build*, so that each new op-build has a new stable skiboot. Currently, this means that we release once every six weeks (or so). Our *goal* is to have roughly the first 4 weeks of a 6 week cycle open for merging new features, and reserving the final two weeks for stabilisation efforts after the first -rcX release.

It is *strongly* preferred to have new features (especially new APIs and device tree bindings) to come in *early* in the cycle.

Once the final release is cut, the *Skiboot stable tree rules and releases* process takes over.

Our process has evolved, and will so into the future. It's inspired by the Linux process, but not a slave to it. For example, there is currently not the volume of patches to justify a next tree.

Here's how some of the recent (at time of writing) releases have gone:

Date	Release
Oct 31st 2017	v5.9
Feb 6th 2018	v5.10-rc1
Feb 9th 2018	v5.10-rc2
Feb 15th 2018	v5.10-rc3
Feb 21st 2018	v5.10-rc4
Feb 23rd 2018	v5.10
Mar 28th 2018	v5.11-rc1
Apr 6th 2018	v5.11

2.1.1 Lifecycle of a patch

Roughly speaking, a patch has the following lifecycle:

- Design

  It is best to do design work in the open, although sometimes this is hard when upcoming unannounced hardware is involved. Often, it can be useful to post an RFC design or patch to encourage discussion. This is especially useful when designing new OPAL APs or device tree bindings. Never be afraid to send a patch (or series of patches) as RFC (Request for Comment) with whatever disclaimer you deem appropriate.

  Once you have a design, sharing it is an important part of the process of getting the applicable code upstream. Different perspectives are important in coming to elegant solutions, as is having more than one person understand the reasoning behind design decisions.

- Review and Test

  Once you think your patch is a state suitable for merging, send it to the mailing list for others to review and test. Using *git format-patch* and *git send-email* is good practice to ensure your patches survive being sent to the list. Ensure you have followed *CONTRIBUTING.md* and have your Signed-off-by present on your patches (*git commit -s* will add this for you).

  It is good practice to solicit review from an expert in the area of code you're modifying. A reviewer will add their Reviewed-by or Acked-by tags as replies, as will anybody testing it add Tested-by. The aim of reviewing and testing code before we merge it is to limit any problems to the smallest number of people possible, only merging code we are collectively confident that will *improve* life for all users and developers.

- Merged to master

  The maintainer as merged your patches to the development tree (the 'master' git branch). Soon after this, many more people are going to be running your code, so good review and testing helps ensure your inbox isn't flooded with bug reports.

  If your patch has also been sent to the stable tree, it's possible it also gets merged there soonafter.

- Stable release

  Once a stable release is made, it's likely that your code makes its way into vendor's firmware releases via their test cycles.

- Bug fixes and maintenance

  Bugs are a fact of life, sometimes in our own code, sometimes in others, and sometimes in hardware. After your patch is accepted, being available for input on possible bugs found and possible fixes is invaluable so that all can ship high quality firmware.

2.1.2 On closed source branches and forks

Even though the license that skiboot is distributed under does *allow* you to keep your changes private, we (the skiboot developers) cannot in any way provide support on the resulting code base.

Additionally, the broader PowerPC Linux community has neither the capacity, time, or resources to support Linux running on such closed source forks. The kernel developers have said that patches to the kernel to support or work around closed skiboot changes will *not* be accepted upstream.

If you keep your changes private, you are *entirely* on your own.

2.1.3 License

Skiboot is licensed under the Apache 2.0 license (see the LICENSE file in the source tree for the full text).

Portions (e.g. our libc, CCAN modules we use) are made available under a CC0, BSD, or BSD-MIT license (see LICENSE files for specifics).

2.2 Contributing to skiboot

skiboot is OPAL (OpenPOWER Abstraction Layer) boot and runtime firmware for POWER.

If you haven't already, join us on IRC (#openpower on Freenode) and on the mailing list (skiboot@lists.ozlabs.org - subscribe by going to https://lists.ozlabs.org/listinfo/skiboot)

While we do use GitHub Issues, patches are accepted via the mailing list. We expect participants to adhere to the GitHub Community Guidelines (found at https://help.github.com/articles/github-community-guidelines/).

All contributions should have a Developer Certificate of Origin (see below).

2.2.1 Development Environment

A host GCC of at least 4.9 is recommended (all modern Linux distributions provide this).

You can build on x86-64, ppc64 or ppc64le, you just need a powerpc64 (BE) cross compiler. The powerpc64le cross compilers packaged in Linux distributions can build BE code, so they are fine.

2.2.2 Developer Certificate of Origin

Contributions to this project should conform to the as defined at http://elinux.org/Developer_Certificate_Of_Origin. Commits to this project need to contain the following line to indicate the submitter accepts the DCO:

```
Signed-off-by: Your Name <your_email@domain.com>
```

By contributing in this way, you agree to the terms as follows:

```
Developer Certificate of Origin
Version 1.1

Copyright (C) 2004, 2006 The Linux Foundation and its contributors.
660 York Street, Suite 102,
San Francisco, CA 94110 USA

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.


Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I
    have the right to submit it under the open source license
    indicated in the file; or

(b) The contribution is based upon previous work that, to the best
    of my knowledge, is covered under an appropriate open source
    license and I have the right under that license to submit that
    work with modifications, whether created in whole or in part
    by me, under the same open source license (unless I am
    permitted to submit under a different license), as indicated
    in the file; or

(c) The contribution was provided directly to me by some other
    person who certified (a), (b) or (c) and I have not modified
    it.

(d) I understand and agree that this project and the contribution
    are public and that a record of the contribution (including all
    personal information I submit with it, including my sign-off) is
    maintained indefinitely and may be redistributed consistent with
    this project or the open source license(s) involved.
```

2.3 Skiboot stable tree rules and releases

If you're at all familiar with the Linux kernel stable trees, this should seem fairly familiar.

The purpose of a -stable tree is to give vendors a stable base to create firmware releases from and to incorporate into service packs. New stable releases contain critical fixes only.

As a general rule, on the most recent skiboot release gets a maintained -stable tree. If you wish to maintain an older tree, speak up! For example, with my IBMer hat on, we'll maintain branches that we ship in products.

2.3.1 What patches are accepted?

- Patches must be obviously correct and tested
  - A Tested-by signoff is *important*
- A patch must fix a real bug
- No trivial patches, such fixups belong in main branch
- Not fix a purely theoretical problem unless you can prove how it's exploitable
- The patch, or an equivalent one, must already be in master
  - Submitting to both at the same time is okay, but backporting is better

2.3.2 HOWTO submit to stable

Two ways:

1. Send patch to the skiboot-stable@lists.ozlabs.org list with "[PATCH <stable version>]" in subject
   - This targets the patch *ONLY* to the stable branch.
     - Such commits will *NOT* be merged into master.
   - Use this when:
     a. cherry-picking a fix from master
     b. fixing something that is only broken in stable
     c. fix in stable needs to be completely different than in master

     If b or c: explain why.
   - If cherry-picking, include the following at the top of your commit message:

     ```
     commit <sha1> upstream.
     ```
   - If the patch has been modified, explain why in description.
2. Add "Cc: skiboot-stable@lists.ozlabs.org" above your Signed-off-by line when sending to skiboot@
   - This targets the patch to master and stable.
   - You can target a patch to a specific stable tree with:

     ```
     Cc: skiboot-stable@lists.ozlabs.org # 5.1.x
     ```

     and that will target it to the 5.1.x branch.
   - You can ask for prerequisites to be cherry-picked:

```
Cc: skiboot-stable@lists.ozlabs.org # 5.1.x 55ae15b Ensure we run pollers in␣
↪cpu_wait_job()
Cc: skiboot-stable@lists.ozlabs.org # 5.1.x
```

Which means:

1. please `git cherry-pick 55ae15b`
2. then apply this patch to 5.1.x".

2.3.3 Trees

- https://github.com/open-power/skiboot/ (or via ssh at `git@github.com:open-power/skiboot.git`)
  - (branches are skiboot-X.Y.x - e.g. skiboot-5.1.x)
- Some stable versions may last longer than others
  - So there may be skiboot-5.1.x and skiboot-5.2.x actively maintained and skiboot-5.1.x could possibly outlast skiboot-5.2.x

2.4 Versioning Scheme of skiboot

2.4.1 History

For roughly the first six months of public life, skiboot just presented a git SHA1 as a version "number". This was "user visible" in two places:

1. `/sys/firmware/opal/msglog` the familiar `SkiBoot 71664fd-dirty starting...` message
2. device tree: `/proc/device-tree/ibm,opal/firmware/git-id`

Builds were also referred to by date and by corresponding PowerKVM release. Clearly, this was unlikely to be good practice going forward.

As of skiboot-4.0, this scheme has changed and we now present a version string instead. This better addresses the needs of everybody who is building OpenPower systems.

2.4.2 Current practice

The version string is constructed from a few places and is designed to be *highly* informative about what you're running. For the most part, it should be automatically constructed by the skiboot build system. The only times you need to do something is if you are a) making an upstream skiboot release or b) building firmware to release for your platform(s).

OPAL/skiboot has several consumers, for example:

- IBM shipping POWER8 systems with an FSP (FW810.XX and future)
- OpenPower
- OpenPower partners manufacturing OpenPower systems
- developers, test and support needing to understand what code a system is running

and there are going to be several concurrent maintained releases in the wild, likely build by different teams of people at different companies.

tl;dr; is you're likely going to see version numbers like this (for the hypothetical platforms 'ketchup' and 'mustard'):

- `skiboot-4.0-ketchup-0`
- `skiboot-4.0-ketchup-1`
- `skiboot-4.1-mustard-4`
- `skiboot-4.1-ketchup-0`

If you see *extra* things on the end of the version, then you're running a custom build from a developer (e.g. `skiboot-4.0-1-g23f147e-stewart-dirty-f42fc40` means something to us - explained below).

If you see less, for example `skiboot-4.0`, then you're running a build directly out of the main git tree. Those producing OPAL builds for users must *not* ship like this, even if the tree is identical.

Here are the components of the version string from master:

```
skiboot-4.0-1-g23f147e-debug-occ-stewart-dirty-f42fc40
^       ^^^ ^  ^^^^^^^ ^_______^    ^      ^   ^^^^^^^
|        |  |     |        |        |      |      |
|        |  |     |        |         \    /        - 'git diff|sha1sum'
|        |  |     |        |          \  /
|        |  |     |        |            - built from a dirty tree of $USER
|        |  |     |        |
|        |  |     |         - $EXTRA_VERSION (optional)
|        |  |     |
|        |  |      - git SHA1 of commit built
|        |  |
|        |   - commits head of skiboot-4.0 tag
|        |
|         - skiboot version number ---\
|                                      >--  from  the 'skiboot-4.0' git tag
 - product name (always skiboot)   ---/
```

When doing a release for a particular platform, you are expected to create and tag a branch from master. For the (hypothetical) ketchup platform which is going to do a release based on skiboot-4.0, you would create a tag 'skiboot-4.0-ketchup-0' pointing to the same revision as the 'skiboot-4.0' tag and then make any additional modifications to skiboot that were not in the 4.0 release. So, you could ship a skiboot with the following version string:

```
skiboot-4.0-ketchup-1
^       ^^^ ^       ^
|        |  |       |
|        |  |        - revision for this platform
|        |  |
|        |  |
|        |   - Platform name/version
|        |
|         - skiboot version number
|
 - product name (always skiboot)
```

This version string tells your users to expect what is in skiboot-4.0 plus some revisions for your platform.

2.4.3 Practical Considerations

You MUST correctly tag your git tree for sensible version numbers to be generated. Look at the (generated) version.c file to confirm you're building the correct version number. You will need annotated tags (git tag -a).

If your build infrastructure does *not* build skiboot from a git tree, you should specify SKIBOOT_VERSION as an environment variable (following this versioning scheme), otherwise the build will fail.

CHAPTER THREE

DEVELOPER GUIDE AND INTERNALS

3.1 SkiBoot Console Log

Skiboot maintains a circular textual log buffer in memory.

It can be accessed using any debugging method that can peek at memory contents. While the debug_descriptor does hold the location of the memory console, we're pretty keen on keeping its location static.

Events are logged in the following format: `[S.T,L] message` where:

S Seconds, which is the timebase divided by 512,000,000. **NOTE**: The timebase is reset during boot, so zero is a few dozen messages into skiboot booting.

T Remaining Timebase. It is *NOT* a fraction of a second, but rather timebase%512000000

L Log level (see below)

Example:

```
[    2.223466021,5] FLASH: Found system flash: Macronix MXxxL51235F id:0
[    3.494892796,7] FLASH: flash subpartition eyecatcher CAPP
```

You should use the new prlog() call for any log message and set the log level/priority appropriately.

printf() is mapped to PR_PRINTF and should be phased out and replaced with prlog() calls.

See timebase.h for full timebase explanation.

3.1.1 Log levels

Define	Value
PR_EMERG	0
PR_ALERT	1
PR_CRIT	2
PR_ERR	3
PR_WARNING	4
PR_NOTICE	5
PR_PRINTF	PR_NOTICE
PR_INFO	6
PR_DEBUG	7
PR_TRACE	8
PR_INSANE	9

The console_log_levels byte in the debug_descriptor controls what messages are written to any console drivers (e.g. fsp, uart) and what level is just written to the in memory console (or not at all).

This enables (advanced) users to vary what level of output they want at runtime in the memory console and through console drivers (fsp/uart)

You can vary two things by poking in the debug descriptor:

1. what log level is printed at all e.g. only turn on PR_TRACE at specific points during runtime
2. what log level goes out the fsp/uart console, defaults to PR_PRINTF

We use two 4bit numbers (1 byte) for this in debug descriptor (saving some space, not needlessly wasting space that we may want in future).

The default is 0x75 (7=PR_DEBUG to in memory console, 5=PR_PRINTF to drivers

If you write 0x77 you will get debug info on uart/fsp console as well as in memory. If you write 0x95 you get PR_INSANE in memory but still only PR_NOTICE through drivers.

People who write something like 0x1f will get a very quiet boot indeed.

3.1.2 Debugging

You can change the log level of what goes to the in memory buffer and whta goes to the driver (i.e. serial port / IPMI Serial over LAN) at boot time by setting NVRAM variables:

```
nvram -p ibm,skiboot --update-config log-level-driver=7
nvram -p ibm,skiboot --update-config log-level-memory=7
```

You can also use the named versions of emerg, alert, crit, err, warning, notice, printf, info, debug, trace or insane. ie.

```
nvram -p ibm,skiboot --update-config log-level-driver=insane
```

You an also write to the debug_descriptor to change it at runtime.

3.2 How to log errors on OPAL

Currently the errors reported by OPAL interfaces are in free form, where as errors reported by service processor is in standard Platform Error Log (PEL) format. For out-of band management via IPMI interfaces, it is necessary to push down the errors to service processor via mailbox (reported by OPAL) in PEL format.

PEL size can vary from 2K-16K bytes, fields of which needs to populated based on the kind of event and error that needs to be reported. All the information needed to be reported as part of the error, is passed by user using the error-logging interfaces outlined below. Following which, PEL structure is generated based on the input and then passed on to service processor.

We do create eSEL error log format for some service processors but it's just a wrapper around PEL format. Actual data still stays in PEL format.

3.2.1 Error logging interfaces in OPAL

Interfaces are provided for the user to log/report an error in OPAL. Using these interfaces relevant error information is collected and later converted to PEL format and then pushed to service processor.

Step 1: To report an error, invoke `opal_elog_create()` with required argument.

```
struct errorlog *opal_elog_create(struct opal_err_info *e_info,
uint32_t tag);
```

Parameters:

- `struct opal_err_info *e_info` Struct to hold information identifying error/event source.
- **`uint32_t tag`: Unique value to identify the data.** Ideal to have ASCII value for 4-byte string.

The opal_err_info struct holds several pieces of information to help identify the error/event. The struct can be obtained via the `DEFINE_LOG_ENTRY` macro as below - it only needs to be called once.

```
DEFINE_LOG_ENTRY(OPAL_RC_ATTN, OPAL_PLATFORM_ERR_EVT, OPAL_CHIP,
                 OPAL_PLATFORM_FIRMWARE, OPAL_PREDICTIVE_ERR_GENERAL,
                 OPAL_NA);
```

The various attributes set by this macro are described below.

`uint8_t opal_error_event_type`: Classification of error/events type reported on OPAL.

```
/* Platform Events/Errors: Report Machine Check Interrupt */
#define OPAL_PLATFORM_ERR_EVT           0x01
/* INPUT_OUTPUT: Report all I/O related events/errors */
#define OPAL_INPUT_OUTPUT_ERR_EVT       0x02
/* RESOURCE_DEALLOC: Hotplug events and errors */
#define OPAL_RESOURCE_DEALLOC_ERR_EVT   0x03
/* MISC: Miscellaneous error */
#define OPAL_MISC_ERR_EVT               0x04
```

`uint16_t component_id`: Component ID of OPAL component as listed in `include/errorlog.h`.

`uint8_t subsystem_id`: ID of the sub-system reporting error.

```
/* OPAL Subsystem IDs listed for reporting events/errors */
#define OPAL_PROCESSOR_SUBSYSTEM        0x10
#define OPAL_MEMORY_SUBSYSTEM           0x20
#define OPAL_IO_SUBSYSTEM               0x30
#define OPAL_IO_DEVICES                 0x40
#define OPAL_CEC_HARDWARE               0x50
#define OPAL_POWER_COOLING              0x60
#define OPAL_MISC                       0x70
#define OPAL_SURVEILLANCE_ERR           0x7A
#define OPAL_PLATFORM_FIRMWARE          0x80
#define OPAL_SOFTWARE                   0x90
#define OPAL_EXTERNAL_ENV               0xA0
```

`uint8_t event_severity`: Severity of the event/error to be reported.

```
#define OPAL_INFO                                   0x00
#define OPAL_RECOVERED_ERR_GENERAL                  0x10

/* 0x2X series is to denote set of Predictive Error */
/* 0x20 Generic predictive error */
#define OPAL_PREDICTIVE_ERR_GENERAL                         0x20
/* 0x21 Predictive error, degraded performance */
#define OPAL_PREDICTIVE_ERR_DEGRADED_PERF                   0x21
/* 0x22 Predictive error, fault may be corrected after reboot */
#define OPAL_PREDICTIVE_ERR_FAULT_RECTIFY_REBOOT            0x22
```

(continues on next page)

(continued from previous page)

```
/*
 * 0x23 Predictive error, fault may be corrected after reboot,
 * degraded performance
 */
#define OPAL_PREDICTIVE_ERR_FAULT_RECTIFY_BOOT_DEGRADE_PERF 0x23
/* 0x24 Predictive error, loss of redundancy */
#define OPAL_PREDICTIVE_ERR_LOSS_OF_REDUNDANCY              0x24

/* 0x4X series for Unrecoverable Error */
/* 0x40 Generic Unrecoverable error */
#define OPAL_UNRECOVERABLE_ERR_GENERAL                      0x40
/* 0x41 Unrecoverable error bypassed with degraded performance */
#define OPAL_UNRECOVERABLE_ERR_DEGRADE_PERF                 0x41
/* 0x44 Unrecoverable error bypassed with loss of redundancy */
#define OPAL_UNRECOVERABLE_ERR_LOSS_REDUNDANCY              0x44
/* 0x45 Unrecoverable error bypassed with loss of redundancy
 * and performance
 */
#define OPAL_UNRECOVERABLE_ERR_LOSS_REDUNDANCY_PERF         0x45
/* 0x48 Unrecoverable error bypassed with loss of function */
#define OPAL_UNRECOVERABLE_ERR_LOSS_OF_FUNCTION             0x48

#define OPAL_ERROR_PANIC                                    0x50
```

`uint8_t event_subtype`: Event Sub-type

```
#define OPAL_NA                                     0x00
#define OPAL_MISCELLANEOUS_INFO_ONLY                0x01
#define OPAL_PREV_REPORTED_ERR_RECTIFIED            0x10
#define OPAL_SYS_RESOURCES_DECONFIG_BY_USER         0x20
#define OPAL_SYS_RESOURCE_DECONFIG_PRIOR_ERR        0x21
#define OPAL_RESOURCE_DEALLOC_EVENT_NOTIFY          0x22
#define OPAL_CONCURRENT_MAINTENANCE_EVENT           0x40
#define OPAL_CAPACITY_UPGRADE_EVENT                 0x60
#define OPAL_RESOURCE_SPARING_EVENT                 0x70
#define OPAL_DYNAMIC_RECONFIG_EVENT                 0x80
#define OPAL_NORMAL_SYS_PLATFORM_SHUTDOWN           0xD0
#define OPAL_ABNORMAL_POWER_OFF                     0xE0
```

`uint8_t opal_srctype`: SRC type, value should be OPAL_SRC_TYPE_ERROR. SRC refers to System Reference Code. It is 4 byte hexa-decimal number that reflects the current system state. Eg: BB821010,

- 1st byte -> BB -> SRC Type
- 2nd byte -> 82 -> Subsystem
- 3rd, 4th byte -> Component ID and Reason Code

SRC needs to be generated on the fly depending on the state of the system. All the parameters needed to generate a SRC should be provided during reporting of an event/error.

`uint32_t reason_code`: Reason for failure as stated in `include/errorlog.h` for OPAL. Eg: Reason code for code-update failures can be

- `OPAL_RC_CU_INIT` -> Initialisation failure
- `OPAL_RC_CU_FLASH` -> Flash failure

Step 2: Data can be appended to the user data section using the either of the below two interfaces:

```
void log_append_data(struct errorlog *buf, unsigned char *data,
                     uint16_t size);
```

Parameters:

`struct opal_errorlog *buf`: struct opal_errorlog pointer returned by `opal_elog_create()` call.

`unsigned char *data`: Pointer to the dump data

`uint16_t size`: Size of the dump data.

`void log_append_msg(struct errorlog *buf, const char *fmt, ...);`

Parameters:

`struct opal_errorlog *buf`: pointer returned by `opal_elog_create()` call.

`const char *fmt`: Formatted error log string.

Additional user data sections can be added to the error log to separate data (eg. readable text vs binary data) by calling `log_add_section()`. The interfaces in Step 2 operate on the 'last' user data section of the error log.

`void log_add_section(struct errorlog *buf, uint32_t tag);`

Parameters:

`struct opal_errorlog *buf`: pointer returned by `opal_elog_create()` call.

`uint32_t tag`: Unique value to identify the data. Ideal to have ASCII value for 4-byte string.

Step 3: There is a platform hook for the OPAL error log to be committed on any service processor(Currently used for FSP and BMC based machines).

Below is snippet of the code of how this hook is called.

```
void log_commit(struct errorlog *elog)
{
        ....
        ....
        if (platform.elog_commit) {
                rc = platform.elog_commit(elog);
                if (rc)
                        prerror("ELOG: Platform commit error %d"
                                "\n", rc);
                return;
        }
        ....
        ....
}
```

Step 3.1 FSP:

```
.elog_commit            = elog_fsp_commit
```

Once all the data for an error is logged in, the error needs to be committed in FSP.

In the process of committing an error to FSP, log info is first internally converted to PEL format and then pushed to the FSP. FSP then take cares of sending all logs including its own and OPAL's one to the POWERNV.

OPAL maintains timeout field for all error logs it is sending to FSP. If it is not logged within allotted time period (e.g if FSP is down), in that case OPAL sends those logs to POWERNV.

Step 3.2 BMC:

```
.elog_commit            = ipmi_elog_commit
```

In case of BMC machines, error logs are first converted to eSEL format. i.e:

```
eSEL = SEL header + PEL data
```

SEL header contains below fields.

```
struct sel_header {
        uint16_t id;
        uint8_t record_type;
        uint32_t timestamp;
        uint16_t genid;
        uint8_t evmrev;
        uint8_t sensor_type;
        uint8_t sensor_num;
        uint8_t dir_type;
        uint8_t signature;
        uint8_t reserved[2];
}
```

After filling up the SEL header fields, OPAL copies the error log PEL data after the header section in the error log buffer. Then using IPMI interface, eSEL gets logged in BMC.

If the user does not intend to dump various user data sections, but just log the error with some amount of description around that error, they can do so using just the simple error logging interface.

```
log_simple_error(uint32_t reason_code, char *fmt, ...);
```

For example:

```
log_simple_error(OPAL_RC_SURVE_STATUS,
                "SURV: Error retrieving surveillance status: %d\n",
                                                        err_len);
```

Using the reason code, an error log is generated with the information derived from the look-up table, populated and committed to service processor. All of it is done with just one call.

3.3 Error logging retrieval from FSP:

FSP sends error log notification to OPAL via mailbox protocol.

OPAL maintains below lists:

- Free list : List of free nodes.
- Pending list : List of nodes which is yet to be read by the POWERNV.
- Processed list : List of nodes which has been read but still waiting for acknowledgement.

Below is the structure of the node:

```
struct fsp_log_entry {
        uint32_t log_id;
        size_t log_size;
```

(continues on next page)

(continued from previous page)

```
        struct list_node link;
};
```

OPAL maintains a state machine which has following states.

```
enum elog_head_state {
        ELOG_STATE_FETCHING,    /*In the process of reading log from FSP. */
        ELOG_STATE_FETCHED_INFO,/* Indicates reading log info is completed */
        ELOG_STATE_FETCHED_DATA,/* Indicates reading log is completed */
        ELOG_STATE_HOST_INFO,   /* Host read log info */
        ELOG_STATE_NONE,        /* Indicates to fetch next log */
        ELOG_STATE_REJECTED,    /* resend all pending logs to linux */
};
```

Initially, state of the state machine is `ELOG_STATE_NONE`. When OPAL gets the notification about the error log, it takes out the node from free list and put it into pending list and update the state machine to fetching state (`ELOG_STATE_FETCHING`). It also gives response back to FSP about the received error log notification.

It then queue mailbox message to get the error log data in OPAL error log buffer, once it is done state machine gets into fetched state (`ELOG_STATE_FETCHED_DATA`). After that, OPAL notifies POWERNV host to fetch new error log.

POWERNV uses the OPAL interface to get the error log info(elogid, elog_size, elog_type) first then it reads the error log data in its buffer that moves the pending error log to processed list. After reading, the state machine moves to `ELOG_STATE_NONE` state.

It acknowledges the error log id after reading error log data by sending the call to OPAL, which in turn sends the acknowledgement mbox message to FSP and moves error log id from processed list to again back to free node list and this process goes on every FSP error log.

3.3.1 Design constraints:

```
#define ELOG_READ_MAX_RECORD            128
```

Currently, the number of error logs from FSP, OPAL can hold is limited to 128. If OPAL run out of free node in the list for the new error log, it sends 'Discarded by OPAL' message to the FSP. At some point in the future, it is upto FSP when it notifies again to OPAL about the discarded error log.

```
#define ELOG_WRITE_MAX_RECORD           64
```

There is also limitation on the number of OPAL error logs OPAL can hold is 64. If it is run out of the buffers in the pool, it will log the message saying 'Failed to get the buffer'.

3.3.2 Note

- For more information regarding error logging and PEL format refer to PAPR doc and P7 PEL and SRC PLDD document.
- Refer to `include/errorlog.h` for all the error logging interface parameters and `include/pel.h` for PEL structures.

3.3.3 Sample error logging

```
DEFINE_LOG_ENTRY(OPAL_RC_ATTN, OPAL_PLATFORM_ERR_EVT, OPAL_ATTN,
               OPAL_PLATFORM_FIRMWARE, OPAL_PREDICTIVE_ERR_GENERAL,
               OPAL_NA);

void report_error(int index)
{
      struct errorlog *buf;
      char data1[] = "This is a sample user defined data section1";
      char data2[] = "Error logging sample. These are dummy errors. Section 2";
      char data3[] = "Sample error Sample error Sample error Sample error \
                      Sample error abcdefghijklmnopqrstuvwxyz";
      int tag;

      printf("ELOG: In machine check report error index: %d\n", index);

      /* To report an error, create an error log with relevant information
       * opal_elog_create(). Call returns a pre-allocated buffer of type
       * 'struct errorlog' buffer with relevant fields updated.
       */

      /* tag -> unique ascii tag to identify a particular data dump section */
      tag = 0x4b4b4b4b;
      buf = opal_elog_create(&e_info(OPAL_RC_ATTN), tag);
      if (buf == NULL) {
              printf("ELOG: Error getting buffer.\n");
              return;
      }

      /* Append data or text with log_append_data() or log_append_msg() */
      log_append_data(buf, data1, sizeof(data1));

      /* In case of user wanting to add multiple sections of various dump data
       * for better debug, data sections can be added using this interface
       * void log_add_section(struct errorlog *buf, uint32_t tag);
       */
      tag = 0x4c4c4c4c;
      log_add_section(buf, tag);
      log_append_data(buf, data2, sizeof(data2));
      log_append_data(buf, data3, sizeof(data3));

      /* Once all info is updated, ready to be sent to FSP */
      printf("ELOG:commit to FSP\n");
      log_commit(buf);
}
```

3.3.4 Sample output PEL dump got from FSP

```
$ errl -d -x 0x533C9B37
|   00000000     50480030  01004154  20150728  02000500     PH.0..AT ..(....   |
|   00000010     20150728  02000566  4B000107  00000000      ..(...fK.......   |
|   00000020     00000000  00000000  B0000002  533C9B37     ............S..7   |
|   00000030     55480018  01004154  80002000  00000000     UH....AT.. .....   |
|   00000040     00002000  01005300  50530050  01004154     .. ...S.PS.P..AT   |
```

(continues on next page)

(continued from previous page)

```
|   00000050     02000008  00000048  00000080  00000000     .......H........   |
|   00000060     00000000  00000000  00000000  00000000     ................   |
|   00000070     00000000  00000000  42423832  31343130     ........BB821410   |
|   00000080     20202020  20202020  20202020  20202020                        |
|   00000090     20202020  20202020  4548004C  01004154             EH.L..AT   |
|   000000A0     38323836  2D343241  31303738  34415400     8286-42A10784AT.   |
|   000000B0     00000000  00000000  00000000  00000000     ................   |
|   000000C0     00000000  00000000  00000000  00000000     ................   |
|   000000D0     00000000  00000000  20150728  02000500     ........ ..(....   |
|   000000E0     00000000  4D54001C  01004154  38323836     ....MT....AT8286   |
|   000000F0     2D343241  31303738  34415400  00000000     -42A10784AT.....   |
|   00000100     5544003C  01004154  4B4B4B4B  00340000     UD....ATKKKK.4..   |
|   00000110     54686973  20697320  61207361  6D706C65     This is a sample   |
|   00000120     20757365  72206465  66696E65  64206461      user defined da   |
|   00000130     74612073  65637469  6F6E3100  554400A7     ta section1.UD..   |
|   00000140     01004154  4C4C4C4C  009F0000  4572726F     ..ATLLLL....Erro   |
|   00000150     72206C6F  6767696E  67207361  6D706C65     r logging sample   |
|   00000160     2E205468  65736520  61726520  64756D6D     . These are dumm   |
|   00000170     79206572  726F7273  2E205365  6374696F     y errors. Sectio   |
|   00000180     6E203200  53616D70  6C652065  72726F72     n 2.Sample error   |
|   00000190     2053616D  706C6520  6572726F  72205361      Sample error Sa   |
|   000001A0     6D706C65  20657272  6F722053  616D706C     mple error Sampl   |
|   000001B0     65206572  726F7220  09090953  616D706C     e error ...Sampl   |
|   000001C0     65206572  726F7220  61626364  65666768     e error abcdefgh   |
|   000001D0     696A6B6C  6D6E6F70  71727374  75767778     ijklmnopqrstuvwx   |
|   000001E0     797A00                                     yz.                |
|------------------------------------------------------------------------------|
|                       Platform Event Log - 0x533C9B37                        |
|------------------------------------------------------------------------------|
|                                Private Header                                |
|------------------------------------------------------------------------------|
| Section Version          : 1                                                 |
| Sub-section type         : 0                                                 |
| Created by               : 4154                                              |
| Created at               : 07/28/2015 02:00:05                               |
| Committed at             : 07/28/2015 02:00:05                               |
| Creator Subsystem        : OPAL                                              |
| CSSVER                   :                                                   |
| Platform Log Id          : 0xB0000002                                        |
| Entry Id                 : 0x533C9B37                                        |
| Total Log Size           : 483                                               |
|------------------------------------------------------------------------------|
|                                 User Header                                  |
|------------------------------------------------------------------------------|
| Section Version          : 1                                                 |
| Sub-section type         : 0                                                 |
| Log Committed by         : 4154                                              |
| Subsystem                : Platform Firmware                                 |
| Event Scope              : Unknown - 0x00000000                              |
| Event Severity           : Predictive Error                                  |
| Event Type               : Not Applicable                                    |
| Return Code              : 0x00000000                                        |
| Action Flags             : Report Externally                                 |
| Action Status            : Sent to Hypervisor                                |
|------------------------------------------------------------------------------|
|                        Primary System Reference Code                         |
|------------------------------------------------------------------------------|
```

(continues on next page)

(continued from previous page)

```
| Section Version          : 1                                                 |
| Sub-section type         : 0                                                 |
| Created by               : 4154                                              |
| SRC Format               : 0x80                                              |
| SRC Version              : 0x02                                              |
| Virtual Progress SRC     : False                                             |
| I5/OS Service Event Bit  : False                                             |
| Hypervisor Dump Initiated: False                                             |
| Power Control Net Fault  : False                                             |
|                                                                              |
| Valid Word Count         : 0x08                                              |
| Reference Code           : BB821410                                          |
| Hex Words 2 - 5          : 00000080 00000000 00000000 00000000               |
| Hex Words 6 - 9          : 00000000 00000000 00000000 00000000               |
|                                                                              |
|------------------------------------------------------------------------------|
|                            Extended User Header                              |
|------------------------------------------------------------------------------|
| Section Version          : 1                                                 |
| Sub-section type         : 0                                                 |
| Created by               : 4154                                              |
| Reporting Machine Type   : 8286-42A                                          |
| Reporting Serial Number  : 10784AT                                           |
| FW Released Ver          :                                                   |
| FW SubSys Version        :                                                   |
| Common Ref Time          : 07/28/2015 02:00:05                               |
| Symptom Id Len           : 0                                                 |
| Symptom Id               :                                                   |
|------------------------------------------------------------------------------|
|                     Machine Type/Model & Serial Number                       |
|------------------------------------------------------------------------------|
| Section Version          : 1                                                 |
| Sub-section type         : 0                                                 |
| Created by               : 4154                                              |
| Machine Type Model       : 8286-42A                                          |
| Serial Number            : 10784AT                                           |
|------------------------------------------------------------------------------|
|                             User Defined Data                                |
|------------------------------------------------------------------------------|
| Section Version          : 1                                                 |
| Sub-section type         : 0                                                 |
| Created by               : 4154                                              |
|                                                                              |
|   00000000     4B4B4B4B  00340000  54686973  20697320     KKKK.4..This is    |
|   00000010     61207361  6D706C65  20757365  72206465     a sample user de   |
|   00000020     66696E65  64206461  74612073  65637469     fined data secti   |
|   00000030     6F6E3100                                   on1.               |
|                                                                              |
|------------------------------------------------------------------------------|
|                             User Defined Data                                |
|------------------------------------------------------------------------------|
| Section Version          : 1                                                 |
| Sub-section type         : 0                                                 |
| Created by               : 4154                                              |
|                                                                              |
|   00000000     4C4C4C4C  009F0000  4572726F  72206C6F     LLLL....Error lo   |
|   00000010     6767696E  67207361  6D706C65  2E205468     gging sample. Th   |
```

(continues on next page)

(continued from previous page)

```
|   00000020     65736520  61726520  64756D6D  79206572     ese are dummy er   |
|   00000030     726F7273  2E205365  6374696F  6E203200     rors. Section 2.   |
|   00000040     53616D70  6C652065  72726F72  2053616D     Sample error Sam   |
|   00000050     706C6520  6572726F  72205361  6D706C65     ple error Sample   |
|   00000060     20657272  6F722053  616D706C  65206572      error Sample er   |
|   00000070     726F7220  09090953  616D706C  65206572     ror ...Sample er   |
|   00000080     726F7220  61626364  65666768  696A6B6C     ror abcdefghijkl   |
|   00000090     6D6E6F70  71727374  75767778  797A00       mnopqrstuvwxyz.    |
|                                                                              |
|------------------------------------------------------------------------------|
```

3.4 OPAL <–> BMC interactions

This document provides information about some of the user-visible interactions that skiboot performs with the BMC.

3.4.1 IPMI sensors

OPAL will interact with a few IPMI sensors during the boot process. These are:

- Boot Count [type 0xc3: OEM reserved]
- FW Boot progress [type 0x0f: System Firmware Progress]

Boot Count: assertion type. When OPAL reaches a late stage of boot, it sets the boot count sensor to 0x02. This is intended to allow the BMC detect a failed or aborted boot, for switching to a known-good firmware image.

FW Boot Progress: assertion type. During boot, skiboot will update this sensor to one of the IPMI-defined progress codes. The codes use by skiboot are:

- **PCI Resource configuration (0x01)**
  - asserted as the PCI devices have been probed and resources allocated
- **Motherboard init (0x14)**
  - asserted as the platform-specific components have been initialised
- **OS boot (0x13)**
  - asserted after skiboot has loaded the PAYLOAD image, and is about to boot it.

3.4.2 Chassis control messages

OPAL uses chassis control messages to instruct the BMC to remove power from the host. These messages are sent during graceful reboot and shutdown processes initiated by the host.

For a BMC-initiated graceful power-down (or reboot), the BMC is expected to send an OEM-defined SEL message, using a SMS_ATN to trigger a BMC-to-host notification. This SEL has a type of 0xc0, and command of 0x04. The data0 field of the SEL indicates shutdown (0x0) or reboot (0x1).

3.4.3 Watchdog support

OPAL supports a BMC watchdog during the boot process. This will be disabled before entering the OS.

3.4.4 Real-time clock

On platforms where a real-time-clock is not available, skiboot may use the IPMI SEL Time as a real-time-clock device.

3.4.5 SBE validation

On some P8 platforms with an AMI or SMC BMC (ie. astbmc) SBE validation is done by a tool on the BMC. This is done to inspect the SBE and detect if a malicious host has written to the SBE, especially in multi-tenant "Bare-Metal-As-A-Service" scenarios.

To complicate this the SBE validation occurs at host-runtime and reads the SBE SEEPROM over I2C using the FSI master which will conflict with anything the host may be doing at the same time. To avoid this Skiboot will pause boot until the validation is complete. If SBE validation is required the BMC will communicate this to Skiboot by setting an IPMI System Boot Option with OEM parameter 0x62. When this flag is set Skiboot will pause and wait for the validation to complete and the flag to be cleared. This ensures the validation completes before the execution is passed to Petitboot and the host operating system and any conflicts could occur. During this process Skiboot will print

> SBE validation required, waiting for completion System will be powered off if validation fails

to the console with an update every minute until complete.

Unfortunately the validation performed by the BMC leaves the SBE in a bad state. Once the validation is complete Skiboot will reboot to reset everything to a good state and normal booting can resume. No such reboot is required if the flag is not set and validation doesn't occur.

3.5 GCOV for skiboot

3.5.1 Unit tests

All unit tests are built+run with gcov enabled.

```
make coverage-report
```

will generate a unit test coverage report like: http://open-power.github.io/skiboot/coverage-report/

3.5.2 Skiboot

You can now build Skiboot itself with gcov support, boot it on a machine, do things, and then extract out gcda files to generate coverage reports from real hardware (or a simulator).

3.5.3 Building Skiboot with GCOV

```
SKIBOOT_GCOV=1 make
```

You may need to `make clean` first.

This will build a skiboot lid roughly *twice* the size.

Flash/Install the skiboot.lid and boot.

3.5.4 Extracting GCOV data

The way we extract the gcov data from a system is by dumping the contents of skiboot memory and then parsing the data structures in user space with the extract-gcov utility in the skiboot repo.

mambo:

```
mysim memory fwrite 0x30000000 0x240000 skiboot.dump
```

FSP:

```
getmemproc 30000000 3407872 -fb skiboot.dump
```

linux (e.g. petitboot environment):

```
dd if=/proc/kcore skip=1572864 count=6656  of=skiboot.dump
```

You basically need to dump out the first 3MB of skiboot memory.

Then you need to find out where the gcov data structures are:

```
perl -e "printf '0x%x', 0x30000000 + 0x`grep gcov_info_list skiboot.map|cut -f 1 -d '
↪'`"
```

That address needs to be supplied to the extract-gcov utility:

```
./extract-gcov skiboot.dump 0x3023ec40
```

Once you've run extract-gcov, it will have extracted the gcda files from the skiboot memory image.

You can then run lcov:

```
lcov -b . -q -c -d . -o skiboot-boot.info
--gcov-tool
/opt/cross/gcc-4.8.0-nolibc/powerpc64-linux/bin/powerpc64-linux-gcov
```

IMPORTANT you should point lcov to the gcov for the compiler you used to build skiboot, otherwise you're likely to get errors.

3.6 Memory in skiboot

There are regions of memory we statically allocate for firmware as well as a HEAP region for boot and runtime allocations.

A design principle of skiboot is to attempt not to allocate memory at runtime, or at least keep it to a minimum, and not do so in any critical code path for the system to remain running.

At no point during runtime should a skiboot memory allocation failure cause the system to stop functioning.

3.6.1 HEAP

Dynamic memory allocations go in a single heap. This is identified as Region ibm,firmware-heap and appears as a reserved section in the device tree.

Originally, it was 12582912 bytes in size (declared in mem_map.h). Now, it is 13631488 bytes after being bumped as part of the GCOV work.

We increased heap size as on larger systems, we were getting close to using all the heap once skiboot became 2MB with GCOV.

Heap usage is printed before running the payload.

For example, as of writing, on a dual socket Tuleta:

```
[45215870591,5] SkiBoot skiboot-5.0.1-94-gb759ce2 starting...
[3680939340,5] CUPD: T side MI Keyword = SV830_027
[3680942658,5] CUPD: T side ML Keyword = FW830.00
[15404383291,5] Region ibm,firmware-heap free: 5378072
```

and on a palmetto:

```
[24748502575,5] SkiBoot skiboot-5.0.1-94-gb759ce2 starting...
[9870429550,5] Region ibm,firmware-heap free: 10814856
```

Our memory allocator is simple, a use pattern of:

```
A = malloc();
B = malloc();
free(A);
```

is likely to generate fragmentation, so it should generally be avoided where possible.

3.7 OPAL/Skiboot Nvlink Interface Documentation

3.7.1 Overview

NV-Link is a high speed interconnect that is used in conjunction with a PCI-E connection to create an interface between chips that provides very high data bandwidth. The PCI-E connection is used as the control path to initiate and report status of large data transfers. The data transfers themselves are sent over the NV-Link.

On IBM Power systems the NV-Link hardware is similar to our standard PCI hardware so to maximise code reuse the NV-Link is exposed as an emulated PCI device through system firmware (OPAL/skiboot). Thus each NV-Link capable device will appear as two devices on a system, the real PCI-E device and at least one emulated PCI device used for the NV-Link.

Presently the NV-Link is only capable of data transfers initiated by the target, thus the emulated PCI device will only handle registers for link initialisation, DMA transfers and error reporting (EEH).

3.7.2 Emulated PCI Devices

Each link will be exported as an emulated PCI device with a minimum of two emulated PCI devices per GPU. Emulated PCI devices are grouped per GPU.

The emulated PCI device will be exported as a standard PCI device by the Linux kernel. It has a standard PCI configuration space to expose necessary device parameters. The only functionality available is related to the setup of DMA windows.

Configuration Space Parameters

Vendor ID	0x1014	(IBM)
Device ID	0x04ea	
Revision ID	0x00	
Class	0x068000 / 0x068001	(Bridge Device Other, ProgIf = 0x0 / 0x1)
BAR0/1	TL/DL Registers	
BAR2/3	GEN-ID Registers	(Only for rev-id = 0x1)

TL/DL Registers

Each link has 128KB of TL/DL registers. These will always be mapped to 64-bit BAR#0 of the emulated PCI device configuration space.

```
BAR#0 + 128K +------------+
            | NTL (64K) |
BAR#0 + 64K  +------------+
            | DL (64K)  |
BAR#0       +------------+
```

Generation Registers (GEN-ID)

On POWER9 each link has 64K of generation ID registers for the relaxed ordering mode syncronisation. Refer to the programming guide for details of the register layout in this BAR.

Relaxed ordering mode will be disabled by default as it requires device driver support. Device drivers will need to request relaxed ordering mode through some yet to be designed mechanism.

Vendor Specific Capabilities

```
+-----------------+----------------+----------------+----------------+
|  Version (0x02) |   Cap Length   |  Next Cap Ptr  |  Cap ID (0x09) |
+-----------------+----------------+----------------+----------------+
|                      Procedure Status Register                     |
+--------------------------------------------------------------------+
|                      Procedure Control Register                    |
+---------------------------------------------------+----------------+
|             Reserved            |  PCI Dev Flag  |   Link Number  |
+---------------------------------------------------+----------------+
```

Version

This refers to the version of the NPU config space. Used by device drivers to determine which fields of the config space they can expect to be available.

Procedure Control Register

Used to start hardware procedures.

Writes will start the corresponding procedure and set bit 31 in the procedure status register. This register must not be written while bit 31 is set in the status register. Performing a write while another procudure is already in progress will abort that procedure.

Reads will return the in progress procedure or the last completed procedure number depending on the procedure status field.

Procedure Numbers:

0. Abort in-progress procedure
1. NOP
2. Unsupported procedure
3. Unsupported procedure
4. Naples PHY - RESET
5. Naples PHY - TX_ZCAL
6. Naples PHY - RX_DCCAL
7. Naples PHY - TX_RXCAL_ENABLE
8. Naples PHY - TX_RXCAL_DISABLE
9. Naples PHY - RX_TRAINING
10. Naples NPU - RESET
11. Naples PHY - PHY preterminate
12. Naples PHY - PHY terminated
13. Witherspoon TL credit validation

Procedure 5 (TX_ZCAL) should only be run once. System firmware will ensure this so device drivers may call this procedure mutiple times.

Procedure Status Register

The procedure status register is used to determine when execution of the procedure number in the control register is complete and if it completed successfully.

This register must be polled frequently to allow system firmware to execute the procedures.

Fields: Bit 31 - Procedure in progress Bit 30 - Procedure complete Bit 3-0 - Procedure completion code

Procedure completion codes: 0 - Procedure completed successfully. 1 - Transient failure. Procedure should be rerun. 2 - Permanent failure. Procedure will never complete successfully. 3 - Procedure aborted. 4 - Unsupported procedure.

PCI Device Flag

Bit 0 - set if the GPU PCIe device associated with this nvlink was found. bit 1 - set if the DL has been taken out of reset.

Link Number

Physical link number this emulated PCI device is associated with. One of 0, 1, 4 or 5 (links 2 & 3 do not exist on Naples).

Reserved

These fields must be ignored and no value should be assumed.

Interrupts

Each link has a single DL/TL interrupt assigned to it. These will be exposed as an LSI via the emulated PCI device. There are 4 links consuming 4 LSI interrupts. The 4 remaining interrupts supported by the corresponding PHB will be routed to OS platform for the purpose of error reporting.

3.7.3 Device Tree Bindings

See *Nvlink Device Tree Bindings*

3.8 PCI

3.8.1 Debugging

There exist a couple of NVRAM options for enabling extra debug functionality to help debug PCI issues. These are not ABI and may be changed or removed at **any** time.

Verbose EEH

```
nvram -p ibm,skiboot --update-config pci-eeh-verbose=true
```

Disable EEH MMIO

:: nvram -p ibm,skiboot –update-config pci-eeh-mmio=disabled

Check for RX errors after link training

Some PHB4 PHYs can get stuck in a bad state where they are constantly retraining the link. This happens transparently to skiboot and Linux but will causes PCIe to be slow. Resetting the PHB4 clears the problem.

We can detect this case by looking at the RX errors count where we check for link stability. This patch does this by modifying the link optimal code to check for RX errors. If errors are occurring we retrain the link irrespective of the chip rev or card.

Normally when this problem occurs, the RX error count is maxed out at 255. When there is no problem, the count is 0. We chose 8 as the max rx errors value to give us some margin for a few errors. There is also a knob that can be used to set the error threshold for when we should retrain the link. i.e.

```
nvram -p ibm,skiboot --update-config phb-rx-err-max=8
```

Retrain link if degraded

On P9 Scale Out (Nimbus) DD2.0 and Scale in (Cumulus) DD1.0 (and below) the PCIe PHY can lockup causing training issues. This can cause a degradation in speed or width in ~5% of training cases (depending on the card). This is fixed in later chip revisions. This issue can also cause PCIe links to not train at all, but this case is already handled.

There is code in skiboot that checks if the PCIe link has trained optimally and if not, does a full PHB reset (to fix the PHY lockup) and retrain.

One complication is some devices are known to train degraded unless device specific configuration is performed. Because of this, we only retrain when the device is in a whitelist. All devices in the current whitelist have been testing on a P9DSU/Boston, ZZ and Witherspoon.

We always gather information on the link and print it in the logs even if the card is not in the whitelist.

For testing purposes, there's an nvram to retry all PCIe cards and all P9 chips when a degraded link is detected. The new option is 'pci-retry-all=true' which can be set using:

```
nvram -p ibm,skiboot --update-config pci-retry-all=true
```

This option may increase the boot time if used on a badly behaving card.

Maximum link speed

Was useful during bringup on P9 DD1.

:: nvram -p ibm,skiboot –update-config pcie-max-link-speed=4

Ric Mata Mode

This mode (for PHB4) will trace the training process closely. This activates as soon as PERST is deasserted and produces human readable output of the process.

It will also add the PCIe Link Training and Status State Machine (LTSSM) tracing and details on speed and link width.

Output looks a bit like this

```
[    1.096995141,3] PHB#0000[0:0]: TRACE:0x0000001101000000  0ms          ␣
↪GEN1:x16:detect
[    1.102849137,3] PHB#0000[0:0]: TRACE:0x0000102101000000 11ms presence␣
↪GEN1:x16:polling
[    1.104341838,3] PHB#0000[0:0]: TRACE:0x0000182101000000 14ms training␣
↪GEN1:x16:polling
[    1.104357444,3] PHB#0000[0:0]: TRACE:0x00001c5101000000 14ms training␣
↪GEN1:x16:recovery
[    1.104580394,3] PHB#0000[0:0]: TRACE:0x00001c5103000000 14ms training␣
↪GEN3:x16:recovery
[    1.123259359,3] PHB#0000[0:0]: TRACE:0x00001c5104000000 51ms training␣
↪GEN4:x16:recovery
[    1.141737656,3] PHB#0000[0:0]: TRACE:0x0000144104000000 87ms presence GEN4:x16:L0
[    1.141752318,3] PHB#0000[0:0]: TRACE:0x0000154904000000 87ms trained  GEN4:x16:L0
[    1.141757964,3] PHB#0000[0:0]: TRACE: Link trained.
[    1.096834019,3] PHB#0001[0:1]: TRACE:0x0000001101000000  0ms          ␣
↪GEN1:x16:detect
[    1.105578525,3] PHB#0001[0:1]: TRACE:0x0000102101000000 17ms presence␣
↪GEN1:x16:polling
[    1.112763075,3] PHB#0001[0:1]: TRACE:0x0000183101000000 31ms training␣
↪GEN1:x16:config
[    1.112778956,3] PHB#0001[0:1]: TRACE:0x00001c5081000000 31ms training␣
↪GEN1:x08:recovery
[    1.113002083,3] PHB#0001[0:1]: TRACE:0x00001c5083000000 31ms training␣
↪GEN3:x08:recovery
[    1.114833873,3] PHB#0001[0:1]: TRACE:0x0000144083000000 35ms presence GEN3:x08:L0
[    1.114848832,3] PHB#0001[0:1]: TRACE:0x0000154883000000 35ms trained  GEN3:x08:L0
[    1.114854650,3] PHB#0001[0:1]: TRACE: Link trained.
```

Enabled via NVRAM:

```
nvram -p ibm,skiboot --update-config pci-tracing=true
```

Named after the person the output of this mode is typically sent to.

WARNING: The documentation below **urgently needs updating** and is *woefully* incomplete.

3.8.2 IODA PE Setup Sequences

(**WARNING**: this was rescued from old internal documentation. Needs verification)

To setup basic PE mappings, the host performs this basic sequence:

For ibm,opal-ioda2, prior to allocating PHB resources to PEs, the host must allocate memory for PE structures and then calls `opal_pci_set_phb_table_memory( phb_id, rtt_addr, ivt_addr, ivt_len, rrba_addr, peltv_addr)` to define them to the PHB. OPAL returns `OPAL_UNSUPPORTED` status for `ibm, opal-ioda` PHBs.

The host calls `opal_pci_set_pe( phb_id, pe_number, bus, dev, func, validate_mask, bus_mask, dev_mask, func mask)` to map a PE to a PCI RID or range of RIDs in the same PE domain.

The host calls `opal_pci_set_peltv(phb_id, parent_pe, child_pe, state)` to set a parent PELT vector bit for the child PE argument to 1 (a child of the parent) or 0 (not in the parent PE domain).

3.8.3 IODA MMIO Setup Sequences

(**WARNING**: this was rescued from old internal documentation. Needs verification)

The host calls `opal_pci_phb_mmio_enable( phb_id, window_type, window_num, 0x0)` to disable the MMIO window.

The host calls `opal_pci_set_phb_mmio_window( phb_id, mmio_window, starting_real_address, starting_pci_address, segment_size)` to change the MMIO window location in PCI and/or processor real address space, or to change the size – and corresponding window size – of a particular MMIO window.

The host calls `opal_pci_map_pe_mmio_window( pe_number, mmio_window, segment_number)` to map PEs to window segments, for each segment mapped to each PE.

The host calls `opal_pci_phb_mmio_enable( phb_id, window_type, window_num, 0x1)` to enable the MMIO window.

3.8.4 IODA MSI Setup Sequences

(**WARNING**: this was rescued from old internal documentation. Needs verification)

To setup MSIs:

1. For ibm,opal-ioda PHBs, the host chooses an MVE for a PE to use and calls `opal_pci_set_mve( phb_id, mve_number, pe_number,)` to setup the MVE for the PE number. HAL treats this call as a NOP and returns hal_success status for ibm,opal-ioda2 PHBs.
2. The host chooses an XIVE to use with a PE and calls a. `opal_pci_set_xive_pe( phb_id, xive_number, pe_number)` to authorize that PE to signal that XIVE as an interrupt. The host must call this function for each XIVE assigned to a particular PE, but may use this call for all XIVEs prior to calling `opel_pci_set_mve()` to bind the PE XIVEs to an MVE. For MSI conventional, the host must bind a unique MVE for each sequential set of 32 XIVEs. b. The host forms the interrupt_source_number from the combination of the device tree MSI property base BUID and XIVE number, as

an input to `opal_set_xive(interrupt_source_number, server_number, priority)` and `opal_get_xive(interrupt_source_number, server_number, priority)` to set or return the server and priority numbers within an XIVE. c. `opal_get_msi_64[32](phb_id, mve_number, xive_num, msi_range, msi_address, message_data)` to determine the MSI DMA address (32 or 64 bit) and message data value for that xive.

> For MSI conventional, the host uses this for each sequential power of 2 set of 1 to 32 MSIs, to determine the MSI DMA address and starting message data value for that MSI range. For MSI-X, the host calls this uniquely for each MSI interrupt with an msi_range input value of 1.

3. For `ibm,opal-ioda` PHBs, once the MVE and XIVRs are setup for a PE, the host calls `opal_pci_set_mve_enable( phb_id, mve_number, state)` to enable that MVE to be a valid target of MSI DMAs. The host may also call this function to disable an MVE when changing PE domains or states.

3.8.5 IODA DMA Setup Sequences

(**WARNING**: this was rescued from old internal documentation. Needs verification)

To Manage DMA Windows :

1. The host calls `opal_pci_map_pe_dma_window( phb_id, dma_window_number, pe_number, tce_levels, tce_table_addr, tce_table_size, tce_page_size, utin64_t* pci_start_addr )` to setup a DMA window for a PE to translate through a TCE table structure in KVM memory.
2. The host calls `opal_pci_map_pe_dma_window_real( phb_id, dma_window_number, pe_number, mem_low_addr, mem_high_addr)` to setup a DMA window for a PE that is translated (but validated by the PHB as an untranlsated address space authorized to this PE).

3.8.6 Device Tree Bindings

See *PCI Device Tree Bindings* for device tree information.

3.9 PCI Slots

The PCI slots are instantiated to represent their associated properties and operations. The slot properties are exported to OS through the device tree node of the corresponding parent PCI device. The slot operations are used to accomodate requests from OS regarding the indicated PCI slot:

- PCI slot reset
- PCI slot property retrival

The PCI slots are expected to be created by individual platforms based on the given templates, which are classified to PHB slot or normal one currently. The PHB slot is instantiated based on PHB types like P7IOC and PHB3. However, the normal PCI slots are created based on general RC (Root Complex), PCIE switch ports, PCIE-to-PCIx bridge. Individual platform may create PCI slot, which doesn't have existing template.

The PCI slots are created at different stages according to their types. PHB slots are expected to be created once the PHB is register (struct platform::pci_setup_phb()) because the PHB slot reset operations are required at early stage of PCI enumeration. The normal slots are populated after their parent PCI devices are instantiated at struct platform::pci_get_slot_info().

The operation set supplied by the template might be overrided and reimplemented, or partially. It's usually done according to the VPD figured out by individual platforms.

3.9.1 PCI Slot Operations

The following operations are supported to one particular PCI slot. More details could be found from the definition of struct pci_slot_ops:

Operation	Definition
get_presence_state	Check if any adapter connected to slot
get_link_state	Retrieve PCIE link status: up, down, link width
get_power_state	Retrieve the power status: on, off
get_attention_state	Retrieve attention status: on, off, blinking
get_latch_state	Retrieve latch status
set_power_state	Configure the power status: on, off
set_attention_state	Configure attention status: on, off, blinking
prepare_link_change	Prepare PCIE link status change
poll_link	Poll PCIE link until it's up or down permanently
creset	Complete reset, only available to PHB slot
freset	Fundamental reset
hreset	Hot reset
poll	Interface for OPAL API to drive internal state machine
add_properties	Additional PCI slot properties seen by platform

3.10 PCI Slot Properties

The following PCI slot properties have been exported through PCI device tree node for a root port, a PCIE switch port, or a PCIE to PCIx bridge. If the individual platforms (e.g. Firenze and Apollo) have VPD for the PCI slot, they should extract the PCI slot properties from VPD and export them accordingly.

Property	Definition
ibm,reset-by-firmware	Boolean indicating whether the slot reset should be done in firmware
ibm,slot-pluggable	Boolean indicating whether the slot is pluggable
ibm,slot-surprise-pluggable	Boolean indicating whether the slot supports surprise hotplug
ibm,slot-broken-pdc	Boolean indicating whether PDC (Presence Detection Change) is broken
ibm,slot-power-ctl	Boolean indicating whether the slot has power control
ibm,slot-wired-lanes	The number of hardware lanes that are wired
ibm,slot-pwr-led-ctl	Presence of slot power led, and controlling entity
ibm,slot-attn-led-ctl	Presence of slot ATTN led, and controlling entity

3.10.1 PCI Hotplug

The implementation of PCI slot hotplug heavily relies on its power state. Initially, the slot is powered off if there are no adapters behind it. Otherwise, the slot should be powered on.

In hot add scenario, the adapter is physically inserted to PCI slot. Then the PCI slot is powered on by OPAL API opal_pci_set_power_state(). The power is supplied to the PCI slot, the adapter behind the PCI slot is probed and the device sub-tree (for hot added devices) is populated. A OPAL message is sent to OS on completion. The OS needs retrieve the device sub-tree through OPAL API opal_get_device_tree(), unflatten it and populate the device sub-tree. After that, the adapter behind the PCI slot should be probed and added to the system.

On the other hand, the OS removes the adapter behind the PCI slot before calling opal_pci_set_power_state(). Skiboot cuts off the power supply to the PCI slot, removes the adapter behind the PCI slot and the corresponding device sub-tree. A OPAL message (OPAL_MSG_ASYNC_COMP) is sent to OS. The OS removes the device sub-tree for the adapter behind the PCI slot.

The OPAL message used in PCI hotplug is comprised of 4 dwords in sequence: asychronous token from OS, PCI slot device node's phandle, OPAL_PCI_SLOT_POWER_{ON, OFF}, OPAL_SUCCESS or errcode.

The states OPAL_PCI_SLOT_OFFLINE and OPAL_PCI_SLOT_ONLINE are used for removing or adding devices behind the slot. The device nodes in the device tree are removed or added accordingly, without actually changing the slot's power state. The API call will return OPAL_SUCCESS immediately and no further asynchronous message will be sent.

3.10.2 PCI Slot on Apollo and Firenze

On IBM's Apollo and Firenze platform, the PCI VPD is fetched from dedicated LID, which is organized in so-called 1004, 1005, or 1006 format. 1006 mapping format isn't supported currently. The PCI slot properties are figured out from the VPD. On the other hand, there might have external power management entity hooked to I2C buses for one PCI slot. The fundamental reset operation of the PCI slot should be implemented based on the external power management entity for that case.

On Firenze platform, PERST pin is accessible through bit#10 of PCI config register (offset: 0x80) for those PCI slots behind some PLX switch downstream ports. For those PCI slots, PERST pin is utilized to implement fundamental reset if external power management entity doesn't exist.

For Apollo and Firenze platform, following PCI slot properties are exported through PCI device tree node except those generic properties (as above):

Property	Definition
ibm,slot-location-code	System location code string for the slot connector
ibm,slot-label	Slot label, part of "ibm,slot-location-code"

3.11 XSCOM Bindings

3.11.1 XSCOM regions

The top-level xscom nodes specify the mapping range from the 64-bit address space into the PCB address space.

There's one mapping range per chip xscom, therefore one node per mapping range.

```
/
/xscom@<chip-base-address-0>/
/xscom@<chip-base-address-1>/
...
/xscom@<chip-base-address-n>/
```

- where <chip-base-address-n> is the xscom base address with the gcid-specific bits (for chip n) OR-ed in.

Each xscom node has the following properties:

- #address-cells = 1
- #size-cells = 1
- reg = <base-address[#parent-address-cells] size[#parent-size-cells]>
- ibm,chip-id = gcid
- compatible = "ibm,xscom", "ibm,power8-scom" / "ibm,power7-xscom"

- ecid = <Electronic Chip ID, applicable for POWER9 onwards>
- wafer-id = <wafer ID, applicable for POWER9 onwards>
- wafer-location = <wafer location, applicable for POWER9 onwards>

3.11.2 ECID

Electronic Chip ID (ECID) is a process by which the wafer number, chip location (i.e. X,Y) and other optional data items are electrically encoded directly on the chip. wafer-id property represents wafer number and wafer-location property represents chip location (both X and Y location).

3.11.3 Chiplet endpoints

One sub-node per endpoint. Endpoints are defined by their (port, endpoint-address) data on the PCB, and are named according to their endpoint types:

```
/xscom@<chip-base-address>/
/xscom@<chip-base-address>/chiptod@<endpoint-addr>
/xscom@<chip-base-address>/lpc@<endpoint-addr>
```

- where the <endpoint-addr> is a single address (as distinct from the current (gcid,base) format), consisting of the SCOM port and SCOM endpoint bits in their 31-bit address format.

Each endpoint node has the following properties:

- reg = <endpoint-address[#parent-address-cells] size[#parent-size-cells]>
- compatible - depends on endpoint type, eg "ibm,power8-chiptod"

The endpoint address specifies the address on the PCB. So, to calculate the MMIO address for a PCB register:

```
mmio_addr  = <xscom-base-addr> | (pcb_addr[1:27] << 4)
                               | (pcb_addr[28:31] << 3)
```

Where:

- xscom-base-addr is the address from the first two cells of the parent node's reg property
- pcb_addr is the first cell of the endpoint's reg property

3.12 P9 XIVE Exploitation

3.12.1 I - Device-tree updates

1) The existing OPAL `/interrupt-controller@0` node remains

   This node represents both the emulated XICS source controller and an abstraction of the virtualization engine. This represents the fact thet OPAL set_xive/get_xive functions are still supported though they don't provide access to the full functionality.

   It is still the parent of all interrupts in the device-tree.

   New or modified properties:

   - `compatible` : This is extended with a new value `ibm,opal-xive-vc`

2) The new `/interrupt-controller@<addr>` node

   This node represents both the emulated XICS presentation controller and the new XIVE presentation layer.

   Unlike the traditional XICS, there is only one such node for the whole system.

   New or modified properties:

   - `compatible` : This contains at least the following strings:
     - `ibm,opal-intc` : This represents the emulated XICS presentation facility and might be the only property present if the version of OPAL doesn't support XIVE exploitation.
     - `ibm,opal-xive-pe` : This represents the XIVE presentation engine.
   - `ibm,xive-eq-sizes` : One cell per size supported, contains log2 of size, in ascending order.
   - `ibm,xive-#priorities` : One cell, the number of supported priorities (the priorities will be 0. . . n)
   - `ibm,xive-provision-page-size` : Page size (in bytes) of the pages to pass to OPAL for provisioning internal structures (see opal_xive_donate_page). If this is absent, OPAL will never require additional provisioning. The page must be naturally aligned.
   - `ibm,xive-provision-chips` : The list of chip IDs for which provisioning is required. Typically, if a VP allocation return OPAL_XIVE_PROVISIONING, opal_xive_donate_page() will need to be called to donate a page to *each* of these chips before trying again.
   - `reg` property contains the addresses & sizes for the register ranges corresponding respectively to the 4 rings:
     - Ultravisor level
     - Hypervisor level
     - Guest OS level
     - User level

     For any of these, a size of 0 means this level is not supported.
   - `single-escalation-support` (option). When present, indicatges that the "single escalation" feature is supported, thus enabling the use of the OPAL_XIVE_VP_SINGLE_ESCALATION flag.

3) Interrupt descriptors

   The interrupt descriptors (aka "interrupts" properties and parts of "interrupt-map" properties) remain 2 cells. The first cell is a global interrupt number which represents a unique interrupt source in the system and is an abstraction provided by OPAL.

   The default configuration for all sources in the IVT/EAS is to issue that number (it's internally a combination of the source chip and per-chip interrupt number but the details of that combination are not exposed and subject to change).

   The second cell remains as usual "0" for an edge interrupt and "1" for a level interrupts.

4) IPIs

   Each `cpu` node now contains an `interrupts` property which has one entry (2 cells per entry) for each thread on that core containing the interrupt number for the IPI targeted at that thread.

5) Interrupt targets

   Targetting of interrupts uses processor targets and priority numbers. The processor target encoding depends on which API is used:

   - The legacy opal_set/get_xive() APIs only support the old "mangled" (ie. shifted by 2) HW processor numbers.

- The new opal_xive_set/get_irq_config API (and other exploitation mode APIs) use a "token" VP number which is described in II-2. Unmodified HW processor numbers are valid VP numbers for those APIs.

3.12.2 II - General operations

Most configuration operations are abstracted via OPAL calls, there is no direct access or exposure of such things as real HW interrupt or VP numbers.

OPAL sets up all the physical interrupts and assigns them numbers, it also allocates enough virtual interrupts to provide an IPI per physical thread in the system.

All interrupts are pre-configured masked and must be set to an explicit target before first use. The default interrupt number is programmed in the EAS and will remain unchanged if the targetting/unmasking is done using the legacy set_xive() interface.

An interrupt "target" is a combination of a target processor number and a priority.

Processor numbers are in a single domain that represents both the physical processors and any virtual processor or group allocated using the interfaces defined in this specification. These numbers are an OPAL maintained abstraction and are only partially related to the real VP numbers:

In order to maintain the grouping ability, when VPs are allocated in blocks of naturally aligned powers of 2, the underlying HW numbers will respect this alignment.

> **Note:** The block group mode extension makes the numbering scheme a bit more tricky than simple powers of two however, see below.

1) Interrupt numbering and allocation

   As specified in the device-tree definition, interrupt numbers are abstracted by OPAL to be a 30-bit number. All HW interrupts are "allocated" and configured at boot time along with enough IPIs for all processor threads.

   Additionally, in order to be compatible with the XICS emulation, all interrupt numbers present in the device-tree (ie all physical sources or pre-allocated IPIs) will fit within a 24-bit number space.

   Interrupt sources that are only usable in exploitation mode, such as escalation interrupts, can have numbers covering the full 30-bit range. The same is true of interrupts allocated dynamically.

   The hypervisor can allocate additional blocks of interrupts, in which case OPAL will return the resulting abstracted global numbers. They will have to be individually configured to map to a given number at the target and be routed to a given target and priority using opal_xive_set_irq_config(). This call is semantically equivalent to the old opal_set_xive() which is still supported with the addition that opal_xive_set_irq_config() can also specify the logical interrupt number.

2) VP numbering and allocation

   A VP number is a 64-bit number. The internal make-up of that number is opaque to the OS. However, it is a discrete integer that will be a naturally aligned power of two when allocating a chunk of VPs representing the "base" number of that chunk, the OS will do basic arithmetic to get to all the VPs in the range.

   Groups, when supported, will also be numbers in that space.

   The physical processors numbering uses the same number space.

   The underlying HW VP numbering is hidden from the OS, the APIs uses the system processor numbers as presented in the `ibm,ppc-interrupt-server#s` which corresponds to the PIR register content to represent physical processors within the same number space as dynamically allocated VPs.

   > **Note:** Note about block group mode:

The block group mode shall as much as possible be handled transparently by OPAL.

For example, on a 2-chips machine, a request to allocate 2^n VPs might result in an allocation of 2^(n-1) VPs per chip allocated accross 2 chips. The resulting VP numbers will encode the order of the allocation allowing OPAL to reconstitute which bits are the block ID bits and which bits are the index bits in a way transparent to the OS. The overall range of numbers passed to Linux will still be contiguous.

That implies however a limitation: We can only allocate within power-of-two number of blocks. Thus the VP allocator will limit itself to the largest power of two that can fit in the number of available chips in the machine: A machine with 3 good chips will only be able to allocate VPs from 2 of them.

---

3) Group numbering and allocation

   The group numbers are in the *same* number space as the VP numbers. OPAL will internally use some bits of the VP number to encode the group geometry.

   [TBD] OPAL may or may not allocate a default group of all physical processors, per-chip groups or per-core groups. This will be represented in the device-tree somewhat...

   [TBD] OPAL will provide interfaces for allocating groups

Note: Note about P/Q bit operation on sources:

opal_xive_get_irq_info() returns a certain number of flags which define the type of operation supported. The following rules apply based on what those flags say:

- The Q bit isn't functional on an LSI interrupt. There is no garantee that the special combination "01" will work for an LSI (and in fact it will not work on the PHB LSIs). However just setting P to 1 is sufficient to mask an LSI (just don't EOI it while masked).
- The recommended setting for a masked interrupt that is temporarily masked by a driver is "10". This means a new occurrence while masked will be recorded and a "StoreEOI" will replay it appropriately.

3.12.3 III - Event queues

Each virtual processor or group has a certain number of event queues associated with it. Each correspond to a given priority. The number of supported priorities is provided in the device-tree (`ibm,xive-#priorities` property of the xive node).

By default, OPAL populates at least one queue for every physical thread in the system. The number of queues and the size used is implementation specific. If the OS wants to re-use these to save memory, it can query the VP configuration.

The opal_xive_get_queue_info() and opal_xive_set_queue_info() can be used to query a queue configuration (ie, to obtain the current page and size for the queue itself, but also to collect some configuration flags for that queue such as whether it coalesces notifications etc...) and to obtain the MMIO address of the queue EOI page (in the case where coalescing is enabled).

3.12.4 IV - OPAL APIs

Warning: *All* the calls listed below may return OPAL_BUSY unless explicitely documented not to. In that case, the call should be performed again. The OS is allowed to insert a delay though no minimum nor maxmimum delay is specified. This will typically happen when performing cache update operations in the XIVE, if they result in a collision.

Warning: Calls that are expected to be called at runtime simultaneously without conflicts such as getting/setting IRQ info or queue info are fine to do so concurrently.

However, there is no internal locking to prevent races between things such as freeing a VP block and getting/setting queue infos on that block.

These aren't fully specified (yet) but common sense shall apply.

OPAL_XIVE_RESET

```
int64_t opal_xive_reset(uint64_t version)
```

The OS should call this once when starting up to re-initialize the XIVE hardware and the OPAL XIVE related state back to all defaults.

It can call it a second time before handing over to another (ie. kexec) to re-enable XICS emulation.

The "version" argument should be set to 1 to enable the XIVE exploitation mode APIs or 0 to switch back to the default XICS emulation mode.

Future versions of OPAL might allow higher versions than 1 to represent newer versions of this API. OPAL will return an error if it doesn't recognize the requested version.

Any page of memory that the OS has "donated" to OPAL, either backing store for EQDs or VPDs or actual queue buffers will be removed from the various HW maps and can be re-used by the OS or freed after this call regardless of the version information. The HW will be reset to a (mostly) clean state.

It is the responsibility of the caller to ensure that no other XIVE or XICS emulation call happens simultaneously to this. This basically should happen on an otherwise quiescent system. In the case of kexec, it is recommended that all processors CPPR is lowered first.

Note: This call always executes fully synchronously, never returns OPAL_BUSY and will work regardless of whether VPs and EQs are left enabled or disabled. It *will* spend a significant amount of time inside OPAL and as such is not suitable to be performed during normal runtime.

OPAL_XIVE_GET_IRQ_INFO

```
int64_t opal_xive_get_irq_info(uint32_t girq,
                               uint64_t *out_flags,
                               uint64_t *out_eoi_page,
                               uint64_t *out_trig_page,
                               uint32_t *out_esb_shift,
                               uint32_t *out_src_chip);
```

Returns info about an interrupt source. This call never returns OPAL_BUSY.

- out_flags returns a set of flags. The following flags are defined in the API (some bits are reserved, so any bit not defined here should be ignored):
  - OPAL_XIVE_IRQ_TRIGGER_PAGE

    Indicate that the trigger page is a separate page. If that bit is clear, there is either no trigger page or the trigger can be done in the same page as the EOI, see below.
  - OPAL_XIVE_IRQ_STORE_EOI

Indicates that the interrupt supports the "Store EOI" option, ie a store to the EOI page will move Q into P and retrigger if the resulting P bit is 1. If this flag is 0, then a store to the EOI page will do a trigger if OPAL_XIVE_IRQ_TRIGGER_PAGE is also 0.

- OPAL_XIVE_IRQ_LSI

  Indicates that the source is a level sensitive source and thus doesn't have a functional Q bit. The Q bit may or may not be implemented in HW but SW shouldn't rely on it doing anything.

- OPAL_XIVE_IRQ_SHIFT_BUG

  Indicates that the source has a HW bug that shifts the bits of the "offset" inside the EOI page left by 4 bits. So when this is set, us 0xc000, 0xd000… instead of 0xc00, 0xd00… as offets in the EOI page.

- OPAL_XIVE_IRQ_MASK_VIA_FW

  Indicates that a FW call is needed (either opal_set_xive() or opal_xive_set_irq_config()) to succesfully mask and unmask the interrupt. The operations via the ESB page aren't fully functional.

- OPAL_XIVE_IRQ_EOI_VIA_FW

  Indicates that a FW call to opal_xive_eoi() is needed to successfully EOI the interrupt. The operation via the ESB page isn't fully functional.

  * out_eoi_page and out_trig_page outputs will be set to the EOI page physical address (always) and the trigger page address (if it exists). The trigger page may exist even if OPAL_XIVE_IRQ_TRIGGER_PAGE is not set. In that case out_trig_page is equal to out_eoi_page. If the trigger page doesn't exist, out_trig_page is set to 0.
  * out_esb_shift contains the size (as an order, ie 2^n) of the EOI and trigger pages. Current supported values are 12 (4k) and 16 (64k). Those cannot be configured by the OS and are set by firmware but can be different for different interrupt sources.
  * out_src_chip will be set to the chip ID of the HW entity this interrupt is sourced from. It's meant to be informative only and thus isn't guaranteed to be 100% accurate. The idea is for the OS to use that to pick up a default target processor on the same chip.

OPAL_XIVE_EOI

```
int64_t opal_xive_eoi(uint32_t girq);
```

Performs an EOI on the interrupt. This should only be called if OPAL_XIVE_IRQ_EOI_VIA_FW is set as otherwise direct ESB access is preferred.

Note: This is the *same* opal_xive_eoi() call used by OPAL XICS emulation. However the XIRR parameter is repurposed as "GIRQ".

The call will perform the appropriate function depending on whether OPAL is in XICS emulation mode or native XIVE exploitation mode.

OPAL_XIVE_GET_IRQ_CONFIG

```
int64_t opal_xive_get_irq_config(uint32_t girq, uint64_t *out_vp,
                                 uint8_t *out_prio, uint32_t *out_lirq);
```

Returns current the configuration of an interrupt source. This is the equivalent of opal_get_xive() with the addition of the logical interrupt number (the number that will be presented in the queue).

- girq: The interrupt number to get the configuration of as provided by the device-tree.
- out_vp: Will contain the target virtual processor where the interrupt is currently routed to. This can return 0xffffffff if the interrupt isn't routed to a valid virtual processor.
- out_prio: Will contain the priority of the interrupt or 0xff if masked
- out_lirq: Will contain the logical interrupt assigned to the interrupt. By default this will be the same as girq.

OPAL_XIVE_SET_IRQ_CONFIG

```
int64_t opal_xive_set_irq_config(uint32_t girq, uint64_t vp, uint8_t prio,
                                 uint32_t lirq);
```

This allows configuration and routing of a hardware interrupt. This is equivalent to opal_set_xive() with the addition of the ability to configure the logical IRQ number (the number that will be presented in the target queue).

- girq: The interrupt number to configure of as provided by the device-tree.
- vp: The target virtual processor. The target VP/Prio combination must already exist, be enabled and populated (ie, a queue page must be provisioned for that queue).
- prio: The priority of the interrupt.
- lirq: The logical interrupt number assigned to that interrupt

Note: Note about masking:

If the prio is set to 0xff, this call will cause the interrupt to be masked (*). This function will not clobber the source P/Q bits (**). It will however set the IVT/EAS "mask" bit if the prio passed is 0xff which means that interrupt events from the ESB will be discarded, potentially leaving the ESB in a stale state. Thus care must be taken by the caller to "cleanup" the ESB state appropriately before enabling an interrupt with this.

(*) Escalation interrupts cannot be masked via this function

(**) The exception to this rule is interrupt sources that have the OPAL_XIVE_IRQ_MASK_VIA_FW flag set. For such sources, the OS should make no assumption as to the state of the ESB and this function *will* perform all the necessary masking and unmasking.

Note: This call contains an implicit opal_xive_sync() of the interrupt source (see OPAL_XIVE_SYNC below)

It is recommended for an OS exploiting the XIVE directly to not use this function for temporary driver-initiated masking of interrupts but to directly mask using the P/Q bits of the source instead.

Masking using this function is intended for the case where the OS has no handler registered for a given interrupt anymore or when registering a new handler for an interrupt that had none. In these case, losing interrupts happening while no handler was attached is considered fine.

OPAL_XIVE_GET_QUEUE_INFO

```
int64_t opal_xive_get_queue_info(uint64_t vp, uint32_t prio,
                                 uint64_t *out_qpage,
                                 uint64_t *out_qsize,
                                 uint64_t *out_qeoi_page,
```

(continues on next page)

(continued from previous page)

```
                                  uint32_t *out_escalate_irq,
                                  uint64_t *out_qflags);
```

This returns informations about a given interrupt queue associated with a virtual processor and a priority.

- out_qpage: will contain the physical address of the page where the interrupt events will be posted or 0 if none has been configured yet.
- out_qsize: will contain the log2 of the size of the queue buffer or 0 if the queue hasn't been populated. Example: 12 for a 4k page.
- out_qeoi_page: will contain the physical address of the MMIO page used to perform EOIs for the queue notifications.
- out_escalate_irq: will contain a girq number for the escalation interrupt associated with that queue.

  Warning: The "escalate_irq" is a special interrupt number, depending on the implementation it may or may not correspond to a normal XIVE source. Those interrupts have no triggers, and will not be masked by opal_set_irq_config() with a prio of 0xff.

  ..note:: The state of the OPAL_XIVE_VP_SINGLE_ESCALATION flag passed to
  opal_xive_set_vp_info() can change the escalation irq number, so make sure you only retrieve this after having set the flag to the desired value. When set, all priorities will have the same escalation interrupt.

- out_qflags: will contain flags defined as follow:
  - OPAL_XIVE_EQ_ENABLED

    This must be set for the queue to be enabled and thus a valid target for interrupts. Newly allocated queues are disabled by default and must be disabled again before being freed (allocating and freeing of queues currently only happens along with their owner VP).

    Note: A newly enabled queue will have the generation set to 1 and the queue pointer to 0. If the OS wants to "reset" a queue generation and pointer, it thus must disable and re-enable the queue.

  - OPAL_XIVE_EQ_ALWAYS_NOTIFY

    When this is set, the HW will always notify the VP on any new entry in the queue, thus the queue own P/Q bits won't be relevant and using the EOI page will be unnecessary.

  - OPAL_XIVE_EQ_ESCALATE

    When this is set, the EQ will escalate to the escalation interrupt when failing to notify.

OPAL_XIVE_SET_QUEUE_INFO

```
int64_t opal_xive_set_queue_info(uint64_t vp, uint32_t prio,
                                  uint64_t qpage,
                                  uint64_t qsize,
                                  uint64_t qflags);
```

This allows the OS to configure the queue page for a given processor and priority and adjust the behaviour of the queue via flags.

- qpage: physical address of the page where the interrupt events will be posted. This has to be naturally aligned.

- qsize: log2 of the size of the above page. A 0 here will disable the queue.
- qflags: Flags (see definitions in opal_xive_get_queue_info)

Note: This call will reset the generation bit to 1 and the queue production pointer to 0.

Note: The PQ bits of the escalation interrupts and of the queue notification will be set to 00 when OPAL_XIVE_EQ_ENABLED is set, and to 01 (masked) when disabling it.

Note: This must be called at least once on a queue with the flag OPAL_XIVE_EQ_ENABLED in order to enable it after it has been allocated (along with its owner VP).

Note: When the queue is disabled (flag OPAL_XIVE_EQ_ENABLED cleared) all other flags and arguments are ignored and the queue configuration is wiped.

OPAL_XIVE_DONATE_PAGE

```
int64_t opal_xive_donate_page(uint32_t chip_id, uint64_t addr);
```

This call is used to donate pages to OPAL for use by VP/EQ provisioning.

The pages must be of the size specified by the “ibm,xive-provision-page-size” property and naturally aligned.

All donated pages are forgotten by OPAL (and thus returned to the OS) on any call to opal_xive_reset().

The chip_id should be the chip on which the pages were allocated or -1 if unspecified. Ideally, when a VP allocation request fails with the OPAL_XIVE_PROVISIONING error, the OS should allocate one such page for each chip in the system and hand it to OPAL before trying again.

Note: It is possible that the provisioning ends up requiring more than one page per chip. OPAL will keep returning the above error until enough pages have been provided.

OPAL_XIVE_ALLOCATE_VP_BLOCK

```
int64_t opal_xive_alloc_vp_block(uint32_t alloc_order);
```

This call is used to allocate a block of VPs. It will return a number representing the base of the block which will be aligned on the alloc order, allowing the OS to do basic arithmetic to index VPs in the block.

The VPs will have queue structures reserved (but not initialized nor provisioned) for all the priorities defined in the “ibm,xive-#priorities” property

This call might return OPAL_XIVE_PROVISIONING. In this case, the OS must allocate pages and provision OPAL using opal_xive_donate_page(), see the documentation for opal_xive_donate_page() for details.

The resulting VPs must be individually enabled with opal_xive_set_vp_info below with the OPAL_XIVE_VP_ENABLED flag set before use.

For all priorities, the corresponding queues must also be individually provisioned and enabled with opal_xive_set_queue_info.

OPAL_XIVE_FREE_VP_BLOCK

```
int64_t opal_xive_free_vp_block(uint64_t vp);
```

This call is used to free a block of VPs. It must be called with the same *base* number as was returned by opal_xive_alloc_vp() (any index into the block will result in an OPAL_PARAMETER error).

The VPs must have been previously all disabled with opal_xive_set_vp_info below with the OPAL_XIVE_VP_ENABLED flag cleared before use.

All the queues must also have been disabled.

Failure to do any of the above will result in an OPAL_XIVE_FREE_ACTIVE error.

OPAL_XIVE_GET_VP_INFO

```
int64_t opal_xive_get_vp_info(uint64_t vp,
                              uint64_t *flags,
                              uint64_t *cam_value,
                              uint64_t *report_cl_pair,
                              uint32_t *chip_id);
```

This call returns information about a VP:

- flags:
  - OPAL_XIVE_VP_ENABLED

    Returns the enabled state of the VP
  - OPAL_XIVE_VP_SINGLE_ESCALATION (if available)

    Returns whether single escalation mode is enabled for this VP (see opal_xive_set_vp_info()).
- cam_value: This is the value to program into the thread management area to dispatch that VP (ie, an encoding of the block + index).
- report_cl_pair: This is the real address of the reporting cache line pair for that VP (defaults to 0, ie disabled)
- chip_id: The chip that VCPU was allocated on

OPAL_XIVE_SET_VP_INFO

```
int64_t opal_xive_set_vp_info(uint64_t vp,
                              uint64_t flags,
                              uint64_t report_cl_pair);
```

This call configures a VP:

- flags:
  - OPAL_XIVE_VP_ENABLED

    This must be set for the VP to be usable and cleared before freeing it.

Note: This can be used to disable the boot time VPs though this isn't recommended. This must be used to enable allocated VPs.

  - OPAL_XIVE_VP_SINGLE_ESCALATION (if available)

    If this is set, the queues are configured such that all priorities turn into a single escalation interrupt. This results in the loss of priority 7 which can no longer be used. This this needs to be set before any interrupt is routed to that priority and queue 7 must not have been already enabled.

    This feature is available if the "single-escalation-property" is present in the xive device-tree node.

    Warning: When enabling single escalation, and pre-existing routing and configuration of the individual queues escalation is lost (except queue 7 which is the new merged escalation). When further disabling it, the previous value is not retrieved and the field cleared, escalation is disabled on all the queues.

- report_cl_pair: This is the real address of the reporting cache line pair for that VP or 0 to disable.

Note: When disabling a VP, all other VP settings are lost.

OPAL_XIVE_ALLOCATE_IRQ

```
int64_t opal_xive_allocate_irq(uint32_t chip_id);
```

This call allocates a software IRQ on a given chip. It returns the interrupt number or a negative error code.

OPAL_XIVE_FREE_IRQ

```
int64_t opal_xive_free_irq(uint32_t girq);
```

This call frees a software IRQ that was allocated by opal_xive_allocate_irq. Passing any other interrupt number will result in an OPAL_PARAMETER error.

OPAL_XIVE_SYNC

```
int64_t opal_xive_sync(uint32_t type, uint32_t id);
```

This call is uses to synchronize some HW queues to ensure various changes have taken effect to the point where their effects are visible to the processor.

- type: Type of synchronization:
  - XIVE_SYNC_EAS: Synchronize a source. "id" is the girq number of the interrupt. This will ensure that any change to the PQ bits or the interrupt targetting has taken effect.
  - XIVE_SYNC_QUEUE: Synchronize a target queue. "id" is the girq number of the interrupt. This will ensure that any previous occurrence of the interrupt has reached the in-memory queue and is visible to the processor.

Note: XIVE_SYNC_EAS and XIVE_SYNC_QUEUE can be used together (ie. XIVE_SYNC_EAS | XIVE_SYNC_QUEUE) to completely synchronize the path of an interrupt to its queue.

- id: Depends on the synchronization type, see above

OPAL_XIVE_DUMP

```
int64_t opal_xive_dump(uint32_t type, uint32_t id);
```

This is a debugging call that will dump in the OPAL console various state information about the XIVE.

- type: Type of info to dump:
  - **XIVE_DUMP_TM_HYP: Dump the TIMA area for hypervisor physical thread** "id" is the PIR value of the thread
  - **XIVE_DUMP_TM_POOL: Dump the TIMA area for the hypervisor pool** "id" is the PIR value of the thread
  - **XIVE_DUMP_TM_OS: Dump the TIMA area for the OS** "id" is the PIR value of the thread
  - **XIVE_DUMP_TM_USER: Dump the TIMA area for the "user" area (unsupported)** "id" is the PIR value of the thread
  - **XIVE_DUMP_VP: Dump the state of a VP structure** "id" is the VP id
  - **XIVE_DUMP_EMU: Dump the state of the XICS emulation for a thread** "id" is the PIR value of the thread

OPAL_XIVE_GET_QUEUE_STATE

```
int64_t opal_xive_get_queue_state(uint64_t vp, uint32_t prio,
                                  uint32_t *out_qtoggle,
                                  uint32_t *out_qindex);
```

This call saves the queue toggle bit and index. This must be called on an enabled queue.

- vp, prio: The target queue
- out_qtoggle: toggle bit of the queue
- out_qindex: index of the queue

OPAL_XIVE_SET_QUEUE_STATE

```
int64_t opal_xive_set_queue_state(uint64_t vp, uint32_t prio,
                                  uint32_t qtoggle,
                                  uint32_t qindex);
```

This call restores the queue toggle bit and index that was previously saved by a call to opal_xive_get_queue_state(). This must be called on an enabled queue.

- vp, prio: The target queue
- qtoggle: toggle bit of the queue
- qindex: index of the queue

OPAL_XIVE_GET_VP_STATE

```
int64_t opal_xive_get_vp_state(uint64_t vp_id,
                               uint64_t *out_state);
```

This call saves the VP HW state in "out_state". The format matches the XIVE NVT word 4 and word 5. This must be called on an enabled VP.

- vp_id: The target VP
- out_state: Location where the state is to be stored

3.13 OPAL/Skiboot In-Memory Collection (IMC) interface Documentation

3.13.1 Overview:

In-Memory-Collection (IMC) is performance monitoring infrastrcuture for counters that (once started) can be read from memory at any time by an operating system. Such counters include those for the Nest and Core units, enabling continuous monitoring of resource utilisation on the chip.

The API is agnostic as to how these counters are implemented. For the Nest units, they're implemented by having microcode in an on-chip microcontroller and for core units, they are implemented as part of core logic to gather data and periodically write it to the memory locations.

3.13.2 Nest (On-Chip, Off-Core) unit:

Nest units have dedicated hardware counters which can be programmed to monitor various chip resources such as memory bandwidth, xlink bandwidth, alink bandwidth, PCI, NVlink and so on. These Nest unit PMU counters can be programmed in-band via scom. But alternatively, programming of these counters and periodically moving the counter data to memory are offloaded to a hardware engine part of OCC (On-Chip Controller).

Microcode, starts to run at system boot in OCC complex, initialize these Nest unit PMUs and periodically accumulate the nest pmu counter values to memory. List of supported events by the microcode is packages as a DTS and stored in IMA_CATALOG partition.

3.13.3 Core unit:

Core IMC PMU counters are handled in the core-imc unit. Each core has 4 Core Performance Monitoring Counters (CPMCs) which are used by Core-IMC logic. Two of these are dedicated to count core cycles and instructions. The 2 remaining CPMCs have to multiplex 128 events each.

Core IMC hardware does not support interrupts and it peridocially (based on sampling duration) fetches the counter data and accumulate to main memory. Memory to accumulate counter data are refered from "PDBAR" (per-core scom) and "LDBAR" per-thread spr.

3.13.4 Trace mode of IMC:

POWER9 support two modes for IMC which are the Accumulation mode and Trace mode. In Accumulation mode event counts are accumulated in system memory. Hypervisor/kernel then reads the posted counts periodically, or when requested. In IMC Trace mode, the 64 bit trace scom value is initialized with the event information. The CPMC*SEL and CPMC_LOAD in the trace scom, specifies the event to be monitored and the sampling duration. On each overflow in the CPMC*SEL, hardware snapshots the program counter along with event counts and writes into memory pointed by LDBAR. LDBAR has bits to indicate whether hardware is configured for accumulation or trace mode. Currently the event monitored for trace-mode is fixed as cycle.

PMI interrupt handling is avoided, since IMC trace mode snapshots the program counter and update to the memory. And this also provide a way for the operating system to do instruction sampling in real time without PMI(Performance Monitoring Interrupts) processing overhead.

Example:

Performance data using 'perf top' with and without trace-imc event:

PMI interrupts count when 'perf top' command is executed without trace-imc event.

```
# cat /proc/interrupts  (a snippet from the output)
9944      1072        804         804        1644        804        1306
804        804        804         804         804        804         804
804        804       1961        1602         804        804        1258
[-----------------------------------------------------------------]
803        803        803         803         803        803         803
803        803        803         803         804        804         804
804        804        804         804         804        804         803
803        803        803         803         803       1306         803
803   Performance monitoring interrupts
```

PMI interrupts count when 'perf top' command executed with trace-imc event (executed right after 'perf top' without trace-imc event).

```
# perf top -e trace_imc/trace_cycles/
12.50%  [kernel]          [k] arch_cpu_idle
11.81%  [kernel]          [k] __next_timer_interrupt
11.22%  [kernel]          [k] rcu_idle_enter
10.25%  [kernel]          [k] find_next_bit
 7.91%  [kernel]          [k] do_idle
 7.69%  [kernel]          [k] rcu_dynticks_eqs_exit
 5.20%  [kernel]          [k] tick_nohz_idle_stop_tick
     [-----------------------]

# cat /proc/interrupts (a snippet from the output)

9944      1072        804         804        1644        804        1306
804        804        804         804         804        804         804
804        804       1961        1602         804        804        1258
[-----------------------------------------------------------------]
803        803        803         803         803        803         803
803        803        803         804         804        804         804
804        804        804         804         804        804         803
803        803        803         803         803       1306         803
803   Performance monitoring interrupts
```

Here the PMI interrupts count remains the same.

3.13.5 OPAL APIs:

The OPAL API is simple: a call to init a counter type, and calls to start and stop collection. The memory locations are described in the device tree.

See *OPAL_IMC_COUNTERS_INIT* and *IMC Device Tree Bindings*

3.14 Power Management

See *ibm,opal/power-mgt device tree entries* for device tree structure describing power management facilities.

3.14.1 Debugging

There exist a few debug knobs that can be set via nvram settings. These are **not** ABI and may be changed or removed at *any* time.

Disabling specific stop states

On boot, specific stop states can be disabled via setting a mask. For example, to disable all but stop 0,1,2, use ~0xE0000000.

```
nvram -p ibm,skiboot --update-config opal-stop-state-disable-mask=0x1FFFFFFF
```

CHAPTER

FOUR

OPAL ABI

4.1 Secure and Trusted Boot Library (LibSTB) Documentation

LibSTB provides APIs to support Secure Boot and Trusted Boot in skiboot.

`Secure Boot: verify and enforce.` When the system is booting in secure mode, Secure Boot MUST ensure that only trusted code is executed during system boot by verifying if the code is signed with trusted keys and halting the system boot if the verification fails.

`Trusted Boot: measure and record.` When the system is booting in trusted mode, Trusted Boot MUST create artifacts during system boot to prove that a particular chain of events have happened during boot. Interested parties can subsequently assess the artifacts to check whether or not only trusted events happened and then make security decisions. These artifacts comprise a log of measurements and the digests extended into the TPM PCRs. Platform Configuration Registers (PCRs) are registers in the Trusted Platform Module (TPM) that are shielded from direct access by the CPU.

In order to support Secure and Trusted Boot, the flash driver calls libSTB to verify and measure the code it fetches from PNOR.

LibSTB is initialized by calling *secureboot_init()*, see `libstb/secureboot.h`.

4.1.1 Secure Boot

`Requirements:`

1. CVC-verify service to verify signed firmware code.

Secure boot is initialized by calling *secureboot_init()* and its API is quite simple, see `libstb/secureboot.h`.

The flash driver calls `secureboot_verify()` to verify if the fetched firmware blob is properly signed with keys trusted by the platform owner. This verification is performed only when the system is booting in secure mode. If the verification fails, it enforces a halt of the system boot.

The verification itself is performed by the *Container Verification Code*, precisely the *CVC-verify* service, which requires both the fetched code and the hardware key hash trusted by the platform owner.

The secure mode status, hardware key hash and hardware key hash size information is found in the device tree, see *doc/device-tree/ibm,secureboot.rst*.

Signing Firmware Code

Fimware code is signed using the `sb-signing-utils` utilities by running it standalone or just calling op-build. The latter will automatically sign the various firmware components that comprise the PNOR image if SECUREBOOT is enabled for the platform.

The signing utilities also allow signing firmware code using published hardware keys (a.k.a. imprint keys, only for development) or production hardware keys, see sb-signing-utils.

The hardware keys are the root keys. The signing tool uses three hardware keys to sign up to three firmware keys, which are then used to sign the firmware code. The resulting signed firmware code is then assembled following the secure boot container format. All the information required to verify the signatures is placed in the first 4K reserved for the container header (e.g. public keys, hashes and signatures). The firmware code itself is placed in the container payload.

4.1.2 Container Verification Code

The *Container Verification Code* (a.k.a. ROM code) is stored in a secure memory region and it provides basic Secure and Trusted Boot services for the entire firmware stack. See *doc/device-tree/ibm,secureboot.rst <device-tree/ibm,secureboot>* and *doc/device-tree/ibm,cvc.rst <device-tree/ibm,cvc>*.

LibSTB uses function wrappers to call into each CVC service, see `libstb/cvc.h`.

CVC-verify Service

```
int call_cvc_verify(void *buf, size_t size, const void *hw_key_hash,
                    size_t hw_key_hash_size, uint64_t *log)
```

This function wrapper calls into the *CVC-verify*, which verifies if the firmware code provided in `@buf` is properly signed with the keys trusted by the platform owner. Its parameters are documented in `libstb/cvc.h`.

`@hw_key_hash` is used to check if the firware keys used to sign the firmware blob can be trusted.

`@log` is optional. If the verification fails, the caller can interpret it to find out what checks has failed.

Enforcement is caller's responsibility.

CVC-sha512 Service

```
int call_cvc_sha512(const uint8_t *data, size_t data_len, uint8_t *digest,
                    size_t digest_size)
```

This function wrapper calls into the *CVC-sha512*, which calculates the sha512 hash of what is provided in @data. Its parameters are documented in `libstb/cvc.h`.

4.1.3 Trusted Boot

Requirements:

1. TPM device and TPM driver. See devices supported in *doc/device-tree/tpm.rst*.
2. TCG Software Stack (TSS) to send commands to the TPM device.
3. Firmware Event Log driver to add new events to the log. Event log address and size information is found in the device tree, see *doc/device-tree/tpm.rst*.
4. CVC-sha512 service to calculate the sha512 hash of the data that will be measured.

The Trusted Boot API is quite simple, see `libstb/trustedboot.h`.

The flash driver calls `trustedboot_measure()` to measure the firmware code fetched from PNOR and also record its measurement in two places. This is performed only when the system is booting in trusted mode (information found in the device tree, see *doc/device-tree/ibm,secureboot.rst*).

Once the firmware code is measured by calling the *CVC-sha512* service, its measurement is first recorded in a TPM PCR statically defined for each event. In order to record it, the skiboot TCG Software Stack (TSS) API is called to extend the measurement into the PCR number of both the sha1 and sha256 banks. The skiboot TSS is a light TSS implementation and its source code is shared between hostboot and skiboot, see `libstb/tss/trustedbootCmds.H`.

PCR extend is an TPM operation that uses a hash function to combine a new measurement with the existing digest saved in the PCR. Basically, it concatenates the existing PCR value with the received measurement, and then records the hash of this new string in the PCR.

The measurement is also recorded in the event log. The `TpmLogMgr_addEvent()` function is called to add the measurement to the log, see `libstb/tss/tpmLogMgr.H`.

When the system boot is complete, each non-zero PCR value represents one or more events measured during the boot in chronological order. Interested parties can make inferences about the system's state by using an attestation tool to remotely compare the PCR values of a TPM against known good values, and also identify unexpected events by replaying the Event Log against known good Event Log entries.

4.2 Device Tree

General notes on the Device Tree produced by skiboot. This chapter **needs updating**.

4.2.1 General comments

- skiboot does not require nodes to have phandle properties, but if you have them then *all* nodes must have them including the root of the device-tree (currently a HB bug !). It is recommended to have them since they are needed to represent the cache levels.
- **NOTE**: The example tree below only has phandle properties for nodes that are referenced by other nodes. This is *not* correct and is purely done for keeping this document smaller, make sure to follow the rule above.
- Only the "phandle" property is required. Sapphire also generates a "linux,phandle" for backward compatibility but doesn't require it as an input
- Any property not specifically documented must be put in "as is"
- All ibm,chip-id properties contain a HW chip ID which correspond on P8 to the PIR value shifted right by 7 bits, ie. it's a 6-bit value made of a 3-bit node number and a 3-bit chip number.
- Unit addresses (@xxxx part of node names) should if possible use lower case hexadecimal to be consistent with what skiboot does and to help some stupid parsers out there…

4.2.2 Reserve Map

Here are the reserve map entries. They should exactly match the reserved-ranges property of the root node (see documentation of that property)

```
/dts-v1/;
/memreserve/  0x00000007fe600000 0x0000000000100000;
/memreserve/  0x00000007fe200000 0x0000000000100000;
/memreserve/  0x0000000031e00000 0x00000000003e0000;
/memreserve/  0x0000000031000000 0x0000000000e00000;
```

(continues on next page)

(continued from previous page)

```
/memreserve/  0x0000000030400000 0x0000000000c00000;
/memreserve/  0x0000000030000000 0x0000000000400000;
/memreserve/  0x0000000400000000 0x0000000000600450;
```

4.2.3 Root Node

Root node of device tree. There are a few required and a few optional properties that sit in the root node. They're described here.

compatible

The "compatible" properties are string lists indicating the overall compatibility from the more specific to the least specific.

The root node compatible property *must* contain "ibm,powernv" for Linux to have the powernv platform match the machine.

Each distinct platform *MUST* also add a more precise property (first in order) indicating the board type.

The standard naming is "vendor,name". For example: *compatible = "goog,rhesus","ibm,powernv";* would work. Or even better: *compatible = "goog,rhesus-v1","goog,rhesus","ibm,powernv";*.

The bare *ibm,powernv* should be reserved for bringup/testing:

```
/dts-v1/;
/ {
        compatible = "ibm,powernv";
  };
```

Example

```
/dts-v1/;
/ {
        compatible = "ibm,powernv";

        /* mandatory */
        #address-cells = <0x2>;
        #size-cells = <0x2>;

        /* User visible board name (will be shown in /proc/cpuinfo) */
        model = "Machine Name";

        /*
         * The reserved-names and reserve-names properties work hand in hand. The␣
↪first one
         * is a list of strings providing a "name" for each entry in the second one␣
↪using
         * the traditional "vendor,name" format.
         *
         * The reserved-ranges property contains a list of ranges, each in the form of␣
↪2 cells
         * of address and 2 cells of size (64-bit x2 so each entry is 4 cells)␣
↪indicating
```

(continues on next page)

(continued from previous page)

```
         * regions of memory that are reserved and must not be overwritten by skiboot␣
↪or
         * subsequently by the Linux Kernel.
         *
         * Corresponding entries must also be created in the "reserved map" part of␣
↪the flat
         * device-tree (which is a binary list in the header of the fdt).
         *
         * Unless a component (skiboot or Linux) specifically knows about a region␣
↪(usually
         * based on its name) and decides to change or remove it, all these regions are
         * passed as-is to Linux and to subsequent kernels across kexec and are kept
         * preserved.
         *
         * NOTE: Do *NOT* copy the entries below, they are just an example and are␣
↪actually
         * created by skiboot itself. They represent the SLW image as "detected" by␣
↪reading
         * the PBA BARs and skiboot own memory allocations.
         *
         * I would recommend that you put in there the SLW and OCC (or HOMER as one␣
↪block
         * if that's how you use it) and any additional memory you want to preserve␣
↪such
         * as FW log buffers etc...
         */

        reserved-names = "ibm,slw-image", "ibm,slw-image", "ibm,firmware-stacks", "ibm,
↪firmware-data", "ibm,firmware-heap", "ibm,firmware-code", "memory@400000000";
        reserved-ranges = <0x7 0xfe600000 0x0 0x100000 0x7 0xfe200000 0x0 0x100000 0x0␣
↪0x31e00000 0x0 0x3e0000 0x0 0x31000000 0x0 0xe00000 0x0 0x30400000 0x0 0xc00000 0x0␣
↪0x30000000 0x0 0x400000 0x4 0x0 0x0 0x600450>;

        /* Mandatory */
        cpus {
                #address-cells = <0x1>;
                #size-cells = <0x0>;

                /*
                 * The following node must exist for each *core* in the system. The␣
↪unit
                 * address (number after the @) is the hexadecimal HW CPU number (PIR␣
↪value)
                 * of thread 0 of that core.
                 */
                PowerPC,POWER8@20 {
                        /* mandatory/standard properties */
                        device_type = "cpu";
                        64-bit;
                        32-64-bridge;
                        graphics;
                        general-purpose;

                        /*
                         * The "status" property indicate whether the core is␣
↪functional. It's
                         * a string containing "okay" for a good core or "bad" for a␣
↪non-functional
```

(continues on next page)

(continued from previous page)

```
                         * one. You can also just ommit the non-functional ones from␣
↪the DT
                         */
                        status = "okay";

                        /*
                         * This is the same value as the PIR of thread 0 of that core
                         * (ie same as the @xx part of the node name)
                         */
                        reg = <0x20>;

                        /* same as above */
                        ibm,pir = <0x20>;

                        /* chip ID of this core */
                        ibm,chip-id = <0x0>;

                        /*
                         * interrupt server numbers (aka HW processor numbers) of all␣
↪threads
                         * on that core. This should have 8 numbers and the first one␣
↪should
                         * have the same value as the above ibm,pir and reg properties
                         */
                        ibm,ppc-interrupt-server#s = <0x20 0x21 0x22 0x23 0x24 0x25␣
↪0x26 0x27>;

                        /*
                         * This is the "architected processor version" as defined in␣
↪PAPR. Just
                         * stick to 0x0f000004 for P8 and things will be fine
                         */
                        cpu-version = <0x0f000004>;

                        /*
                         * These are various definitions of the page sizes and segment␣
↪sizes
                         * supported by the MMU, those values are fine for P8 for now
                         */
                        ibm,processor-segment-sizes = <0x1c 0x28 0xffffffff 0xffffffff>
↪;
                        ibm,processor-page-sizes = <0xc 0x10 0x18 0x22>;
                        ibm,segment-page-sizes = <0xc 0x0 0x3 0xc 0x0 0x10 0x7 0x18␣
↪0x38 0x10 0x110 0x2 0x10 0x1 0x18 0x8 0x18 0x100 0x1 0x18 0x0 0x22 0x120 0x1 0x22␣
↪0x3>;

                        /*
                         * Similarly that might need to be reviewed later but will do␣
↪for now...
                         */
                        ibm,pa-features = [0x6 0x0 0xf6 0x3f 0xc7 0x0 0x80 0xc0];

                        /* SLB size, use as-is */
                        ibm,slb-size = <0x20>;

                        /* VSX support, use as-is */
                        ibm,vmx = <0x2>;
```

(continues on next page)

(continued from previous page)

```
                        /* DFP support, use as-is */
                        ibm,dfp = <0x2>;

                        /* PURR/SPURR support, use as-is */
                        ibm,purr = <0x1>;
                        ibm,spurr = <0x1>;

                        /*
                         * Old-style core clock frequency. Only create this property␣
↪if the frequency fits
                         * in a 32-bit number. Do not create it if it doesn't
                         */
                        clock-frequency = <0xf5552d00>;

                        /*
                         * mandatory: 64-bit version of the core clock frequency,␣
↪always create this
                         * property.
                         */
                        ibm,extended-clock-frequency = <0x0 0xf5552d00>;

                        /* Timebase freq has a fixed value, always use that */
                        timebase-frequency = <0x1e848000>;

                        /* Same */
                        ibm,extended-timebase-frequency = <0x0 0x1e848000>;

                        /* Use as-is, values might need to be adjusted but that will␣
↪do for now */
                        reservation-granule-size = <0x80>;
                        d-tlb-size = <0x800>;
                        i-tlb-size = <0x0>;
                        tlb-size = <0x800>;
                        d-tlb-sets = <0x4>;
                        i-tlb-sets = <0x0>;
                        tlb-sets = <0x4>;
                        d-cache-block-size = <0x80>;
                        i-cache-block-size = <0x80>;
                        d-cache-size = <0x10000>;
                        i-cache-size = <0x8000>;
                        i-cache-sets = <0x4>;
                        d-cache-sets = <0x8>;
                        performance-monitor = <0x0 0x1>;

                        /*
                         * optional: phandle of the node representing the L2 cache for␣
↪this core,
                         * note: it can also be named "next-level-cache", Linux will␣
↪support both
                         * and Sapphire doesn't currently use those properties, just␣
↪passes them
                         * along to Linux
                         */
                        l2-cache = < 0x4 >;
                };
```

(continues on next page)

(continued from previous page)

```
		/*
		 * Cache nodes. Those are siblings of the processor nodes under /cpus␣
↪and
		 * represent the various level of caches.
		 *
		 * The unit address (and reg property) is mostly free-for-all as long␣
↪as
		 * there is no collisions. On HDAT machines we use the following␣
↪encoding
		 * which I encourage you to also follow to limit surprises:
		 *
		 * L2   :  (0x20 << 24) | PIR (PIR is PIR value of thread 0 of core)
		 * L3   :  (0x30 << 24) | PIR
		 * L3.5 :  (0x35 << 24) | PIR
		 *
		 * In addition, each cache points to the next level cache via its
		 * own "l2-cache" (or "next-level-cache") property, so the core node
		 * points to the L2, the L2 points to the L3 etc...
		 */

		l2-cache@20000020 {
			phandle = <0x4>;
			device_type = "cache";
			reg = <0x20000020>;
			status = "okay";
			cache-unified;
			d-cache-sets = <0x8>;
			i-cache-sets = <0x8>;
			d-cache-size = <0x80000>;
			i-cache-size = <0x80000>;
			l2-cache = <0x5>;
		};

		l3-cache@30000020 {
			phandle = <0x5>;
			device_type = "cache";
			reg = <0x30000020>;
			status = "bad";
			cache-unified;
			d-cache-sets = <0x8>;
			i-cache-sets = <0x8>;
			d-cache-size = <0x800000>;
			i-cache-size = <0x800000>;
		};

	};

	/*
	 * Interrupt presentation controller (ICP) nodes
	 *
	 * There is some flexibility as to how many of these are presents since
	 * a given node can represent multiple ICPs. When generating from HDAT we
	 * chose to create one per core
	 */
	interrupt-controller@3ffff80020000 {
		/* Mandatory */
		compatible = "IBM,ppc-xicp", "IBM,power8-icp";
```

(continues on next page)

(continued from previous page)

```
                interrupt-controller;
                #address-cells = <0x0>;
                device_type = "PowerPC-External-Interrupt-Presentation";

                /*
                 * Range of HW CPU IDs represented by that node. In this example
                 * the core starting at PIR 0x20 and 8 threads, which corresponds
                 * to the CPU node of the example above. The property in theory
                 * supports multiple ranges but Linux doesn't.
                 */
                ibm,interrupt-server-ranges = <0x20 0x8>;

                /*
                 * For each server in the above range, the physical address of the
                 * ICP register block and its size. Since the root node #address-cells
                 * and #size-cells properties are both "2", each entry is thus
                 * 2 cells address and 2 cells size (64-bit each).
                 */
                reg = <0x3ffff 0x80020000 0x0 0x1000 0x3ffff 0x80021000 0x0 0x1000␣
↪0x3ffff 0x80022000 0x0 0x1000 0x3ffff 0x80023000 0x0 0x1000 0x3ffff 0x80024000 0x0␣
↪0x1000 0x3ffff 0x80025000 0x0 0x1000 0x3ffff 0x80026000 0x0 0x1000 0x3ffff␣
↪0x80027000 0x0 0x1000>;
        };

        /*
         * The "memory" nodes represent physical memory in the system. They
         * do not represent DIMMs, memory controllers or Centaurs, thus will
         * be expressed separately.
         *
         * In order to be able to handle affinity properly, we require that
         * a memory node is created for each range of memory that has a different
         * "affinity", which in practice means for each chip since we don't
         * support memory interleaved across multiple chips on P8.
         *
         * Additionally, it is *not* required that one chip = one memory node,
         * it is perfectly acceptable to break down the memory of one chip into
         * multiple memory nodes (typically skiboot does that if the two MCs
         * are not interlaved).
         */
        memory@0 {
                device_type = "memory";

                /*
                 * We support multiple entries in the ibm,chip-id property for
                 * memory nodes in case the memory is interleaved across multiple
                 * chips but that shouldn't happen on P8
                 */
                ibm,chip-id = <0x0>;

                /* The "reg" property is 4 cells, as usual for a child of
                 * the root node, 2 cells of address and 2 cells of size
                 */
                reg = <0x0 0x0 0x4 0x0>;
        };

        /*
         * The XSCOM node. This is the closest thing to a "chip" node we have.
```

(continues on next page)

(continued from previous page)

```
         * there must be one per chip in the system (thus a DCM has two) and
         * while it represents the "parent" of various devices on the PIB/PCB
         * that we want to expose, it is also used to store all sort of
         * miscellaneous per-chip information on HDAT based systems (such
         * as VPDs).
         */
        xscom@3fc0000000000 {
                /* standard & mandatory */
                #address-cells = <0x1>;
                #size-cells = <0x1>;
                scom-controller;
                compatible = "ibm,xscom", "ibm,power8-xscom";

                /* The chip ID as usual ... */
                ibm,chip-id = <0x0>;

                /* The base address of xscom for that chip */
                reg = <0x3fc00 0x0 0x8 0x0>;

                /*
                 * This comes from HDAT and I *think* is the raw content of the
                 * module VPD eeprom (and thus doesn't have a standard ASCII keyword
                 * VPD format). We don't currently use it though ...
                 */
                ibm,module-vpd = < /* ... big pile of binary data ... */ >;

                /* PSI host bridge XSCOM register set */
                psihb@2010900 {
                        reg = <0x2010900 0x20>;
                        compatible = "ibm,power8-psihb-x", "ibm,psihb-x";
                };

                /* Chip TOD XSCOM register set */
                chiptod@40000 {
                        reg = <0x40000 0x34>;
                        compatible = "ibm,power-chiptod", "ibm,power8-chiptod";

                        /*
                         * Create that property with no value if this chip has
                         * the Primary TOD in the topology. If it has the secondary
                         * one (backup master ?) use "secondary".
                         */
                        primary;
                };

                /* NX XSCOM register set */
                nx@2010000 {
                        reg = <0x2010000 0x4000>;
                        compatible = "ibm,power-nx", "ibm,power8-nx";
                };

                /*
                 * PCI "PE Master" XSCOM register set for each active PHB
                 *
                 * For now, do *not* create these if the PHB isn't connected,
                 * clocked, or the PHY/HSS not configured.
                 */
```

(continues on next page)

(continued from previous page)

```
                pbcq@2012000 {
                        reg = <0x2012000 0x20 0x9012000 0x5 0x9013c00 0x15>;
                        compatible = "ibm,power8-pbcq";

                        /* Indicate the PHB index on the chip, ie, 0,1 or 2 */
                        ibm,phb-index = <0x0>;

                        /* Create that property to use the IBM-style "A/B" dual input
                         * slot presence detect mechanism.
                         */
                        ibm,use-ab-detect;

                        /*
                         * TBD: Lane equalization values. Not currently used by
                         * skiboot but will have to be sorted out
                         */
                        ibm,lane_eq = <0x0>;
                };

                pbcq@2012400 {
                        reg = <0x2012400 0x20 0x9012400 0x5 0x9013c40 0x15>;
                        compatible = "ibm,power8-pbcq";
                        ibm,phb-index = <0x1>;
                        ibm,use-ab-detect;
                        ibm,lane_eq = <0x0>;
                };

                /*
                 * Here's the LPC bus. Ideally each chip has one but in
                 * practice it's ok to only populate the ones actually
                 * used for something. This is not an exact representation
                 * of HW, in that case we would have eccb -> opb -> lpc,
                 * but instead we just have an lpc node and the address is
                 * the base of the ECCB register set for it
                 *
                 * Devices on the LPC are represented as children nodes,
                 * see example below for a standard UART.
                 */
                lpc@b0020 {
                        /*
                         * Empty property indicating this is the primary
                         * LPC bus. It will be used for the default UART
                         * if any and this is the bus that will be used
                         * by Linux as the virtual 64k of IO ports
                         */
                        primary;

                        /*
                         * 2 cells of address, the first one indicates the
                         * address type, see below
                         */
                        #address-cells = <0x2>;
                        #size-cells = <0x1>;
                        reg = <0xb0020 0x4>;
                        compatible = "ibm,power8-lpc";

                        /*
```

(continues on next page)

(continued from previous page)

```
                                 * Example device: a UART on IO ports.
                                 *
                                 * LPC address have 2 cells. The first cell is the
                                 * address type as follow:
                                 *
                                 *   0 : LPC memory space
                                 *   1 : LPC IO space
                                 *   2:  LPC FW space
                                 *
                                 * (This corresponds to the OPAL_LPC_* arguments
                                 * passed to the opal_lpc_read/write functions)
                                 *
                                 * The unit address follows the old ISA convention
                                 * for open firmware which prefixes IO ports with "i".
                                 *
                                 * (This is not critical and can be 1,3f8 if that's
                                 * problematic to generate)
                                 */
                                serial@i3f8 {
                                        reg = <0x1 0x3f8 8>;
                                        compatible = "ns16550", "pnpPNP,501";

                                        /* Baud rate generator base frequency */
                                        clock-frequency = < 1843200 >;

                                        /* Default speed to use */
                                        current-speed = < 115200 >;

                                        /* Historical, helps Linux */
                                        device_type = "serial";

                                        /*
                                         * Indicate which chip ID the interrupt
                                         * is routed to (we assume it will always
                                         * be the "host error interrupt" (aka
                                         * "TPM interrupt" of that chip).
                                         */
                                         ibm,irq-chip-id = <0x0>;
                                };
                        };
                };
};
```

4.3 Device Tree

Device Tree for OPAL. Please refer to Device Tree Spec.

4.3.1 ibm,cvc

This describes the code (a.k.a container verification code) that skiboot uses to verify signed firmware blobs. Each ibm,cvc child node describes CVC service, which has a version and offset (reg).

Added in the device tree from `ibm,secureboot-v2`.

Required properties

```
compatible:       should be "ibm,container-verification-code"

memory-region:    this points to the reserved memory where the
                  container-verification-code is stored.
```

Example

```
ibm,cvc {
        phandle = <0x10f>;
        #address-cells = <0x1>;
        #size-cells = <0x0>;
        compatible = "ibm,container-verification-code";
        memory-region = <0xaa>;

        ibm,cvc-service@40 {
                phandle = <0x110>;
                compatible = "ibm,cvc-sha512";
                reg = <0x40>;
                version = <0x1>;
        };

        ibm,cvc-service@50 {
                phandle = <0x111>;
                compatible = "ibm,cvc-verify";
                reg = <0x50>;
                version = <0x1>;
        };
};
```

4.3.2 ibm,firmware-versions node

The *ibm,firmware-versions* node contains information on the versions of various firmware components as they were **during boot**. It **does not** change if there are pending or runtime updates. It represents (to the best of boot firmware's ability) what versions of firmware were during this boot.

Property	Required	Value
version	POWER9	See below
skiboot	N	component version number
occ	N	component version number
buildroot	N	component version number
capp-ucode	N	component version number
petitboot	N	component version number
open-power	N	component version number
hostboot-binaries	N	component version number
MACHINE-xml	N	MACHINE (e.g. habanero) machine XML version
hostboot	N	component version number
linux	N	component version number

`version` property

This property **must** exist on POWER9 and above systems. It **may** exist on POWER8 systems.

If this property exists, it **must** conform to this specification. It's a single version number of the firmware image. In the event of a system supporting multiple firmware sides, this represents the **default** boot side. That is, the version that is applicable when determining if a machine requires a firmware update.

Examples (for three different platforms):

- `IBM-sandwich-20170217`
- `open-power-habanero-v1.14-45-g78d89280c3f9-dirty`
- `open-power-SUPERMICRO-P8DTU-V2.00.GA2-20161028`

To compare two versions (for the purpose of determining if the current installed firmware is in need of updating to the one being compared against) we need a defined set of rules on how to do this comparison.

Version numbers are **not** intended to be compared across platforms.

The version string may include a description at the start of it. This description can contain any set of characters but **must not** contain a '-' followed by a digit. It also **must not** contain '-v' or '-V' followed by a digit.

Each part of the version string is separated by a '-' character. Leading sections are ignored, until one starts with a digit (0-9) or a 'v' or 'V', followed by a digit. Where there is a leading 'v' or 'V', it is also stripped.

For the above three examples, we'd be left with:

- `20170217`
- `1.14-45-g78d89280c3f9-dirty`
- `2.00.GA2-20161028`

Each section is now compared until a difference is found. All comparisons are done *lexically*. The lexical comparison sorts in this order: tilde (~), all letters, non-letters. The tilde is special and sorts before an end of part. This allows the common usage of designating pre-release builds by a tailing section beginning with a '~'.

For example: "1.0~20170217", "1.0~rc4" and "1.0~beta1" all sort **before** "1.0"

Note that "1.0beta" sorts **after** "1.0"

The start of the version string contains an optional *epoch*. If not present, it is zero. This allows a reset of versioning schemes. All versions with an epoch of N+1 are greater than those with epoch N, no matter what the version strings would compare. For example "0:4.0" is **less** than "1:1.0". Increasing the epoch should **not** be a regular occurance.

For the remainder of the version strings, each part (separated by '.' or '-') is compared lexically. There are two exceptions: any part beginning with "-g" or "-p" followed by a hexadecimal string is compared as a string, and if they are different the versions are determined to be different. For example, the sections "-g78d89280c3f9" and "-g123456789abc" differ and for all comparisons (less than, greater than, equal) the result should be true.

For those who have been paying attention, this scheme should look very familiar to those who are familiar with RPM and Debian package versioning.

The below table shows comparisons between versions and what the result should be:

A	B	Result
1.14-45-g78d89280c3f9-dirty	1.14-45-g78d89280c3f9-dirty	Equal
1.14-45-g78d89280c3f9-dirty	1.14-45-g78d89280c3f9	A > B
1.14-45-g78d89280c3f9-dirty	1.14-45-g123456789abc	A < B, A > B, A != B
1.14-45-g78d89280c3f9-dirty	1.14-46	A < B
1.14-45-g78d89280c3f9-dirty	1.15	A < B
1.14-45-g78d89280c3f9-dirty	1:1.0	A < B
1.0	1.0~daily20170201	A > B
1.0.1	1.0~daily20170201	A > B
1.0	1.0.1	A < B
1.0	1.0beta	A < B

Examples

New style (required for POWER9 and above):

```
ibm,firmware-versions {
             version = "open-power-habanero-v1.14-45-g78d89280c3f9-dirty";
             skiboot = "5.4.0";
             occ = "d7efe30";
             linux = "4.4.32-openpower1";
};
```

Old-style:

```
ibm,firmware-versions {
        occ = "d7efe30-opdirty";
        skiboot = "5.4.0-opdirty";
        buildroot = "211bd05";
        capp-ucode = "1bb7503-opdirty";
        petitboot = "v1.3.1-opdirty-d695626";
        open-power = "habanero-f7b8f65-dirty";
        phandle = <0x1000012e>;
        hostboot-binaries = "56532f5-opdirty";
        habanero-xml = "6a78496-opdirty-526ff79";
        hostboot = "09cfacb-opdirty";
        linux = "4.4.32-openpower1-opdirty-85cf528";
};
```

4.3.3 ibm,opal

ibm,opal/diagnostics device tree entries

The diagnostics node under ibm,opal describes a userspace-to-firmware interface, supporting the runtime processor recovery diagnostics functions.

Note: Some systemd init scripts look for the presence of the path `/ibm,opal/diagnostics` in order to run the opal-prd daemon.

The properties of a prd node are:

```
/ {
   ibm,opal {
```

(continues on next page)

(continued from previous page)

```
      diagnostics {
        compatible = "ibm,opal-prd";
    };
  };
};
```

System Firmware

The 'firmware' node under 'ibm,opal' lists system and OPAL firmware version.

```
firmware {
      symbol-map = <0x0 0x300ac650 0x0 0x1b3f5>;
      compatible = "ibm,opal-firmware";
      ml-version = [4d 4c 20 46 57 37 37 30 2e 32 30 20 46 57 37 37 30 2e 32 30 20 46␣
↪57 37 37 30 2e 32 30];
      mi-version = <0x4d49205a 0x4c373730 0x5f303735 0x205a4c37 0x37305f30 0x3735205a␣
↪0x4c373730 0x5f303735>;
      version = "skiboot-5.0-rc2";
      phandle = <0x8e>;
      linux,phandle = <0x8e>;
};
```

compatible property describes OPAL compatibility.

symbol-map property describes OPAL symbol start address and size.

version property describes OPAL version. Replaces 'git-id', so may not be present. On POWER9 and above, it is always present.

mi-version property describes Microcode Image. Only on IBM FSP systems. Will (likely) not be present on POWER9 systems.

ml-version property describes Microcode Level. Only on IBM FSP systems. Will (likely) not be present on POWER9 systems.

MI/ML format

```
<ML/MI> <T side version> <P side version> <boot side version>
```

ibm,opal/flash device tree entries

The flash@<n> nodes under ibm,opal describe flash devices that can be accessed through the OPAL_FLASH_{READ,ERASE,WRITE} interface.

These interfaces take an 'id' parameter, which corresponds to the ibm,opal-id property of the node.

The properties under a flash node are:

- `compatible = "ibm,opal-flash"`

ibm,opal-id = <id> provides the index used for the OPAL_FLASH_XXX calls to reference this flash device

reg = <0 size> the offset and size of the flash device

ibm,flash-block-size the read/write/erase block size for the flash interface. Calls to read/write/erase must be aligned to the block size.

#address-cells = <1>, #size-cells = <1> flash devices are currently 32-bit addressable

If valid partitions are found on the flash device, then `partition@<offset>` sub-nodes are added to the flash node. These match the Linux binding for flash partitions; the reg parameter contains the offset and size of the partition.

Example:

```
flash@0 {
  reg = <0x0 0x4000000>;
  compatible = "ibm,opal-flash";
  ibm,opal-id = <0x0>;
  ibm,flash-block-size = <0x1000>;
  #address-cells = <0x1>;
  phandle = <0x100002bf>;
  #size-cells = <0x1>;
};
```

Service Indicators (LEDS)

The ‘leds’ node under ‘ibm,opal’ lists service indicators available in the system and their capabilities.

```
leds {
        compatible = "ibm,opal-v3-led";
        phandle = <0x1000006b>;
        linux,phandle = <0x1000006b>;
        led-mode = "lightpath";

        U78C9.001.RST0027-P1-C1 {
                led-types = "identify", "fault";
                phandle = <0x1000006f>;
                linux,phandle = <0x1000006f>;
        };
        /* Other LED nodes like the above one */
};
```

compatible property describes LEDs compatibility.

led-mode property describes service indicator mode (lightpath/guidinglight).

Each node under ‘leds’ node describes location code of FRU/Enclosure.

The properties under each node:

led-types Supported indicators (attention/identify/fault).

These LEDs can be accessed through OPAL_LEDS_{GET/SET}_INDICATOR interfaces. Refer to *Service Indicators (LEDS)* for interface details.

nvram Device Tree Node

```
nvram {
        compatible = "ibm,opal-nvram";
        #bytes = <0x90000>;
};
```

Indicates support (and size of) the *OPAL NVRAM* facility.

Operator Panel (oppanel)

```
oppanel {
      compatible = "ibm,opal-oppanel";
      #lines = <0x2>;
      #length = <0x10>;
};
```

The Operator Panel is a device for displaying small amounts of textual data to an administrator. On IBM POWER8 systems with an FSP, this is a small 16x2 LCD panel that can be viewed either from the Web UI of the FSP (known as ASM) or by physically going to the machine and looking at the panel.

The operator panel does not have to be present.

If it is, there are OPAL calls to read and write to it.

The device tree entry is so that the host OS knows the size of the panel and can pass buffers of the appropriate size to the OPAL calls.

ibm,opal/power-mgt device tree entries

ibm,opal/power-mgt/occ device tree entries

This node exports the per-chip pstate table properties to kernel.

Example:

```
occ@7ffddf8000 {
       ibm,pstate-vdds = [45 45 46 46 46 47 48 49 4a 4b 4c 4d 4f 50 51 52 53 54 55 57␣
↪58 59 5a 5b 5c 5d 5e 5f 5f 60 61 62 63 64 65 65 66 67 68 69 6a 6a 6b 6c 6d 6e 6f 70␣
↪70 71];
       ibm,chip-id = <0x1>;
       phandle = <0x100003b8>;
       ibm,pstate-vcss = [3b 3d 3f 41 42 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f 50 50 51␣
↪52 53 54 55 56 56 57 57 58 58 59 59 5a 5a 5b 5b 5c 5c 5d 5d 5e 5e 5f 5f 60 60 61 61␣
↪62 62];
       reg = <0x7f 0xfddf8000 0xb98>;
};
```

ibm,chip-id

This property denotes the ID of chip to which OCC belongs to.

reg

This tuple gives the statring address of the OPAL data in HOMER and the size of the OPAL data.

The top-level /ibm,opal/power-mgt contains :

```
#size-cells = <1>
#address-cells = <2>
```

ibm,pstate-vcss ibm,pstate-vdds

These properties list a voltage-identifier of each of the pstates listed in ibm,pstate-ids for the Vcs and Vdd values used for that pstate in that chip. Each VID is a single byte.

ibm,opal/power-mgt/freq-domain-mask

This property is a bitmask which will have different value depending upon the generation of the processor. Frequency domain would indicate group of CPUs which would share same frequency. Bitwise AND is taken between this bitmask value and PIR of cpu. All the CPUs lying in the same frequency domain will have same result for AND. Thus frequency management can be done based on frequency domain. A frequency domain may be a core or a quad, etc depending upon the generation of the processor.

For example, for POWER8 0xFFF8 indicates core wide frequency domain. Taking AND with the PIR of CPUs will yield us a frequency domain which is core wide distribution as last 3 bits have been masked which represent the threads.

For POWER9, 0xFFF0 indicates quad wide frequency domain. Taking AND with the PIR of CPUs will yield us frequency domain which is quad wise distribution as last 4 bits have been masked which represent the cores.

ibm,opal/power-mgt/domain-runs-at

There are two strategies in which the OCC can change the frequency of the cores in the quad on P9. 1) FREQ_MAX_IN_DOMAIN : the OCC sets the frequency of the quad to the maximum of the latest frequency requested by each of the component cores. 2) FREQ_MOST_RECENTLY_SET : the OCC sets the frequency of the quad to the most recent frequency value requested by the CPUs in the quad

In case of P8, the domain is the core and the strategy by default is FREQ_MOST_RECENTLY_SET since the PMCRs of the threads in the core are mirrored. However on P9, the domain is quad and the strategy is FREQ_MAX_IN_DOMAIN since each core has its own PMCR.

domain-runs-at denotes the strategy which OCC is using to change the frequency of a frequency domain.

power-mgt/powercap

The powercap sensors are populated in this node. Each child node in the "powercap" node represents a power-cappable component.

For example :

```
system-powercap/
```

The *OPAL_GET_POWERCAP* and *OPAL_SET_POWERCAP* calls take a handle for what powercap property to get/set which is defined in the child node.

The compatible property for the linux driver which will be "ibm,opal-powercap"

Each child node has below properties:

powercap-current Handle to indicate the current powercap

powercap-min Absolute minimum possible powercap. This points to the soft powercap minimum limit as exported by OCC. The powercap set in the soft powercap range may or may not be maintained.

powercap-max Maximum possible powercap

powercap-hard-min This value points to the hard minimum powercap limit. The powercap set above this limit is guaranteed unless there is a hardware failure

Powercap handle uses the following encoding:

```
| Class |     Reserved    | Attribute |
|-------|-----------------|-----------|
```

Note: The format of the powercap handle is `NOT` ABI and may change in the future.

```
power-mgt {
  powercap {
     compatible = "ibm,opal-powercap";

     system-powercap {
             name = "system-powercap";
             powercap-current = <0x00000002>;
             powercap-min = <0x00000000>;
             powercap-max = <0x00000001>;
             powercap-hard-min = <0x000000003>;
     };
  };
 };
```

power-mgt/psr

Some systems allow modification of how power consumption throttling is balanced between entities in a system. A typical one may be how the power management complex should balance throttling CPU versus the GPU. An OPAL call can be used to set these ratios, which are described in the device tree.

In the future, there may be more available settings than just CPU versus GPU.

Each child node in the "psr" node represents a configurable psr sensor.

For example [::] cpu-to-gpu@1

The compatible property is set to "ibm,opal-power-shift-ratio".

Each child node has below properties:

handle Handle to indicate the type of psr

label Name of the psr sensor

The format of the handle is internal, and `not` ABI, although currently it uses the following encoding

```
| Class |Reserved|  RID | Type |
|-------|--------|------|------|
```

```
power-mgt {
  psr {
     compatible = "ibm,opal-power-shift-ratio";

     cpu-to-gpu@0 {
             name = "cpu-to-gpu";
             handle = <0x00000000>;
             label = "cpu_to_gpu_0";
     };
```

(continues on next page)

(continued from previous page)

```
     cpu-to-gpu@1 {
             name = "cpu-to-gpu";
             handle = <0x00000100>;
             label = "cpu_to_gpu_1";
     };
  };
 };
```

All available CPU idle states are listed in ibm,cpu-idle-state-names

For example:

```
power-mgt {
  ibm,cpu-idle-state-names = "nap", "fastsleep_", "winkle";
  ibm,cpu-idle-state-residency-ns = <0x1 0x2 0x3>;
  ibm,cpu-idle-state-latencies-ns = <0x1 0x2 0x3>;
};
```

The idle states are characterized by latency and residency numbers which determine the breakeven point for entry into them. The latency is a measure of the exit overhead from the idle state and residency is the minimum amount of time that a CPU must be predicted to be idle so as to reap the powersavings from entering into that idle state.

These numbers are made use of by the cpuidle governors in the kernel to arrive at the appropriate idle state that a CPU must enter into when there is no work to be done. The values in ibm,cpu-idle-state-latencies-ns are the the measured latency numbers for the idle states. The residency numbers have been arrived at experimentally after ensuring that the performance of latency sensitive workloads do not regress while allowing deeper idle states to be entered into during low load situations. The kernel is expected to use these values for optimal power efficiency.

Example:

```
/ {
  ibm,opal {
    power-mgt {
             ibm,pstate-frequencies-mhz = <0xda3 0xd82 0xd60 0xd3f 0xd1e 0xcfd 0xcdb␣
↪0xcba 0xc99 0xc78 0xc56 0xc35 0xc14 0xbf3 0xbd1 0xbb0 0xb8f 0xb6e 0xb4c 0xb2b 0xb0a␣
↪0xae9 0xac7 0xaa6 0xa85 0xa64 0xa42 0xa21 0xa00 0x9df 0x9bd 0x99c 0x97b 0x95a 0x938␣
↪0x917 0x8f6 0x8d5 0x8b3 0x892 0x871 0x850 0x82e 0x80d>;
             ibm,cpu-idle-state-latencies-ns = <0xfa0 0x9c40 0x989680>;
             ibm,cpu-idle-state-flags = <0x11000 0x81003 0x47003>;
             ibm,cpu-idle-state-names = "nap", "fastsleep_", "winkle";
             ibm,cpu-idle-state-pmicr = <0x0 0x0 0x20 0x0 0x0 0x0>;
             ibm,pstate-nominal = <0xffffffef>;
             ibm,cpu-idle-state-residency-ns = <0x186a0 0x11e1a300 0x3b9aca00>;
             ibm,cpu-idle-state-pmicr-mask = <0x0 0x0 0x30 0x0 0x0 0x0>;
             phandle = <0x100002a0>;
             ibm,pstate-ids = <0x0 0xffffffff 0xfffffffe 0xfffffffd 0xfffffffc␣
↪0xfffffffb 0xfffffffa 0xfffffff9 0xfffffff8 0xfffffff7 0xfffffff6 0xfffffff5␣
↪0xfffffff4 0xfffffff3 0xfffffff2 0xfffffff1 0xfffffff0 0xffffffef 0xffffffee␣
↪0xffffffed 0xffffffec 0xffffffeb 0xffffffea 0xffffffe9 0xffffffe8 0xffffffe7␣
↪0xffffffe6 0xffffffe5 0xffffffe4 0xffffffe3 0xffffffe2 0xffffffe1 0xffffffe0␣
↪0xffffffdf 0xffffffde 0xffffffdd 0xffffffdc 0xffffffdb 0xffffffda 0xffffffd9␣
↪0xffffffd8 0xffffffd7 0xffffffd6 0xffffffd5>;
             ibm,pstate-max = <0x0>;
             ibm,pstate-min = <0xffffffd5>;
    };
  };
};
```

ibm,cpu-idle-state-pmicr ibm,cpu-idle-state-pmicr-mask

In POWER8, idle states sleep and winkle have 2 modes- fast and deep. In fast mode, idle state puts the core into threshold voltage whereas deep mode completely turns off the core. Choosing fast vs deep mode for an idle state can be done either via PM_GP1 scom or by writing to PMICR special register. If using the PMICR path to choose fast/deep mode then ibm,cpu-idle-state-pmicr and ibm,cpu-idle-state-pmicr-mask properties expose relevant PMICR bits and values for corresponding idle states.

ibm,cpu-idle-state-psscr ibm,cpu-idle-state-psscr-mask

In POWER ISA v3, there is a common instruction 'stop' to enter any idle state and SPR PSSCR is used to specify which idle state needs to be entered upon executing stop instruction. Properties ibm,cpu-idle-state-psscr and ibm,cpu-idle-state-psscr-mask expose the relevant PSSCR bits and values for corresponding idle states.

ibm,cpu-idle-state-flags

These flags are used to describe the characteristics of the idle states like the kind of core state loss caused. These flags are used by the kernel to save/restore appropriate context while using the idle states.

ibm,pstate-ids

This property lists the available pstate identifiers, as signed 32-bit big-endian values. While the identifiers are somewhat arbitrary, these define the order of the pstates in other ibm,pstate-* properties.

ibm,pstate-frequencies-mhz

This property lists the frequency, in MHz, of each of the pstates listed in the ibm,pstate-ids file. Each frequency is a 32-bit big-endian word.

ibm,pstate-max ibm,pstate-min ibm,pstate-nominal

These properties give the maximum, minimum and nominal pstate values, as an id specified in the ibm,pstate-ids file.

ibm,pstate-ultra-turbo ibm,pstate-turbo

These properties are added when ultra-turbo(WOF) is enabled. These properties give the max turbo and max ultra-turbo pstate.

Example:

```
power-mgt {
        ibm,pstate-core-max = <0x0 0x0 0x0 0x0 0x0 0x0 0x0>;
        ibm,pstate-turbo = <0xfffffffb>
        ibm,pstate-ultra-turbo = <0x0>;
};
```

ibm,pstate-core-max

This property is added when ultra_turbo(WOF) is enabled. This property gives the list of max pstate for each 'n' number of active cores in the chip.

ibm,opal/sensor-groups

This node contains all sensor groups defined in the system. Each child node here represents a sensor group.

For example [::] occc-csm@1c00020/

The compatible property is set to "ibm,opal-sensor-group"

Each child node has below properties:

type string to indicate the sensor group

sensor-group-id Uniquely identifies a sensor group.

ibm,chip-id This property is added if the sensor group is chip specific

sensors Phandles of all sensors belonging to this sensor group

ops Array of opal call numbers to indicate the available sensor group operations

```
ibm,opal {
  sensor-groups {
     compatible = "ibm,opal-sensor-group";

     occ-csm@1c00020 {
             name = "occ-csm";
             type = "csm";
             sensor-group-id = <0x01c00020>;
             ibm,chip-id = <0x00000008>;
             ops = <0x9c>;
             sensors = <0x00000175 0x00000176 0x00000177 0x00000178 0x00000179␣
↪0x0000017a 0x0000017b 0x0000017c>;
     };
  };
 };
```

ibm,opal/sensors/ device tree nodes

All sensors of a POWER8 system are made available to the OS in the ibm,opal/sensors/ directory. Each sensor is identified with a node which name follows this pattern :

```
<resource class name>@<resource identifier>/
```

For example :

```
core-temp@20/
```

Each node has a minimum set of properties describing the sensor :

- a "compatible" property which should be "ibm,opal-sensor"
- a "sensor-type" property, which can be "temp", "fan", "power". More will be added when new resources are supported. This type is used "as is" by the Linux driver to map sensors in the sysfs interface of the hwmon framework of Linux.

- a "sensor-data" property giving a unique handler for the OPAL_SENSOR_READ call to be used by Linux to get the value of a sensor attribute. This value is opaque to the OS but is *currently* constructed using the following encoding :

  ```
  |  Attr.  |Fam|Res. |   Resource     |
  | Number  |ily|Class|      Id        |
  |---------|---|-----|----------------|
  ```

  The sensor family (FSP, DTS, etc) is used to dispatch the call to the appriopriate skiboot component.

- a "sensor-status" property giving the state of the sensor. The status bits have the slightly meanings depending on the resource type but testing against 0x6 should raise an alarm.
- an optional "label" property

Each node can have some extra properties depending on the resource they represent. See the tree below for more information.

```
ibm,opal {
  sensors {

      /*
       * Core temperatures (DTS) nodes.
       *
       * We use the PIR of the core as a resource identifier.
       */
      core-temp@20 {
              compatible = "ibm,opal-sensor";
              name = "core-temp";
              sensor-type = "temp";

              /* Status bits :
               *
               * 0x0003       FATAL
               * 0x0002       CRITICAL
               * 0x0001       WARNING
               */
              sensor-data = <0x00800020>;

              /*
               * These are extra properties to help Linux output.
               */
              ibm,pir = <0x20>;
              label = "Core";
      };

      /*
       * Centaur temperatures (DTS) nodes. Open Power only.
       *
       * We use the PIR of the core as a resource identifier.
       */
      mem-temp@1 {
              compatible = "ibm,opal-sensor";
              name = "mem-temp";
              sensor-type = "temp";

              /* Status bits :
               *
               * 0x0003       FATAL
```

(continues on next page)

(continued from previous page)

```
                 * 0x0002        CRITICAL
                 * 0x0001        WARNING
                 */
                sensor-data = <0x00810001>;

                /*
                 * These are extra properties to help Linux output.
                 */
                ibm,chip-id = <0x80000001>;
                label = "Centaur";
        };
    };
};
```

sysparams

```
/* System parameter permission */
enum OpalSysparamPerm {
     OPAL_SYSPARAM_READ  = 0x1,
     OPAL_SYSPARAM_WRITE = 0x2,
     OPAL_SYSPARAM_RW    = (OPAL_SYSPARAM_READ | OPAL_SYSPARAM_WRITE),
};
```

```
sysparams {
             compatible = "ibm,opal-sysparams";
             param-id = <0xf0000001 0xf0000003 0xf0000012 0xf0000016 0xf000001d␣
↪0xf0000023 0xf0000024 0xf0000025 0xf0000026 0xf0000027>;
             param-name = "surveillance", "hmc-management", "cupd-policy", "plat-hmc-
↪managed", "fw-license-policy", "world-wide-port-num", "default-boot-device", "next-
↪boot-device", "console-select", "boot-device-path";
             param-perm = [03 01 03 03 03 02 03 03 03 03];
             phandle = <0x10000032>;
             param-len = <0x4 0x4 0x4 0x4 0x4 0xc 0x1 0x1 0x1 0x30>;
             linux,phandle = <0x10000032>;
};
```

Device tree node for system parameters accessible through the *Get/Set System Parameters* calls *OPAL_GET_PARAM* and *OPAL_SET_PARAM*.

While many systems and platforms will support parameters and configuration via either nvram or over IPMI, some platforms may have parameters that need to be set a different way.

Some parameters may be set Read Only, so the *param-perm* property indicates permissions.

Currently, this is only something that exists on FSP based systems.

Top level ibm,opal node

```
ibm,opal {
             #address-cells = <0x0>;
             #size-cells = <0x0>;
             compatible = "ibm,opal-v2", "ibm,opal-v3";

/* v2 is maintained for possible compatibility with very, very old kernels
```

(continues on next page)

(continued from previous page)

```
 * it will go away at some point in the future. Detect and rely on ibm,opal-v3
 * ibm,opal-v2 is *NOT* present on POWER9 and above.
 */

             ibm,associativity-reference-points = <0x4 0x3, 0x2>;
             ibm,heartbeat-ms = <0x7d0>;

/* how often any OPAL call needs to be made to avoid a watchdog timer on BMC
 * from kicking in
 */

             ibm,opal-memcons = <0x0 0x3007a000>;

/* location of in memory OPAL console buffer. */

             ibm,opal-trace-mask = <0x0 0x3008c3f0>;
             ibm,opal-traces = <0x0 0x3007b010 0x0 0x10077 0x0 0x3b001010 0x0␣
↪0x1000a7 0x0 0x3b103010 0x0 0x1000a7 0x0 0x3b205010 0x0 0x1000a7 0x0 0x3b307010 0x0␣
↪0x1000a7 0x0 0x3b409010 0x0 0x1000a7 0x10 0x1801010 0x0 0x1000a7 0x10 0x1903010 0x0␣
↪0x1000a7 0x10 0x1a05010 0x0 0x1000a7 0x10 0x1b07010 0x0 0x1000a7 0x10 0x1c09010 0x0␣
↪0x1000a7 0x10 0x1d0b010 0x0 0x1000a7 0x10 0x1e0d010 0x0 0x1000a7 0x10 0x1f0f010 0x0␣
↪0x1000a7 0x10 0x2011010 0x0 0x1000a7 0x10 0x2113010 0x0 0x1000a7 0x10 0x2215010 0x0␣
↪0x1000a7 0x10 0x2317010 0x0 0x1000a7 0x10 0x2419010 0x0 0x1000a7 0x10 0x251b010 0x0␣
↪0x1000a7 0x10 0x261d010 0x0 0x1000a7>;

/* see docs on tracing */

             linux,phandle = <0x10000003>;
             opal-base-address = <0x0 0x30000000>;
             opal-entry-address = <0x0 0x300050c0>;
             opal-interrupts = <0x10 0x11 0x12 0x13 0x14 0x20010 0x20011 0x20012␣
↪0x20013 0x20014 0xffe 0xfff 0x17fe 0x17ff 0x2ffe 0x2fff 0x37fe 0x37ff 0x20ffe␣
↪0x20fff 0x22ffe 0x22fff 0x237fe 0x237ff>;
             opal-msg-async-num = <0x8>;
             opal-msg-size = <0x48>;
             opal-runtime-size = <0x0 0x9a00000>;
             phandle = <0x10000003>;
 };
```

ibm,heartbeat-ms

```
ibm,opal {
             ibm,heartbeat-ms = <0x7d0>;
}
```

Any OS targetting POWER9 processors *must* respect *ibm,heartbeat-ms*.

Linux kernels prior to v4.1-rc1 would ignore *ibm,heartbeat-ms*. These only supported POWER8 systems.

On the earliest POWER8 OPAL systems, there was *ibm,heartbeat-freq* instead. However, no OS at the time ever looked at that value, so it can be ignored by any new operating systems.

fast-reboot property

This property of the *ibm,opal* node is an option property that will either be the string *okay* or the reason the fast reboot feature was disabled on boot.

The motivation behind adding this property is to help the OPAL test suite work out if it should even try the fast reboot test on a particular platform (without it having to resort to grepping firmware logs).

4.3.4 ibm,secureboot

The `ibm,secureboot` node provides secure boot and trusted boot information up to the target OS. Further information can be found in *Secure and Trusted Boot Library (LibSTB) Documentation*.

Required properties

```
compatible:         Either one of the following values:

                    ibm,secureboot-v1  :  The container-verification-code
                                          is stored in a secure ROM memory.

                    ibm,secureboot-v2  :  The container-verification-code
                                          is stored in a reserved memory.
                                          It described by the ibm,cvc child
                                          node.

secure-enabled:     this property exists when the firmware stack is booting
                    in secure mode (hardware secure boot jumper asserted).

trusted-enabled:    this property exists when the firmware stack is booting
                    in trusted mode.

hw-key-hash:        hash of the three hardware public keys trusted by the
                    platformw owner. This is used to verify if a firmware
                    code is signed with trusted keys.

hw-key-hash-size:   hw-key-hash size
```

Obsolete properties

```
hash-algo:          Superseded by the hw-key-hash-size property in
                    'ibm,secureboot-v2'.
```

Example

```
ibm,secureboot {
    compatible = "ibm,secureboot-v2";
    secure-enabled;
    trusted-enabled;
    hw-key-hash-size = <0x40>;
    hw-key-hash = <0x40d487ff 0x7380ed6a 0xd54775d5 0x795fea0d 0xe2f541fe
                   0xa9db06b8 0x466a42a3 0x20e65f75 0xb4866546 0x0017d907
```

(continues on next page)

(continued from previous page)

```
                    0x515dc2a5 0xf9fc5095 0x4d6ee0c9 0xb67d219d 0xfb708535
                    0x1d01d6d1>;
    phandle = <0x100000fd>;
    linux,phandle = <0x100000fd>;
};
```

4.3.5 IMC Device Tree Bindings

See *OPAL/Skiboot In-Memory Collection (IMC) interface Documentation* for general In-Memory Collection (IMC) counter information.

imc-counters top-level node

```
imc-counters {
  compatible = "ibm,opal-in-memory-counters";
  #address-cells = <0x1>;
  #size-cells = <0x1>;
  phandle = <0x1000023a>;
  version-id = <0xd>;
  /* Denote IMC Events Catalog version used to build this DTS file. */

};
```

IMC device/units bindings

```
mcs3 {
        compatible = "ibm,imc-counters";
        events-prefix = "PM_MCS3_"; /* denotes event name to be prefixed to get␣
↪complete event name supported by this device */

        phandle = <0x10000241>;
        events = <0x10000242>; /* phandle of the events node supported by this device␣
↪*/

        unit = "MiB";
        scale = "4"; /* unit and scale for all the events for this device */

        reg = <0x118 0x8>; /* denotes base address for device event updates */
        type = <0x10>;
        size = 0x40000;
        offset = 0x180000;
        base_addr = <Base address of the counter in reserve memory>
        /* This is per-chip memory field and OPAL files it based on the no of chip in␣
↪the system */
        /* base_addr property also indicates (or hints) kernel whether to memory */
        /* should be mmapped or allocated at system start for the counters */
        chipids = <chip-id for the base_addr >
};

trace@0 {
         compatible = "ibm,imc-counters";
```

(continues on next page)

(continued from previous page)

```
        events-prefix = "trace_";
        reg = <0x0 0x8>;
        events = < &TRACE_IMC >;
        type = <0x2>;
        size = <0x40000>;
};
```

IMC device event bindings

```
nest-mcs-events {
        #address-cells = <0x1>;
        #size-cells = <0x1>;
        phandle = <0x10000242>;

        event@98 {
              desc = "Total Write Bandwidth seen on both MCS"; /* event description */

              phandle = <0x1000023d>;
              reg = <0x98 0x8>; /* event offset,when added with (nest-offset-address␣
↪+ device reg) will point to actual counter memory */

              event-name = "DOWN_128B_DATA_XFER"; /* denotes the actual event name */

        };

        /* List of events supported */

};

TRACE_IMC: trace-events {
      #address-cells = <0x1>;
      #size-cells = <0x1>;

      event@10200000 {
            event-name = "cycles" ; /* For trace node, we only have cycles event now␣
↪*/
            reg = <0x10200000 0x8>;
            desc = "Reference cycles" ;
      };
};
```

Trace-mode SCOM

Trace scom is a 64 bit value which contains the event information for IMC-trace mode. Following is the trace-scom layout.

TRACE_IMC_SCOM bit representation

0-1 SAMPSEL

2-33 CPMC_LOAD

34-40 CPMC1SEL

41-47 CPMC2SEL

48-50 BUFFERSIZE

51-63 RESERVED

CPMC_LOAD contains the sampling duration. *SAMPSEL* and *CPMC*SEL* determines the event to count. *BUFFRSIZE* indicates the memory range.

BUFFERSIZE can be

```
b'000' - 4K entries * 64 per entry = 256K
b'001' - 8K entries * 64 per entry = 512K
b'010' - 16K entries * 64 per entry = 1M
b'011' - 32K entries * 64 per entry = 2M
b'100' - 64K entries * 64 per entry = 4M
```

4.3.6 P9 memory hierarchy

P9 Nimbus supports direct attached DDR memory through 4 DDR ports per side of the processor. Device tree contains memory hierarchy so that one can traverse from chip to DIMM like below:

xscom@<addr>/mcbist@<mcbist_id>/mcs@<mcs_id>/mca@<mca_id>/dimm@<resource_id>

Example of dimm node: .. code-block:: dts

dimm@d00e { memory-id = <0xc>; /* DRAM Device Type. 0xc = DDR4 / *product-version = <0x32>;* / Module Revision Code / *device_type = "memory-dimm-ddr4"; serial-number = <0x15d9ad1c>; status = "okay"; size = <0x4000>; phandle = <0xd2>; ibm,loc-code = "UOPWR.0000000-Node0-DIMM14"; part-number = "36ASF2G72PZ-2G6B2 "; reg = <0xd00e>; manufacturer-id = <0x802c>;* / Vendor ID, we can get vendor name from this ID */

};

4.3.7 Nvlink Device Tree Bindings

See *OPAL/Skiboot Nvlink Interface Documentation* for general Nvlink information.

NPU bindings:

```
xscom@3fc0000000000 {
  npu@8013c00 {
    reg = <0x8013c00 0x2c>;
    compatible = "ibm,power8-npu";
    ibm,npu-index = <0x0>;
    ibm,npu-links = <0x4>; /* Number of links wired up to this npu. */

    phandle = <0x100002bc>;
    linux,phandle = <0x100002bc>;

    link@0 {
      ibm,npu-pbcq = <0x1000000b>; /* phandle to the pbcq which connects to the GPU.␣
↪*/
      ibm,npu-phy = <0x80000000 0x8010c3f>; /* SCOM address of the IBM PHY␣
↪controlling this link. */
      compatible = "ibm,npu-link";
      ibm,npu-lane-mask = <0xff>; /* Mask specifying which IBM PHY lanes are used for␣
↪this link. */
```

(continues on next page)

(continued from previous page)

```
      phandle = <0x100002bd>;
      ibm,npu-link-index = <0x0>; /* Hardware link index. Naples systems
                                    * contain links at index 0,1,4 & 5.
                                    * Used to calculate various address offsets. */

      linux,phandle = <0x100002bd>;
    };

    link@1 {
      ibm,npu-pbcq = <0x1000000b>;
      ibm,npu-phy = <0x80000000 0x8010c3f>;
      compatible = "ibm,npu-link";
      ibm,npu-lane-mask = <0xff00>;
      phandle = <0x100002be>;
      ibm,npu-link-index = <0x1>;
      linux,phandle = <0x100002be>;
    };

    link@4 {
      ibm,npu-pbcq = <0x1000000a>;
      ibm,npu-phy = <0x80000000 0x8010c7f>;
      compatible = "ibm,npu-link";
      ibm,npu-lane-mask = <0xff00>;
      phandle = <0x100002bf>;
      ibm,npu-link-index = <0x4>;
      linux,phandle = <0x100002bf>;
    };

    link@5 {
      ibm,npu-pbcq = <0x1000000a>;
      ibm,npu-phy = <0x80000000 0x8010c7f>;
      compatible = "ibm,npu-link";
      ibm,npu-lane-mask = <0xff>;
      phandle = <0x100002c0>;
      ibm,npu-link-index = <0x5>;
      linux,phandle = <0x100002c0>;
    };
  };
};
```

GPU memory bindings

```
memory@100000000 {
        device_type = "memory"
        compatible = "ibm,coherent-device-memory";
        linux,usable-memory = <0x0 0x100000000 0x0 0x0>;

; denotes a region of unplugged system memory

        reg = <0x0 0x100000000 0x0 0x80000000>;
        ibm,associativity = <0x4 0x0 0x0 0x0 0x64>;

; numa associativity for the memory once it is hotplugged

        phandle = <0x10000abc>;
```

(continues on next page)

(continued from previous page)

```
        linux,phandle = <0x10000abc>;
};
```

Emulated PCI device bindings

```
pciex@3fff000400000 {
        ibm,npcq = <0x100002bc>; /* phandle to the NPU node. Used to find associated␣
↪PCI GPU devices. */
        compatible = "ibm,power8-npu-pciex", "ibm,ioda2-npu-phb";

        pci@0 {
                reg = <0x0 0x0 0x0 0x0 0x0>;
                revision-id = <0x0>;
                interrupts = <0x1>;
                device-id = <0x4ea>;
                ibm,pci-config-space-type = <0x1>;
                vendor-id = <0x1014>;
                ibm,gpu = <0x100002f7>; /* phandle pointing the associated GPU PCI␣
↪device node */
                memory-region = <0x10000abc>; /* phandle pointing to the GPU memory␣
↪*/
                ibm,nvlink-speed = <0x1>;

        ; Denotes the speed the link is running at:
        ; 0x3 == 20 Gbps, 0x8 = 25.78125 Gbps, 0x9 == 25.00000 Gbps

                phandle = <0x100002fc>;
        };

        pci@1 {
                reg = <0x800 0x0 0x0 0x0 0x0>;
                revision-id = <0x0>;
                interrupts = <0x1>;
                device-id = <0x4ea>;
                ibm,pci-config-space-type = <0x1>;
                vendor-id = <0x1014>;
                ibm,gpu = <0x100002f5>;
                memory-region = <0x10000def>;
                phandle = <0x100002fe>;
                class-code = <0x60400>;
                linux,phandle = <0x100002fe>;
        };

        pci@0,1 {
                reg = <0x100 0x0 0x0 0x0 0x0>;
                revision-id = <0x0>;
                interrupts = <0x2>;
                device-id = <0x4ea>;
                ibm,pci-config-space-type = <0x1>;
                vendor-id = <0x1014>;
                ibm,gpu = <0x100002f7>;
                memory-region = <0x10000abc>;
                phandle = <0x100002fd>;
                class-code = <0x60400>;
                linux,phandle = <0x100002fd>;
```

(continues on next page)

(continued from previous page)

```
        };

        pci@1,1 {
               reg = <0x900 0x0 0x0 0x0 0x0>;
                revision-id = <0x0>;
                interrupts = <0x2>;
                device-id = <0x4ea>;
                ibm,pci-config-space-type = <0x1>;
                vendor-id = <0x1014>;
                ibm,gpu = <0x100002f5>;
                memory-region = <0x10000def>;
                phandle = <0x100002ff>;
                class-code = <0x60400>;
                linux,phandle = <0x100002ff>;
        };
};
```

4.3.8 Nest (NX) Accelerator Coprocessor

The NX coprocessor is present in P7+ or later processors. Each NX node represents a unique NX coprocessor. The nodes are located under an xscom node, as:

```
/xscom@<xscom_addr>/nx@<nx_addr>
```

With unique xscom and nx addresses. Their compatible node contains "ibm,power-nx".

NX Compression Coprocessor

This is the memory compression coprocessor. which uses the IBM proprietary 842 compression algorithm and format. Each NX node contains an 842 engine.

```
ibm,842-coprocessor-type     : CT value common to all 842 coprocessors
ibm,842-coprocessor-instance : CI value unique to all 842 coprocessors
```

Access to the coprocessor requires using the ICSWX instruction, which uses a specific format including a Coprocessor Type (CT) and Coprocessor Instance (CI) value to address each request to the right coprocessor. The driver should use the CT and CI values for a particular node to communicate with it. For all 842 coprocessors in the system, the CT value will (should) be the same, while each will have a different CI value. The driver can use CI 0 to allow the hardware to automatically select which coprocessor instance to use.

On P9, this compression coprocessor also supports standard GZIP/ZLIB compression algorithm and format. Virtual Accelerator Swirchboard (VAS) is used to access this coprocessor. VAS writes each request to receive FIFOs (RXFIFO) which are either high or normal priority and these FIFOs are bound to coprocessor types (842 and gzip).

VAS distinguishes NX requests for the target engines based on logical partition ID (lpid), process ID (pid) and Thread ID (tid). So (lpid, pid, tid) combination has to be unique in the system. Each NX node contains high and normal FIFOs for each 842 and GZIP engines.

```
/ibm,842-high-fifo           : High priority 842 RxFIFO
/ibm,842-normal-fifo         : Normal priority 842 RxFIFO
/ibm,gzip-high-fifo          : High priority gzip RxFIFO
/ibm,gzip-normal-fifo        : Normal priority gzip RxFIFO
```

Each RxFIFO node contains:

```
compatible                : ibm,p9-nx-842 or ibm,p9-nx-gzip
priority                  : High or Normal
rx-fifo-address           : RxFIFO buffer address
rx-fifo-size              : RxFIFO size
lpid                      : 0xfff (1's for 12 bits in UMAC notify match
                            register)
pid                       : Coprocessor type (either 842 or gzip)
tid                       : counter in each coprocessor type
```

During initialization, the driver invokes VAS interface for each coprocessor type (842 and gzip) to configure the RxFIFO with rx_fifo_address, lpid, pid and tid for high and nornmal priority FIFOs.

NX RNG Coprocessor

This is the Random Number Generator (RNG) coprocessor, which is a part of each NX coprocessor. Each node represents a unique RNG coprocessor. Its nodes are not under the main nx node, they are located at:

```
/hwrng@<addr>           : RNG at address <addr>
ibm,chip-id             : chip id where the RNG is
reg                     : address of the register to read from
```

Each read from the RNG register will provide a new random number.

4.3.9 OpenCAPI Device Tree Bindings

NPU bindings

The NPU nodes are similar to those in *Nvlink Device Tree Bindings*.

We distinguish between OpenCAPI and NVLink links using the *ibm.npu-link-type* property. NPUs with a mixture of OpenCAPI and NVLink links are currently unsupported.

```
xscom@603fc00000000 {
  npu@5011000 {
    compatible = "ibm,power9-npu";
    phandle = <0xe6>;
    ibm,phb-index = <0x7>;
    reg = <0x5011000 0x2c>;
    ibm,npu-index = <0x0>;
    ibm,npu-links = <0x2>; /* Number of links wired up to this npu. */

    link@2 {
      compatible = "ibm,npu-link";
      ibm,npu-link-type = "opencapi";
      ibm,npu-group-id = <0x1>;
      ibm,npu-lane-mask = <0xf1e000>; /* Mask specifying which IBM PHY lanes
                                        * are used for this link. 24-bit,
                                        * lane 0 is most significant bit */
      ibm,npu-phy = <0x80000000 0x9010c3f>; /* SCOM address of the IBM PHY
                                              * controlling this link. */
      ibm,npu-link-index = <0x2>; /* Hardware link index.
                                    * Used to calculate various address offsets. */
      phandle = <0xe7>;
    };
```

(continues on next page)

(continued from previous page)

```
    link@3 {
      compatible = "ibm,npu-link";
      ibm,npu-link-type = "opencapi";
      ibm,npu-group-id = <0x2>;
      ibm,npu-lane-mask = <0x78f>;
      ibm,npu-phy = <0x80000000 0x9010c3f>;
      ibm,npu-link-index = <0x3>;
      phandle = <0xe8>;
    };
  };
};
```

PCI device bindings

The PCI devices mostly look like regular PCI devices (see *PCI Device Tree Bindings*), but have a few additional fields to allow the devices to be associated with the relevant NPU. These fields are presently not consumed by anything but may be used in future.

```
pciex@600e800000000 {
  /* OpenCAPI specific properties */
  compatible = "ibm,power9-npu-opencapi-pciex", "ibm,ioda2-npu2-opencapi-phb";
  ibm,npcq = <0xe6>; /* phandle to the NPU node */
  ibm,npu-index = <0x0>;
  ibm,links = <0x1>;
  /* Generic PCI fields here */
}
```

4.3.10 PCI Device Tree Bindings

The following is an example PCI host bridge and device from a POWER9 machine.

```
pciex@600c3c0300000 {
  ibm,capi-flags = <0x1>;
  ibm,phb-stack-index = <0x0>;
  compatible = "ibm,power9-pciex", "ibm,ioda3-phb";
  ibm,opal-single-pe;
  ibm,opal-num-pes = <0x200>;
  ibm,supported-tce-sizes = <0xc 0x10 0x15 0x1e>;
  device_type = "pciex";
  ibm,opal-peltv-table = <0x0 0x543c0000 0x20000>;
  ibm,associativity = <0x4 0x0 0x0 0x1 0x0>;
  ibm,phb-diag-data-size = <0x2180>;
  ranges = <0x2000000 0x0 0x80000000 0x600c1 0x80000000 0x0 0x7fff0000>;
  ibm,lane-eq = <0x54545454 0x54545454 0x54545454 0x54545454 0x54545454
                 0x54545454 0x54545454 0x54545454 0x77777777 0x77777777
                 0x77777777 0x77777777>;
  status = "okay";
  #interrupt-cells = <0x1>;
  bus-range = <0x0 0xff>;
  interrupt-parent = <0x126>;
  #address-cells = <0x3>;
  ibm,opal-phbid = <0x0 0x3>;
  ibm,opal-pest-table = <0x0 0x543e2000 0x2000>;
```

(continues on next page)

(continued from previous page)

```
  ibm,chip-id = <0x0>;
  #size-cells = <0x2>;
  ibm,opal-m64-segment-splits = <0x200 0x1 0xc 0x0 0x1 0x2 0xc 0x0>;
  ibm,opal-m64-window = <0x60200 0x0 0x60200 0x0 0x40 0x0>;
  phandle = <0x617>;
  ibm,phb-stack = <0xd8>;
  ibm,phb-index = <0x3>;
  ibm,phb-pec-index = <0x2>; /* skiboot-v6.2 or later */
  reg = <0x600c3 0xc0300000 0x0 0x1000 0x600c3 0x60000000 0x0 0x10000000>;
  ibm,mmio-windows = <0x60200 0x0 0x40 0x0 0x600c1 0x80000000 0x0 0x80000000>;
  clock-frequency = <0x200 0x0>;
  ibm,xscom-bases = <0x4011400 0x4011440 0xf010800 0xf010840 0xf010900>;
  ibm,opal-reserved-pe = <0x1ff>;
  ibm,capp-timebase-sync = [00];
  ibm,opal-available-m64-ranges = <0x1 0x1f>;
  ibm,opal-rtt-table = <0x0 0x54380000 0x20000>;
  ibm,opal-msi-ranges = <0xfc000 0xff8>;

  pci@0 {
    device_type = "pciex";
    revision-id = <0x0>;
    ibm,pci-config-space-type = <0x1>;
    interrupt-map-mask = <0x0 0x0 0x0 0x7>;
    class-code = <0x60400>;
    ranges = <0x2000000 0x0 0x0 0x2000000 0x0 0x0 0xf0000000 0x0>;
    vendor-id = <0x1014>;
    #interrupt-cells = <0x1>;
    #address-cells = <0x3>;
    interrupt-map = <0x0 0x0 0x0 0x1 0x126 0xfcff8 0x1 0x0 0x0 0x0 0x2 0x126
                     0xfcff9 0x1 0x0 0x0 0x0 0x3 0x126 0xfcffa 0x1 0x0 0x0 0x0
                     0x4 0x126 0xfcffb 0x1>;
    #size-cells = <0x2>;
    device-id = <0x4c1>;
    phandle = <0x622>;
    reg = <0x0 0x0 0x0 0x0 0x0>;
  };
};
```

4.3.11 reserved-memory device tree nodes

OPAL exposes reserved memory through a top-level reserved-memory node, containing subnodes that represent each reserved memory region.

This follows the Linux specification for the /reserved-memory node, described in the kernel source tree, in:

> Documentation/devicetree/bindings/reserved-memory/reserved-memory.txt

The top-level /reserved-memory node contains:

```
#size-cells = <2>
#address-cells = <2>
```

Addresses and sizes are all 64-bits.

`ranges` the empty ranges node indicates no translation of physical addresses in the subnodes.

The sub-nodes under the /reserved-memory node contain:

reg = <address size> the address and size of the reserved memory region. The address and size values are two cells each, as signified by the top-level `#{address,size}-cells`

ibm,prd-label = "string" a string token for use by the prd system. Specific ranges may be used by prd - those will be referenced by this label.

4.3.12 Trusted Platform Module (TPM)

The tpm node describes a TPM device present in the platform. It also includes event log information.

Required properties

All these properties are consumed by skiboot and the linux kernel (tpm and vtpm codes)

```
compatible :       should have "nuvoton,npct650"

linux,sml-base:    64-bit base address of the reserved memory allocated for firmware␣
↪event log.
                   sml stands for shared memory log.

linux,sml-size:    size of the memory allocated for firmware event log.
```

Optional properties

```
status:            indicates whether the device is enabled or disabled. "okay" for
                   enabled and "disabled" for disabled.
```

Example

```
tpm@57 {
    reg = <0x57>;
    compatible = "nuvoton,npct650", "nuvoton,npct601";
    linux,sml-base = <0x7f 0xfd450000>;
    linux,sml-size = <0x10000>;
    status = "okay";
    phandle = <0x10000017>;
    linux,phandle = <0x10000017>;
};
```

4.3.13 Virtual Accelerator Switchboard (VAS)

VAS is present in P9 or later processors. In P9, each chip has one instance of VAS. Each instance of VAS is represented as a "platform device" i.e as a node in root of the device tree:

```
/vas@<vas_addr>
```

with unique VAS address which also represents the Hypervisor window context address for the instance of VAS.

Each VAS node contains:

```
compatible: "ibm,power9-vas", "ibm,vas"

ibm,chip-id: Chip-id of the chip containing this instance of VAS.

ibm,vas-id: unique identifier for each instance of VAS in the system.

reg: contains 8 64-bit fields.

      Fields [0] and [1] represent the Hypervisor window context BAR
      (start and length). Fields [2] and [3] represent the OS/User
      window context BAR (start and length). Fields [4] and [5]
      contain the start and length of paste power bus address region
      for this chip. Fields [6] and [7] represent the bit field (start
      bit and number of bits) where the window id of the window should
      be encoded when computing the paste address for the window.
```

4.3.14 VPD (Vital Product Data)

VPD provides the information about the FRUs (Field Replaceable Unit) present in the system and each vpd node in the device tree represents a FRU. These properties are passed to skiboot in the form of HDAT structure. Skiboot parses these structures and adds respective nodes in the device tree. These properties are available in all system except POWER8 BMC based system.

```
vpd {                          /* VPD root node */
  fru-name@rsrc-id {           /* Node name formatted as such */
  ibm,vpd = <                  /* VPD data binary blob */ >;
    ccin = "524D";             /* Customer Card Identification Number */
    fru-type = [ 41 56 ];  /* FRU type label (2 bytes ASCII character) */
    fru-number     =           "FRU stocking part number";
    ibm,loc-code   =           "Location code";
    part-number    =           "ABC123456";
    serial-number =            "ABC123456";
    ibm,chip-id    = <0x0>; /* If part is a chip, Processor Id */
    size = "0032768";        /* DIMM size (applicable for DIMM VPD only) */
    ibm,memory-bus-frequency = <0x0 0x0>; /* DIMM frequency (applicable for DIMM VPD␣
↪only) */
    vendor = <vendor name> /* Vendor name */
    build-date = <build date> /* Manufacturing build date (applicate for P9 BMC␣
↪systems only) */
  };
};
```

The VPD tree in the device tree depicts the hierarchial structure of the FRUs having parent-child relationship.

```
root-node-vpd@a000
   |-- enclosure@1e00
   |    |-- air-mover@3a00
   |    |-- air-mover@3a01
   |    |-- backplane@800
   |    |    |-- anchor-card@500
   |    |    |-- backplane-extender@900
   |    |    |    |-- serial-connector@2a00
   |    |    |    |-- usb-connector@2900
   |    |    |    `-- usb-connector@2901
   |    |    |-- ethernet-connector@2800
```

(continues on next page)

(continued from previous page)

```
|   |   |-- ethernet-connector@2801
|   |   |-- ms-dimm@d002
|   |   |-- ms-dimm@d003
|   |   |-- processor@1000
|   |   |-- processor@1001
|   |   |-- usb-connector@2902
|   |   |-- usb-connector@2903
|   |   |-- usb-connector@2904
|   |   `-- usb-connector@2905
|   |-- dasd-backplane@2400
|   |-- dasd-backplane@2401
|   |-- power-supply@3103
|   `-- service-processor@200
|-- enclosure-fault-led@a300
|-- enclosure-led@a200
|-- root-node-vpd@a001
`-- system-vpd@1c00
```

Example vpd node:

```
anchor-card@500 {
        ccin = "52FE";
        fru-number = "00E2147";
        description = "System Anchor Card - IBM Power 824";
        ibm,loc-code = "U78C9.001.WZS007X-P1-C13";
        serial-number = "YL10113BJ001";
        ibm,vpd = <0x84cc0052 0x54045649 0x4e494452 0x10414e43 0x484f5220 0x20202020␣
↪0x20202020 0x20434501 0x31565a02 0x3031464e 0x7303045 0x32313437 0x504e0730␣
↪0x30453231 0x3438534e 0xc594c31 0x30313133 0x424a3030 0x31434304 0x35324645␣
↪0x50520881 0x300000 0x48 0x45043030 0x31304354 0x440b400 0x485702 0x14233 0x6000000␣
↪0x142 0x34010042 0x370c0000 0x0 0x0 0x4239 0x3c435333 0x22071917 0xd1569c53␣
↪0x50973c87 0x71f9c40 0x1d4d3142 0x985e80f1 0x5cb3614d 0x32a902cb 0xd9d714ab␣
↪0x164d3322 0xdda4f986 0x5a618f4d 0x340b157c 0x2cac0a94 0x6504603 0x78 0x0>;
        fru-type = [41 56];
        part-number = "00E2148";
        phandle = <0x8d>;
        linux,phandle = <0x8d>;
};
```

4.3.15 ibm,powerpc-cpu-features Binding

This device tree binding describes CPU features available to software, with enablement, privilege, and compatibility metadata.

More general description of design and implementation of this binding is found in design.txt, which also points to documentation of specific features.

/cpus/ibm,powerpc-cpu-features node binding

Node: ibm,powerpc-cpu-features

Description: Container of CPU feature nodes.

The node name must be “ibm,powerpc-cpu-features”.

It is implemented as a child of the node “/cpus”, but this must not be assumed by parsers.

The node is optional but should be provided by new OPAL firmware.

Properties:

- device_type

  Usage: required

  Value type: string

  Definition: "cpu-features"

- compatible

  Usage: required

  Value type: string

  Definition: "ibm,powerpc-cpu-features"

  This compatibility refers to backwards compatibility of the overall design with parsers that behave according to these guidelines. This can be extended in a backward compatible manner which would not warrant a revision of the compatible property.

- isa

  Usage: required

  Value type: <u32>

  Definition:

  isa that the CPU is currently running in. This provides instruction set compatibility, less the individual feature nodes. For example, an ISA v3.0 implementation that lacks the "transactional-memory" cpufeature node should not use transactional memory facilities.

  Value corresponds to the "Power ISA Version" multiplied by 1000. For example, <3000> corresponds to Version 3.0, <2070> to Version 2.07. The minor digit is available for revisions.

/cpus/ibm,powerpc-cpu-features/example-feature node bindings

Each child node of cpu-features represents a CPU feature / capability.

Node: A string describing an architected CPU feature, e.g., "floating-point".

Description: A feature or capability supported by the CPUs.

The name of the node is a human readable string that forms the interface used to describe features to software. Features are currently documented in the code where they are implemented in skiboot/core/cpufeatures.c

Presence of the node indicates the feature is available.

Properties:

- isa

  Usage: required

  Value type: <u32>

  Definition:

  First level of the Power ISA that the feature appears in. Software should filter out features when constraining the environment to a particular ISA version.

  Value is defined similarly to /cpus/features/isa

- usable-privilege

  Usage: required

  Value type: <u32> bit mask

  Definition: Bit numbers are LSB0

    bit 0: PR (problem state / user mode)

    bit 1: OS (privileged state)

    bit 2: HV (hypervisor state)

    All other bits reserved and should be zero.

  This property describes the privilege levels and/or software components that can use the feature.

  If bit 0 is set, then the hwcap-bit-nr property will exist.

- hv-support

  Usage: optional

  Value type: <u32> bit mask

  Definition: Bit numbers are LSB0

    bit 0: HFSCR

    All other bits reserved and should be zero.

  This property describes the HV privilege support required to enable the feature to lesser privilege levels. If the property does not exist then no support is required.

  If no bits are set, the hypervisor must have explicit/custom support for this feature.

  If the HFSCR bit is set, then the hfscr-bit-nr property will exist and the feature may be enabled by setting this bit in the HFSCR register.

- os-support

  Usage: optional

  Value type: <u32> bit mask

  Definition: Bit numbers are LSB0

    bit 0: FSCR

    All other bits reserved and should be zero.

  This property describes the OS privilege support required to enable the feature to lesser privilege levels. If the property does not exist then no support is required.

  If no bits are set, the operating system must have explicit/custom support for this feature.

  If the FSCR bit is set, then the fscr-bit-nr property will exist and the feature may be enabled by setting this bit in the FSCR register.

- hfscr-bit-nr

  Usage: optional

  Value type: <u32>

  Definition: HFSCR bit position (LSB0)

This property exists when the hv-support property HFSCR bit is set. This property describes the bit number in the HFSCR register that the hypervisor must set in order to enable this feature.

This property also exists if an HFSCR bit corresponds with this feature. This makes CPU feature parsing slightly simpler.

- fscr-bit-nr

  Usage: optional

  Value type: <u32>

  Definition: FSCR bit position (LSB0)

  This property exists when the os-support property FSCR bit is set. This property describes the bit number in the FSCR register that the operating system must set in order to enable this feature.

  This property also exists if an FSCR bit corresponds with this feature. This makes CPU feature parsing slightly simpler.

- hwcap-bit-nr

  Usage: optional

  Value type: <u32>

  Definition: Linux ELF AUX vector bit position (LSB0)

  This property may exist when the usable-privilege property value has PR bit set. This property describes the bit number that should be set in the ELF AUX hardware capability vectors in order to advertise this feature to userspace. Bits 0-31 correspond to bits 0-31 in AT_HWCAP vector. Bits 32-63 correspond to 0-31 in AT_HWCAP2 vector, and so on. Missing AT_HWCAPx vectors implies that the feature is not enabled or can not be advertised. Operating systems may provide a number of unassigned hardware capability bits to allow for new features to be advertised.

  Some properties representing features created before this binding are advertised to userspace without a one-to-one hwcap bit number may not specify this bit. Operating system will handle those bits specifically. All new features usable by userspace will have a hwcap-bit-nr property.

- dependencies

  Usage: optional

  Value type: <prop-encoded-array>

  Definition:

  If this property exists then it is a list of phandles to cpu feature nodes that must be enabled for this feature to be enabled.

- Custom properties of the feature

  Usage: optional

  Definition:

  Particular features may define their own properties.

Example

```
/cpus/ibm,powerpc-cpu-features {
        device_type = "ibm,powerpc-cpu-features";
```

(continues on next page)

(continued from previous page)

```
        isa = <3020>;

        darn {
                isa = <3000>;
                usable-privilege = <1 | 2 | 4>;
                hwcap-bit-nr = <xx>;
        };

        scv {
                isa = <3000>;
                usable-privilege = <1 | 2>;
                os-support = <0>;
                hwcap-bit-nr = <xx>;
        };

        stop {
                isa = <3000>;
                usable-privilege = <2 | 4>;
                hv-support = <0>;
                os-support = <0>;
        };

        vsx2 (hypothetical) {
                isa = <3010>;
                usable-privilege = <1 | 2 | 4>;
                hv-support = <0>;
                os-support = <0>;
                hwcap-bit-nr = <xx>;
        };

        vsx2-newinsns {
                isa = <3020>;
                usable-privilege = <1 | 2 | 4>;
                os-support = <1>;
                fscr-bit-nr = <xx>;
                hwcap-bit-nr = <xx>;
                dependencies = <&vsx2>;
        };

};
```

4.3.16 ibm,powerpc-cpu-features Design

The OPAL / skiboot code is the canonical location for this specification. All definitions of features, constant, bit positions, etc. must be documented here before being deployed in Linux. This is not presently part of LoPAPR.

Interfaces

This specification describes the ibm,powerpc-cpu-features binding (the formal definition of binding can be found in binding.txt in this directory).

This specification also involves the Linux ELF AUXV AT_HWCAP and AT_HWCAP2 interfaces for PPC_FEATURE* bits. Allocation of new AT_HWCAP bits should be done in coordination with OPAL / skiboot, Linux, and glibc projects.

The binding is passed to the hypervisor by firmware. The hypervisor may build a subset with unsupported/disabled features and hypervisor specifics removed, and pass that to a guest OS. The OS may advertise features to userspace.

Background

The cpu-features binding (subsequently "cpu-features") aims to provide an extensible metadata and protocol between different levels of system software (firmware, hypervisor, OS/guest, userspace) to advertise the CPU features available on the system. With each level able to shape the features available to the next.

The binding specifies features common to all CPUs in the system. Heterogeneous CPU features are not supported at present (such could be added by providing additional cpu-features nodes and linking those to particular CPUs with additional features).

There is no strict definition for what a CPU feature must be, but an architectural behaviour or performance characteristic (or group of related behaviours). They must be documented in skiboot/core/cpufeatures.c sufficiently precisely. More guidelines for feature definitions below.

cpu-features is intended to provide fine grained control of CPU features at all levels of the stack (firmware, hypervisor, OS, userspace), with the ability for new CPU features to be used by some components without all components being upgraded (e.g., a new floating point instruction could be used by userspace math library without upgrading kernel and hypervisor).

Overview

The cpu-features node is created by firmware and passed to the hypervisor. The hypervisor may create cpu-features node to be passed to guest, based on the features that have been enabled, and policy decisions. Hypervisor specific features, and hypervisor bits and properties should not be advertised to guests. Guest OS may advertise features to userspace using another method (e.g., using AUXV vectors, userspace typically does not parse DT).

When the cpu-features node is present, ibm,pa-features and individual feature properties (e.g., "ibm,vsx"), and cpu-version under the "cpu" compatible nodes can be ignored by the consumer. For compatibility, the provider must continue to provide those older properties and the consumer must not assume cpu-features exists.

When this node exists, software may assume a base feature set which is ISA v2.07B (BookS) minus the explicit features listed in core/cpufeatures.c entries in this source tree.

Each feature is advertised as a node underneath the cpu-features node, named with a human-readable string name that uniquely identifies specification of that capability.

A feature node has a number of metadata properties describing privilege levels a feature may be used (HV, OS, PR/user), and information about how it is to be enabled and advertised to lesser privilege levels. Enabling means to make it available at a lesser privilege level, (how to enable a given feature for this privilege level is implicit: if the software know how to use a feature, it also knows how to enable it).

Feature node properties:

- "isa", the Power ISA version where this feature first became available. In case of an implementation specific feature
- "usable-privilege", a bitmask (HV, OS, PR/user) specifying which privilege levels this feature may be used in.
- "hv-support", a bitmask. If this exists, the hypervisor must do some work to enable support for lesser privilege levels. Bits can be set in this mask to specify prescription/recipes to enable the feature without custom code. If no bits are set, no recipe exists and custom code must be used. HFSCR register enable bit is the only such recipe currently.
- "os-support", similar to hv-support. FSCR recipe.
- Features may have additional properties associated, must be documented with the feature.

- Recipes may have additional properties associated. HFSCR recipe has hfscr-bit-nr, and FSCR recipe has fscr-bit-nr.
- "dependencies" array of phandles. If this exists, it links to the features that must be enabled in order for this feature to be enabled.
- "hwcap-bit-nr" if it exists provides a Linux ELF AUXV HWCAP bit number that can be used to advertise this feature to userspace.

Together, these compatibility, support, and dependencies properties allow unknown features to be enabled and advertised to lesser privilege levels (when possible).

All bits not defined in usable, support masks must be 0, and should be ignored by consumers. This allows extensibility to add new privilege levels and new recipes. Unknown properties should also be ignored. This allows extensibility for additional methods and metadata for enablement and advertisement.

The policy for selecting and configuring which features to advertise and use is left for implementations.

Guidelines for defining features

As a rough guide, features should be based on functional groups of changes to the ISA, or related performance characteristics.

Grouping should be made by one or a combination of those that:

- Share common enablement requirements (e.g., share particular registers or firmware setup requirements).
- Share common usage patterns (e..g, likely to be used together).
- Are implemented with a particular new hardware unit.
- Are optional in the ISA.

Granularity can be debated, but fine grained and encompassing is generally preferable. For example, memory management unit may be considered fundamental, but the MMU in POWER9 is very different and in many ways incompatible from that in POWER8 even in hash mode.

For example, "POWER9" would be too general, but a new feature for every instruction would be too specific. The "summary of changes" preface in Power ISA specification is a good starting point to give a guideline for granularity of the architected features.

New features that offer additional or incompatible functionality beyond an existing feature may contain an ISA version postfix.

Implementation specific behaviour should contain a CPU type postfix. E.g., "machine-check-power9" gives exact MCE properties. If a future CPU has the same MCE architecture, it should define the same property. If it has a backward-compatible superset, it could additionally define "machine-check-newcpu".

Features should be "positive" as much as possible. That is, the presence of a feature should indicate the presence of an additional CPU feature (e.g., a new instruction or register). This requires some anticipation and foresight for defining CPU features. "Negative" features may be unavoidable in some cases.

4.4 OPAL API Documentation

The OPAL API is the interface between an Operating System and OPAL.

Name	API Token ID	Introduced
OPAL_TEST	0	v1.0 (Initial Release)

Table 1 – continued from previous pa

OPAL_CONSOLE_WRITE	1	v1.0 (Initial Release)
OPAL_CONSOLE_READ	2	v1.0 (Initial Release)
OPAL_RTC_READ	3	v1.0 (Initial Release)
OPAL_RTC_WRITE	4	v1.0 (Initial Release)
OPAL_CEC_POWER_DOWN	5	v1.0 (Initial Release)
OPAL_CEC_REBOOT	6	v1.0 (Initial Release)
OPAL_READ_NVRAM	7	v1.0 (Initial Release)
OPAL_WRITE_NVRAM	8	v1.0 (Initial Release)
OPAL_HANDLE_INTERRUPT	9	v1.0 (Initial Release)
OPAL_POLL_EVENTS	10	v1.0 (Initial Release)
OPAL_PCI_SET_HUB_TCE_MEMORY	11	N/A Present only on internal systems.
OPAL_PCI_SET_PHB_TCE_MEMORY	12	N/A Present only on internal systems.
OPAL_PCI_CONFIG_READ_BYTE	13	v1.0 (Initial Release)
OPAL_PCI_CONFIG_READ_HALF_WORD	14	v1.0 (Initial Release)
OPAL_PCI_CONFIG_READ_WORD	15	v1.0 (Initial Release)
OPAL_PCI_CONFIG_WRITE_BYTE	16	v1.0 (Initial Release)
OPAL_PCI_CONFIG_WRITE_HALF_WORD	17	v1.0 (Initial Release)
OPAL_PCI_CONFIG_WRITE_WORD	18	v1.0 (Initial Release)
OPAL_SET_XIVE	19	v1.0 (Initial Release)
OPAL_GET_XIVE	20	v1.0 (Initial Release)
OPAL_GET_COMPLETION_TOKEN_STATUS	21	Never
OPAL_REGISTER_OPAL_EXCEPTION_HANDLER	22	v1.0 (Initial Release)
OPAL_PCI_EEH_FREEZE_STATUS	23	v1.0 (Initial Release)
OPAL_PCI_SHPC	24	Never
OPAL_CONSOLE_WRITE_BUFFER_SPACE	25	v1.0 (Initial Release)
OPAL_PCI_EEH_FREEZE_CLEAR	26	v1.0 (Initial Release)
OPAL_PCI_PHB_MMIO_ENABLE	27	v1.0 (Initial Release)
OPAL_PCI_SET_PHB_MEM_WINDOW	28	v1.0 (Initial Release)
OPAL_PCI_MAP_PE_MMIO_WINDOW	29	v1.0 (Initial Release)
OPAL_PCI_SET_PHB_TABLE_MEMORY	30	Never
OPAL_PCI_SET_PE	31	v1.0 (Initial Release)
OPAL_PCI_SET_PELTV	32	v1.0 (Initial Release)
OPAL_PCI_SET_MVE	33	v1.0 (Initial Release)
OPAL_PCI_SET_MVE_ENABLE	34	v1.0 (Initial Release)
OPAL_PCI_GET_XIVE_REISSUE	35	Never
OPAL_PCI_SET_XIVE_REISSUE	36	Never
OPAL_PCI_SET_XIVE_PE	37	v1.0 (Initial Release)
OPAL_GET_XIVE_SOURCE	38	v1.0 (Initial Release)
OPAL_GET_MSI_32	39	v1.0 (Initial Release)
OPAL_GET_MSI_64	40	v1.0 (Initial Release)
OPAL_START_CPU	41	v1.0 (Initial Release)
OPAL_QUERY_CPU_STATUS	42	v1.0 (Initial Release)
OPAL_WRITE_OPPANEL	43	v1.0 (Initial Release)
OPAL_PCI_MAP_PE_DMA_WINDOW	44	v1.0 (Initial Release)
OPAL_PCI_MAP_PE_DMA_WINDOW_REAL	45	v1.0 (Initial Release)
OPAL_PCI_RESET	49	v1.0 (Initial Release)
OPAL_PCI_GET_HUB_DIAG_DATA	50	v1.0 (Initial Release)
OPAL_PCI_GET_PHB_DIAG_DATA	51	N/A
OPAL_PCI_FENCE_PHB	52	Never
OPAL_PCI_REINIT	53	v1.0 (Initial Release)

Table 1 – continued from previous pa

OPAL_PCI_MASK_PE_ERROR	54	Never
OPAL_SET_SLOT_LED_STATUS	55	Never
OPAL_GET_EPOW_STATUS	56	v1.0 (Initial Release)
OPAL_SET_SYSTEM_ATTENTION_LED	57	Never
OPAL_RESERVED1	58	Never
OPAL_RESERVED2	59	Never
OPAL_PCI_NEXT_ERROR	60	v1.0 (Initial Release)
OPAL_PCI_EEH_FREEZE_STATUS2	61	v1.0 (Initial Release)
OPAL_PCI_POLL	62	v1.0 (Initial Release)
OPAL_PCI_MSI_EOI	63	v1.0 (Initial Release)
OPAL_PCI_GET_PHB_DIAG_DATA2	64	v1.0 (Initial Release)
OPAL_XSCOM_READ	65	v1.0 (Initial Release)
OPAL_XSCOM_WRITE	66	v1.0 (Initial Release)
OPAL_LPC_READ	67	v1.0 (Initial Release)
OPAL_LPC_WRITE	68	v1.0 (Initial Release)
OPAL_RETURN_CPU	69	v1.0 (Initial Release)
OPAL_REINIT_CPUS	70	v1.0 (Initial Release)
OPAL_ELOG_READ	71	v1.0 (Initial Release)
OPAL_ELOG_WRITE	72	N/A
OPAL_ELOG_ACK	73	v1.0 (Initial Release)
OPAL_ELOG_RESEND	74	v1.0 (Initial Release)
OPAL_ELOG_SIZE	75	v1.0 (Initial Release)
OPAL_FLASH_VALIDATE	76	v1.0 (Initial Release)
OPAL_FLASH_MANAGE	77	v1.0 (Initial Release)
OPAL_FLASH_UPDATE	78	v1.0 (Initial Release)
OPAL_RESYNC_TIMEBASE	79	v1.0 (Initial Release)
OPAL_CHECK_TOKEN	80	v1.0 (Initial Release)
OPAL_DUMP_INIT	81	v1.0 (Initial Release)
OPAL_DUMP_INFO	82	v1.0 (Initial Release)
OPAL_DUMP_READ	83	v1.0 (Initial Release)
OPAL_DUMP_ACK	84	v1.0 (Initial Release)
OPAL_GET_MSG	85	v1.0 (Initial Release)
OPAL_CHECK_ASYNC_COMPLETION	86	v1.0 (Initial Release)
OPAL_SYNC_HOST_REBOOT	87	v1.0 (Initial Release)
OPAL_SENSOR_READ	88	v1.0 (Initial Release)
OPAL_GET_PARAM	89	v1.0 (Initial Release)
OPAL_SET_PARAM	90	v1.0 (Initial Release)
OPAL_DUMP_RESEND	91	v1.0 (Initial Release)
OPAL_ELOG_SEND	92	Never
OPAL_PCI_SET_PHB_CAPI_MODE	93	v1.0 (Initial Release)
OPAL_DUMP_INFO2	94	v1.0 (Initial Release)
OPAL_WRITE_OPPANEL_ASYNC	95	v1.0 (Initial Release)
OPAL_PCI_ERR_INJECT	96	v1.0 (Initial Release)
OPAL_PCI_EEH_FREEZE_SET	97	v1.0 (Initial Release)
OPAL_HANDLE_HMI	98	v1.0 (Initial Release)
OPAL_CONFIG_CPU_IDLE_STATE	99	v1.0 (Initial Release)
OPAL_SLW_SET_REG	100	v1.0 (Initial Release)
OPAL_REGISTER_DUMP_REGION	101	v1.0 (Initial Release)
OPAL_UNREGISTER_DUMP_REGION	102	v1.0 (Initial Release)
OPAL_WRITE_TPO	103	v1.0 (Initial Release)

Table 1 – continued from previous pa

OPAL_READ_TPO	104	v1.0 (Initial Release)
OPAL_GET_DPO_STATUS	105	v1.0 (Initial Release)
OPAL_OLD_I2C_REQUEST	106	Introduced and deprecated in *skiboot 4.0*. Should be complet
OPAL_IPMI_SEND	107	*skiboot 4.0*
OPAL_IPMI_RECV	108	*skiboot 4.0*
OPAL_I2C_REQUEST	109	*skiboot 4.0*
OPAL_FLASH_READ	110	*skiboot 5.0*
OPAL_FLASH_WRITE	111	*skiboot 5.0*
OPAL_FLASH_ERASE	112	*skiboot 5.0*
OPAL_PRD_MSG	113	*skiboot 5.0*
OPAL_LEDS_GET_INDICATOR	114	*skiboot 5.0*
OPAL_LEDS_SET_INDICATOR	115	*skiboot 5.0*
OPAL_CEC_REBOOT2	116	*skiboot-5.1.0*
OPAL_CONSOLE_FLUSH	117	*skiboot-5.1.13*
OPAL_GET_DEVICE_TREE	118	*skiboot-5.3.0*
OPAL_PCI_GET_PRESENCE_STATE	119	*skiboot-5.3.0*
OPAL_PCI_GET_POWER_STATE	120	*skiboot-5.3.0*
OPAL_PCI_SET_POWER_STATE	121	*skiboot-5.3.0*
OPAL_INT_GET_XIRR	122	*skiboot-5.3.0*
OPAL_INT_SET_CPPR	123	*skiboot-5.3.0*
OPAL_INT_EOI	124	*skiboot-5.3.0*
OPAL_INT_SET_MFRR	125	*skiboot-5.3.0*
OPAL_PCI_TCE_KILL	126	*skiboot-5.3.0*
OPAL_NMMU_SET_PTCR	127	*skiboot-5.4.0*
OPAL_XIVE_RESET	128	*skiboot-5.5.0*
OPAL_XIVE_GET_IRQ_INFO	129	*skiboot-5.5.0*
OPAL_XIVE_GET_IRQ_CONFIG	130	*skiboot-5.5.0*
OPAL_XIVE_SET_IRQ_CONFIG	131	*skiboot-5.5.0*
OPAL_XIVE_GET_QUEUE_INFO	132	*skiboot-5.5.0*
OPAL_XIVE_SET_QUEUE_INFO	133	*skiboot-5.5.0*
OPAL_XIVE_DONATE_PAGE	134	*skiboot-5.5.0*
OPAL_XIVE_ALLOCATE_VP_BLOCK	135	*skiboot-5.5.0*
OPAL_XIVE_FREE_VP_BLOCK	136	*skiboot-5.5.0*
OPAL_XIVE_GET_VP_INFO	137	*skiboot-5.5.0*
OPAL_XIVE_SET_VP_INFO	138	*skiboot-5.5.0*
OPAL_XIVE_ALLOCATE_IRQ	139	*skiboot-5.5.0*
OPAL_XIVE_FREE_IRQ	140	*skiboot-5.5.0*
OPAL_XIVE_SYNC	141	*skiboot-5.5.0*
OPAL_XIVE_DUMP	142	*skiboot-5.5.0*
OPAL_XIVE_GET_QUEUE_STATE	143	*skiboot-6.3*
OPAL_XIVE_SET_QUEUE_STATE	144	*skiboot-6.3*
OPAL_SIGNAL_SYSTEM_RESET	145	*skiboot-5.5.0*
OPAL_NPU_INIT_CONTEXT	146	*skiboot-5.5.0*
OPAL_NPU_DESTROY_CONTEXT	147	*skiboot-5.5.0*
OPAL_NPU_MAP_LPAR	148	*skiboot-5.5.0*
OPAL_IMC_COUNTERS_INIT	149	*skiboot-5.7*
OPAL_IMC_COUNTERS_START	150	*skiboot-5.7*
OPAL_IMC_COUNTERS_STOP	151	*skiboot-5.7*
OPAL_GET_POWERCAP	152	*skiboot-5.8*
OPAL_SET_POWERCAP	153	*skiboot-5.8*

Table 1 – continued from previous pa

OPAL_GET_POWER_SHIFT_RATIO	154	*skiboot-5.8*
OPAL_SET_POWER_SHIFT_RATIO	155	*skiboot-5.8*
OPAL_SENSOR_GROUP_CLEAR	156	*skiboot-5.8*
OPAL_PCI_SET_P2P	157	*skiboot-5.8*
OPAL_QUIESCE	158	*skiboot-5.10*
OPAL_NPU_SPA_SETUP	159	*skiboot-5.11*
OPAL_NPU_SPA_CLEAR_CACHE	160	*skiboot-5.11*
OPAL_NPU_TL_SET	161	*skiboot-5.11*
OPAL_SENSOR_READ_U64	162	*skiboot-5.10*
OPAL_SENSOR_GROUP_ENABLE	163	*skiboot-5.10*
OPAL_PCI_GET_PBCQ_TUNNEL_BAR	164	*skiboot-5.11*
OPAL_PCI_SET_PBCQ_TUNNEL_BAR	165	*skiboot-5.11*
OPAL_HANDLE_HMI2	166	*skiboot-6.0*
OPAL_NX_COPROC_INIT	167	*skiboot-6.1 skiboot-6.0.5*
OPAL_NPU_SET_RELAXED_ORDER	168	*skiboot-6.2*
OPAL_NPU_GET_RELAXED_ORDER	169	*skiboot-6.2*
OPAL_XIVE_GET_VP_STATE	170	*skiboot-6.3*
OPAL_NPU_MEM_ALLOC	171	Future, likely 6.4
OPAL_NPU_MEM_RELEASE	172	Future, likely 6.4

4.4.1 OPAL_CEC_POWER_DOWN

```
#define OPAL_CEC_POWER_DOWN                    5

int64 opal_cec_power_down(uint64 request)
```

As powering down the system is likely an asynchronous operation that we have to wait for a service processor to do, *OPAL_CEC_POWER_DOWN* should be called like the example code below:

```
int rc = OPAL_BUSY;

do {
  rc = opal_cec_power_down(0);
  if (rc == OPAL_BUSY_EVENT)
    opal_poll_events(NULL);
} while (rc == OPAL_BUSY || rc == OPAL_BUSY_EVENT);

for (;;)
  opal_poll_events(NULL);
```

Arguments

uint64 request values as follows:

0 Power down normally

1 Power down immediately

This OPAL call requests OPAL to power down the system. The exact difference between a normal and immediate shutdown is platform specific.

Current Linux kernels just use power down normally (0). It is valid for a platform to only support some types of power down operations.

Return Values

OPAL_SUCCESS the power down request was successful. This may/may not result in immediate power down. An OS should spin in a loop after getting *OPAL_SUCCESS* as it is likely that there will be a delay before instructions stop being executed.

OPAL_BUSY unable to power down, try again later.

OPAL_BUSY_EVENT Unable to power down, call *OPAL_POLL_EVENTS* and try again.

OPAL_PARAMETER a parameter was incorrect

OPAL_INTERNAL_ERROR Something went wrong, and waiting and trying again is unlikely to be successful. Although, considering that in a shutdown code path, there's unlikely to be any other valid option to take, retrying is perfectly valid.

In older OPAL versions (prior to skiboot v5.9), on IBM FSP systems, this return code was returned erroneously instead of *OPAL_BUSY_EVENT* during an FSP Reset/Reload.

OPAL_UNSUPPORTED this platform does not support being powered off. Practically speaking, this should **never** be returned, but in various simulation or bring-up situations, it's plausible it is, so code should handle this gracefully.

4.4.2 OPAL_CEC_REBOOT and OPAL_CEC_REBOOT2

```
#define OPAL_CEC_REBOOT                 6
#define OPAL_CEC_REBOOT2        116
```

There are two opal calls to invoke system reboot.

OPAL_CEC_REBOOT Original reboot call for a normal reboot. It is recommended to first try *OPAL_CEC_REBOOT2* (use *OPAL_CHECK_TOKEN* first), and then, if not available, fall back to *OPAL_CEC_REBOOT*. All POWER9 systems shipped with support for *OPAL_CEC_REBOOT2*, so it is safe to exclusively call the new call if an OS only targets POWER9 and above.

OPAL_CEC_REBOOT2 Newer call for rebooting a system, supporting different types of reboots. For example, the OS may request a reboot due to a platform or OS error, which may trigger host or BMC firmware to save debugging information.

OPAL_CEC_REBOOT

Syntax:

```
int64_t opal_cec_reboot(void)
```

System reboots normally, equivalent to *OPAL_CEC_REBOOT2*. See *OPAL_CEC_REBOOT2* for details, as both OPAL calls should be called in the same way.

OPAL_CEC_REBOOT2

Syntax:

```
int64_t opal_cec_reboot2(uint32_t reboot_type, char *diag)
```

A reboot call is likely going to involve talking to a service processor to request a reboot, which can be quite a slow operation. Thus, the correct way for an OS to make an OPAL reboot call is to spin on *OPAL_POLL_EVENTS* to crank any state machine needed for the reboot until the machine reboots from underneath the OS.

For example, the below code could be part of an OS calling to do any type of reboot, and falling back to a normal reboot if that type is not supported.

```
int rc;
int reboot_type = OPAL_REBOOT_NORMAL;

do {
  if (opal_check_token(OPAL_CEC_REBOOT2) == 0) {
    rc = opal_cec_reboot2(reboot_type, NULL);
  } else {
    rc = opal_cec_reboot();
  }
  if (rc == OPAL_UNSUPPORTED) {
    printf("Falling back to normal reboot\n");
    reboot_type = OPAL_REBOOT_NORMAL;
    rc = OPAL_BUSY;
  }
  opal_poll_events(NULL);
} while (rc == OPAL_BUSY || rc == OPAL_BUSY_EVENT);

for (;;)
  opal_poll_events(NULL);
```

Input parameters

`reboot_type` Type of reboot. (see below)

`diag` Null-terminated string.

Depending on reboot type, this call will carry out additional steps before triggering a reboot.

Return Codes

OPAL_SUCCESS The system will soon reboot. The OS should loop on *OPAL_POLL_EVENTS* in case there's any work for OPAL to do.

OPAL_BUSY* or *OPAL_BUSY_EVENT OPAL is currently busy and can't issue a reboot, call *OPAL_POLL_EVENTS* and retry reboot call.

OPAL_UNSUPPORTED Unsupported reboot type (applicable to *OPAL_CEC_REBOOT2* only), retry with other reboot type.

Other error codes Keep calling reboot and hope for the best? In theory this should never happen.

Supported reboot types:

OPAL_REBOOT_NORMAL = 0 Behavior is as similar to that of opal_cec_reboot()

OPAL_REBOOT_PLATFORM_ERROR = 1 Log an error to the BMC and then trigger a system checkstop, using the information provided by 'ibm,sw-checkstop-fir' property in the device-tree. Post the checkstop trigger, OCC/BMC will collect relevant data for error analysis and trigger a reboot.

In absence of 'ibm,sw-checkstop-fir' device property, this function will return with OPAL_UNSUPPORTED and no reboot will be triggered.

OPAL_REBOOT_FULL_IPL = 2 Force a full IPL reboot rather than using fast reboot.

On platforms that don't support fast reboot, this is equivalent to a normal reboot.

Unsupported Reboot type For unsupported reboot type, this function will return with OPAL_UNSUPPORTED and no reboot will be triggered.

Debugging

This is **not** ABI and may change or be removed at any time.

You can change if the software checkstop trigger is used or not by an NVRAM variable:

```
nvram -p ibm,skiboot --update-config opal-sw-xstop=enable
nvram -p ibm,skiboot --update-config opal-sw-xstop=disable
```

4.4.3 OPAL_CHECK_ASYNC_COMPLETION

OPAL_CHECK_ASYNC_COMPLETION checks if an async OPAL pending message was completed. (see *OPAL_MESSAGE*).

```
#define OPAL_CHECK_ASYNC_COMPLETION          86

int64_t opal_check_completion(uint64_t *buffer, uint64_t size, uint64_t token);
```

Parameters:

buffer buffer to copy message into

size sizeof buffer to copy message into

token async message token

Currently unused by Linux, but it is used by FreeBSD.

Return values

OPAL_PARAMETER buffer parameter is an invalid pointer (NULL or > top of RAM).

OPAL_SUCCESS message successfully copied to buffer.

OPAL_BUSY message is still pending and should be re-checked later.

4.4.4 OPAL_CHECK_TOKEN

```
#define OPAL_CHECK_TOKEN                     80

int64_t opal_check_token(uint64_t token);
```

This OPAL call allows the host OS to determine if a particular OPAL call is present on a system. This allows for simple compatibility between OPAL versions and different OPAL implementations/platforms.

One parameter is accepted: the OPAL token number.

OPAL_CHECK_TOKEN will return:

```
enum OpalCheckTokenStatus {
  OPAL_TOKEN_ABSENT = 0,
  OPAL_TOKEN_PRESENT = 1
};
```

indicating the presence/absence of the particular OPAL_CALL.

OPAL_CHECK_TOKEN is REQUIRED to be implemented by a conformant OPAL implementation.

For skiboot, only positively ancient internal-to-IBM versions were missing OPAL_CHECK_TOKEN. In this case, OPAL_PARAMETER would be returned. There is no reason for a host OS to support this behaviour.

4.4.5 Code Update on FSP based machine

There are three OPAL calls for code update. These are currently only implemented on FSP based machines.

OPAL_FLASH_VALIDATE

```
#define OPAL_FLASH_VALIDATE  76

int64_t fsp_opal_validate_flash(uint64_t buffer, uint32_t *size, uint32_t *result);
```

Validate new image is valid for this platform or not. We do below validation in OPAL:

- We do below sys parameters validation to confirm inband update is allowed. - Platform is managed by HMC or not?. - Code update policy (inband code update allowed?).
- We parse candidate image header (first 4k bytes) to perform below validations. - Image magic number. - Image version to confirm image is valid for this platform.

Input

buffer First 4k bytes of new image

size Input buffer size

Output

buffer Output result (current and new image version details)

size Output buffer size

result Token to identify what will happen if update is attempted See hw/fsp/fsp-codeupdate.h for token values.

Return value

Validation status

OPAL_FLASH_MANAGE

```
#define OPAL_FLASH_MANAGE     77

int64_t fsp_opal_manage_flash(uint8_t op);
```

Commit/Reject image.

- We can commit new image (T -> P), if system is running with T side image.
- We can reject T side image, if system is running with P side image.

Note: If a platform is running from a T side image when an update is to be applied, then the platform may automatically commit the current T side image to the P side to allow the new image to be updated to the temporary image area.

Input

op Operation (1 : Commit /0 : Reject)

Return value Commit operation status (0 : Success)

OPAL_FLASH_UPDATE

```
#define OPAL_FLASH_UPDATE    78

int64_t fsp_opal_update_flash(struct opal_sg_list *list);
```

Update new image. It only sets the flag, actual update happens during system reboot/shutdown.

Host splits FW image to scatter/gather list and sends it to OPAL. OPAL parse the image to get indivisual LID and passes it to FSP via MBOX command.

FW update flow :

- **if (running side == T)** Swap P & T side
- Start code update
- Delete T side LIDs
- Write LIDs
- Code update complete
- Deep IPL

Input

list Real address of image scatter/gather list of the FW image

Return value: Update operation status (0: update requested)

4.4.6 OPAL_CONFIG_CPU_IDLE_STATE

```
#define OPAL_CONFIG_CPU_IDLE_STATE          99

/*
 * Setup and cleanup method for fast-sleep workarounds
 * state = 1 fast-sleep
 * enter = 1 Enter state
 * exit  = 0 Exit state
 */

#define OPAL_PM_SLEEP_ENABLED_ER1    0x00080000 /* with workaround */

int64_t opal_config_cpu_idle_state(uint64_t state, uint64_t enter);
```

If the *OPAL_PM_SLEEP_ENABLED_ER1* bit is set on a stop state, then this OPAL call needs to be made upon entry and exit of stop state. This is currently needed for the *fastsleep_* idle state, present on POWER8 systems.

Returns

OPAL_SUCCESS Applied workaround

OPAL_PARAMETER Invalid state or enter/exit.

4.4.7 OPAL Console calls

There are four OPAL calls relating to the OPAL console:

Name	API Token ID	Introduced	Required as of	Notes
OPAL_CONSOLE_WRITE	1	v1.0 (Initial Release)	POWER8	
OPAL_CONSOLE_READ	2	v1.0 (Initial Release)	POWER8	
OPAL_CONSOLE_WRITE_BUFFER_SPACE	25	v1.0 (Initial Release)	POWER8	
OPAL_CONSOLE_FLUSH	117	*skiboot-5.1.13*	POWER9	

The OPAL console calls can support multiple consoles. Each console MUST be represented in the device tree.

A conforming implementation SHOULD have at least one console. It is valid for it to simply be an in-memory buffer and only support writing.

[TODO: details on device tree specs for console]

OPAL_CONSOLE_WRITE

Parameters:

```
int64_t term_number
int64_t *length,
const uint8_t *buffer
```

Returns:

- *OPAL_SUCCESS*
- *OPAL_PARAMETER* on invalid term_number
- *OPAL_CLOSED* if console device closed

- *OPAL_BUSY_EVENT* if unable to write any of buffer

`term_number` is the terminal number as represented in the device tree. `length` is a pointer to the length of buffer.

A conforming implementation SHOULD try to NOT do partial writes, although partial writes and not writing anything are valid.

OPAL_CONSOLE_WRITE_BUFFER_SPACE

Parameters:

```
int64_t term_number
int64_t *length
```

Returns:

- *OPAL_SUCCESS*
- *OPAL_PARAMETER* on invalid term_number

Returns the available buffer length for OPAL_CONSOLE_WRITE in `length`. This call can be used to help work out if there is sufficient buffer space to write your full message to the console with OPAL_CONSOLE_WRITE.

OPAL_CONSOLE_READ

Parameters:

```
int64_t term_number
int64_t *length
uint8_t *buffer
```

Returns:

- *OPAL_SUCCESS*
- *OPAL_PARAMETER* on invalid term_number
- *OPAL_CLOSED*

Use *OPAL_POLL_EVENTS* for how to determine

OPAL_CONSOLE_FLUSH

Parameters:

```
int64_t term_number
```

Returns:

- *OPAL_SUCCESS*
- *OPAL_UNSUPPORTED* if the console does not implement a flush call
- *OPAL_PARAMETER* on invalid term_number
- *OPAL_PARTIAL* if more to flush, call again
- *OPAL_BUSY* if nothing was flushed this call

4.4.8 OPAL Dumps

```
#define OPAL_REGISTER_DUMP_REGION              101
#define OPAL_UNREGISTER_DUMP_REGION            102

int64_t opal_register_dump_region(uint32_t id, uint64_t addr, uint64_t size);
int64_t opal_unregister_dump_region(uint32_t id);
```

In the event of crashes, some service processors and firmware support gathering a limited amount of memory from a limited number of memory regions to save into a debug dump that can be useful for firmware and operating system developers in diagnosing problems. Typically, firmware and kernel log buffers are useful to save in a dump.

An OS can register a memory region with *OPAL_REGISTER_DUMP_REGION* and should, when the region is no longer valid (e.g. when kexec()ing), it should unregister the region with *OPAL_UNREGISTER_DUMP_REGION*.

An OS will be made aware of a dump being available through *OPAL_EVENT_DUMP_AVAIL = 0x400* being set from a *OPAL_POLL_EVENTS* call.

Retreiving dumps from a service processor can also be performed using the *OPAL_DUMP_INFO*, *OPAL_DUMP_INFO2*, *OPAL_DUMP_READ*, and *OPAL_DUMP_RESEND* calls. Dumps are identified by a `uint32_t`, and once a dump is acknowledged, this ID can be re-used.

Dumps can also be initiated by OPAL through the *OPAL_DUMP_INIT* call, which will request that the service processor performs the requested type of dump.

A call to *OPAL_DUMP_ACK* indicates to the service processor that we have retreived/acknowledged the dump and the service processor can free up the storage used by it.

```
#define OPAL_DUMP_INIT                                 81
#define OPAL_DUMP_INFO                                 82
#define OPAL_DUMP_READ                                 83
#define OPAL_DUMP_ACK                                  84
#define OPAL_DUMP_RESEND                               91
#define OPAL_DUMP_INFO2                                94

int64_t opal_dump_init(uint8_t dump_type);
int64_t opal_dump_info(uint32_t *dump_id, uint32_t *dump_size);
int64_t opal_dump_read(uint32_t dump_id, struct opal_sg_list *list);
int64_t opal_dump_ack(uint32_t dump_id);
int64_t opal_dump_resend_notification(void);
int64_t opal_dump_info2(uint32_t *dump_id, uint32_t *dump_size, uint32_t *dump_type);
```

OPAL_REGISTER_DUMP_REGION

```
#define OPAL_REGISTER_DUMP_REGION              101

int64_t opal_register_dump_region(uint32_t id, uint64_t addr, uint64_t size);
```

This call is used to register regions of memory for a service processor to capture when the host crashes.

e.g. if an assert is hit in OPAL, a service processor will copy the region of memory into some kind of crash dump for further analysis.

This is an OPTIONAL feature that may be unsupported, the host OS should use an *OPAL_CHECK_TOKEN* call to find out if *OPAL_REGISTER_DUMP_REGION* is supported.

OPAL_REGISTER_DUMP_REGION accepts 3 parameters:

- region ID

- address
- length

There is a range of region IDs that can be used by the host OS. A host OS should start from OPAL_DUMP_REGION_HOST_END and work down if it wants to add a not well defined region to dump. Currently the only well defined region is for the host OS log buffer (e.g. dmesg on linux).

```
/*
 * Dump region ID range usable by the OS
 */
#define OPAL_DUMP_REGION_HOST_START           0x80
#define OPAL_DUMP_REGION_LOG_BUF              0x80
#define OPAL_DUMP_REGION_HOST_END             0xFF
```

OPAL_REGISTER_DUMP_REGION will return *OPAL_UNSUPPORTED* if the call is present but the system doesn't support registering regions to be dumped.

In the event of being passed an invalid region ID, *OPAL_REGISTER_DUMP_REGION* will return *OPAL_PARAMETER*.

Systems likely have a limit as to how many regions they can support being dumped. If this limit is reached, *OPAL_REGISTER_DUMP_REGION* will return *OPAL_INTERNAL_ERROR*.

BUGS

Some skiboot versions incorrectly returned *OPAL_SUCCESS* in the case of *OPAL_REGISTER_DUMP_REGION* being supported on a platform (so the call was present) but the call being unsupported for some reason (e.g. on an IBM POWER7 machine).

See also: *OPAL_UNREGISTER_DUMP_REGION*

OPAL_UNREGISTER_DUMP_REGION

```
#define OPAL_UNREGISTER_DUMP_REGION          102

int64_t opal_unregister_dump_region(uint32_t id);
```

While *OPAL_REGISTER_DUMP_REGION* registers a region, *OPAL_UNREGISTER_DUMP_REGION* will unregister a region by region ID.

OPAL_UNREGISTER_DUMP_REGION takes one argument: the region ID.

A host OS should check *OPAL_UNREGISTER_DUMP_REGION* is supported through a call to *OPAL_CHECK_TOKEN*.

If *OPAL_UNREGISTER_DUMP_REGION* is called on a system where the call is present but unsupported, it will return *OPAL_UNSUPPORTED*.

BUGS

Some skiboot versions incorrectly returned *OPAL_SUCCESS* in the case of *OPAL_UNREGISTER_DUMP_REGION* being supported on a platform (so the call was present) but the call being unsupported for some reason (e.g. on an IBM POWER7 machine).

OPAL_DUMP_INIT

```
#define OPAL_DUMP_INIT                          81

#define DUMP_TYPE_FSP               0x01
#define DUMP_TYPE_SYS               0x02
#define DUMP_TYPE_SMA               0x03

int64_t opal_dump_init(uint8_t dump_type);
```

Ask the service processor to initiate a dump. Currently, only `DUMP_TYPE_FSP` is supported.

Currently only implemented on FSP based systems. Use *OPAL_CHECK_TOKEN* to ensure the call is valid.

Returns

OPAL_SUCCESS Dump initiated

OPAL_PARAMETER Unsupported dump type. Currently only `DUMP_TYPE_FSP` is supported and only on FSP based platforms.

OPAL_INTERNAL_ERROR Failed to ask service processor to initiated dump.

OPAL_DUMP_INFO

```
#define OPAL_DUMP_INFO                          82

int64_t opal_dump_info(uint32_t *dump_id, uint32_t *dump_size);
```

Obsolete, use *OPAL_DUMP_INFO2* instead.

No upstream Linux code ever used *OPAL_DUMP_INFO*, although early PowerKVM trees did. *OPAL_DUMP_INFO* is implemented as a wrapper around *OPAL_DUMP_INFO2*.

OPAL_DUMP_READ

```
#define OPAL_DUMP_READ                          83

int64_t opal_dump_read(uint32_t dump_id, struct opal_sg_list *list);
```

Read `dump_id` dump from the service processor into memory.

Returns

OPAL_INTERNAL_ERROR Invalid Dump ID or internal error.

OPAL_PARAMETER Insuffcient space to store dump.

OPAL_BUSY_EVENT Fetching dump, call *OPAL_POLL_EVENTS* to crank the state machine, and call *OPAL_DUMP_READ* again until neither *OPAL_BUSY_EVENT* nor *OPAL_BUSY* are returned.

OPAL_PARTIAL Only part of the dump was read.

OPAL_SUCCESS Dump successfully read.

OPAL_DUMP_ACK

```
#define OPAL_DUMP_ACK                               84

int64_t opal_dump_ack(uint32_t dump_id);
```

Acknowledge the dump to the service processor. This means the service processor can re-claim the storage space used by the dump. Effectively, this is an `unlink` style operation.

Returns

OPAL_SUCCESS Dump successfully acknowledged.

OPAL_INTERNAL_ERROR Failed to acknowledge the dump, e.g. could not communicate with service processor.

OPAL_PARAMETER Invalid dump ID.

OPAL_DUMP_RESEND

```
#define OPAL_DUMP_RESEND                            91

int64_t opal_dump_resend_notification(void);
```

Resend notification to the OS if there is a dump available. This will cause OPAL to check if a dump is available and set the *OPAL_EVENT_DUMP_AVAIL = 0x400* bit in the next *OPAL_POLL_EVENTS* call.

Returns

OPAL_SUCCESS Successfully reset *OPAL_EVENT_DUMP_AVAIL = 0x400* bit.

In future, this may return other standard OPAL error codes.

OPAL_DUMP_INFO2

```
#define OPAL_DUMP_INFO2                             94

#define DUMP_TYPE_FSP                0x01
#define DUMP_TYPE_SYS                0x02
#define DUMP_TYPE_SMA                0x03

int64_t opal_dump_info2(uint32_t *dump_id, uint32_t *dump_size, uint32_t *dump_type);
```

Retreives information about a dump, notably it's `dump_id`, size, and type. Call this after the *OPAL_EVENT_DUMP_AVAIL = 0x400* bit is set from a *OPAL_POLL_EVENTS* call. It will retreive the information on the *next* dump to be retreived and/or ACKed, even though there may be more than one dump available for retreiving.

This call replaces *OPAL_DUMP_INFO*.

Returns

OPAL_SUCCESS Information retreived.

OPAL_INTERNAL_ERROR No dump available or internal error.

4.4.9 OPAL_ELOG: Error logging

OPAL provides an abstraction to platform specific methods of storing and retrieving error logs. Some service processors may be able to store information in the Platform Error Log (PEL) format. These may be generated at runtime by the service processor or OPAL in reaction to certain events. For example, an IPL failure could be recorded in an error log, as could the reason and details of an unexpected shut-down/reboot (e.g. hard thermal limits, check-stop).

There are five OPAL calls from host to OPAL on error log:

```
#define OPAL_ELOG_READ          71
#define OPAL_ELOG_WRITE         72
#define OPAL_ELOG_ACK           73
#define OPAL_ELOG_RESEND        74
#define OPAL_ELOG_SIZE          75
```

Note: *OPAL_ELOG_WRITE* (72) Unused for now, might be supported in the future.

Not all platforms support these calls, so it's important for a host Operating System to use the *OPAL_CHECK_TOKEN* call first. If *OPAL_ELOG_READ*, *OPAL_ELOG_ACK*, *OPAL_ELOG_RESEND*, or *OPAL_ELOG_SIZE* is present, then the rest of that group is also present. The presence of *OPAL_ELOG_WRITE* must be checked separately.

TODO: we need a good explanation of the notification mechanism and in what order and *when* to call each of the OPAL APIs.

OPAL_ELOG_READ

The *OPAL_ELOG_READ* call will copy the error log identified by `id` into the `buffer` of size `size`.

`OPAL_ELOG_READ` accepts 3 parameters:

```
uint64_t *elog_buffer
uint64_t elog_size
uint64_t elog_id
```

Returns:

OPAL_WRONG_STATE When there are no error logs to read, or `OPAL_ELOG` calls are done in the wrong order.

OPAL_PARAMETER The `id` does not match the log id that is available.

OPAL_SUCCESS Error log is copied to `buffer`.

Other generic OPAL error codes may also be returned and should be treated like *OPAL_INTERNAL_ERROR*.

OPAL_ELOG_ACK

Acknowledging (ACKing) an error log tells OPAL and the service processor that the host operating system has dealt with the error log successfully. This allows OPAL and the service processor to delete the error log from their memory/storage.

OPAL_ELOG_ACK accepts 1 parameter:

```
uint64_t ack_id
```

Returns:

OPAL_INTERNAL_ERROR OPAL failed to send acknowledgement to the error log creator.

OPAL_SUCCESS Success!

Other generic OPAL error codes may also be returned, and should be treated like *OPAL_INTERNAL_ERROR*.

OPAL_ELOG_RESEND

The *OPAL_ELOG_RESEND* call will cause OPAL to resend notification to the host operating system of all outstanding error logs. This is commonly used (although doesn't have to be) in a kexec scenario.

The call registered with this token accepts no parameter and returns type is void.

OPAL_ELOG_SIZE

The *OPAL_ELOG_SIZE* call retrieves information about an error log.

Here, `type` specifies error log format. Supported types are :

```
0 -> Platform Error Log
```

OPAL_ELOG_SIZE accepts 3 parameters:

```
uint64_t *elog_id
uint64_t *elog_size
uint64_t *elog_type
```

Returns:

OPAL_WRONG_STATE There is no error log to fetch information about.

OPAL_SUCCESS Success.

Other general OPAL errors may be returned.

4.4.10 OPAL Flash calls

There are three OPAL calls for interacting with flash devices:

```
#define OPAL_FLASH_READ         110
#define OPAL_FLASH_WRITE        111
#define OPAL_FLASH_ERASE        112
```

Multiple flash devices are supported by OPAL - each of these calls takes an id parameter, which must match an ID found in the corresponding `ibm,opal/flash@n` device tree node. See *ibm,opal/flash device tree entries* for details of the device tree bindings.

All operations on the flash device must be aligned to the block size of the flash. This applies to both offset and size arguments.

This interface is asynchronous; all calls require a 'token' argument. On success, the calls will return *OPAL_ASYNC_COMPLETION*, and an opal_async_completion message will be sent (with the appropriate token argument) when the operation completes.

Note: These calls can have higher than normal latency, spending many **milliseconds** inside OPAL. This is due to the OPAL_FLASH_* calls typically being backed by flash on the other side of the LPC bus, which has a maximum transfer rate of 5MB/sec, or to/from flash attached to the ast2400/ast2500 (the typical setup for OpenPOWER systems) of only 1.75MB/sec.

All calls share the same return values:

OPAL_ASYNC_COMPLETION operation started, an async completion will be triggered with the `token` argument

OPAL_PARAMETER invalid flash id

OPAL_PARAMETER invalid size or offset (alignment, or access beyond end of device)

OPAL_BUSY flash in use

OPAL_HARDWARE error accessing flash device

OPAL_FLASH_READ

```
#define OPAL_FLASH_READ     110

int64_t opal_flash_read(uint64_t id, uint64_t offset, uint64_t buf,
                        uint64_t size, uint64_t token);
```

Reads from the specified flash id, at the specified offset, into the buffer. Will trigger an async completion with token when completed.

OPAL_FLASH_ERASE

```
#define OPAL_FLASH_ERASE     112

int64_t opal_flash_erase(uint64_t id, uint64_t offset, uint64_t size,
                         uint64_t token);
```

Erases the specified flash id, at the specified offset and size. Will trigger an async completion with token when completed.

OPAL_FLASH_WRITE

```
#define OPAL_FLASH_WRITE     111

int64_t opal_flash_write(uint64_t id, uint64_t offset, uint64_t buf,
                         uint64_t size, uint64_t token);
```

Writes buffer to the specified flash id, at the specified offset and size. The flash must be erased before being written. Will trigger an async completion with token when completed.

4.4.11 OPAL_GET_DEVICE_TREE

```
#define OPAL_GET_DEVICE_TREE                 118

int64_t opal_get_device_tree(uint32_t phandle, uint64_t buf, uint64_t len);
```

Get device sub-tree.

`uint32_t phandle` root device node phandle of the device sub-tree

`uint64_t buf` FDT blob buffer or NULL

`uint64_t len` length of the FDT blob buffer

Retrieve device sub-tree. The root node's phandle is identified by @phandle. The typical use is for the kernel to update its device tree following a change in hardware (e.g. PCI hotplug).

Return Codes

FDT blob size returned FDT blob buffer size when `buf` is NULL

OPAL_SUCCESS FDT blob is created successfully

OPAL_PARAMETER invalid argument @phandle or @len

OPAL_INTERNAL_ERROR failure creating FDT blob when calculating its size

OPAL_NO_MEM not enough room in buffer for device sub-tree

OPAL_EMPTY failure creating FDT blob

4.4.12 OPAL_GET_EPOW_STATUS

```
#define OPAL_GET_EPOW_STATUS                    56

enum OpalEpowStatus {
     OPAL_EPOW_NONE = 0,
     OPAL_EPOW_UPS = 1,
     OPAL_EPOW_OVER_AMBIENT_TEMP = 2,
     OPAL_EPOW_OVER_INTERNAL_TEMP = 3
};

/* System EPOW type */
enum OpalSysEpow {
     OPAL_SYSEPOW_POWER       = 0,     /* Power EPOW */
     OPAL_SYSEPOW_TEMP        = 1,     /* Temperature EPOW */
     OPAL_SYSEPOW_COOLING     = 2,     /* Cooling EPOW */
     OPAL_SYSEPOW_MAX         = 3,     /* Max EPOW categories */
};

/* Power EPOW */
enum OpalSysPower {
     OPAL_SYSPOWER_UPS        = 0x0001, /* System on UPS power */
     OPAL_SYSPOWER_CHNG       = 0x0002, /* System power configuration change */
     OPAL_SYSPOWER_FAIL       = 0x0004, /* System impending power failure */
     OPAL_SYSPOWER_INCL       = 0x0008, /* System incomplete power */
     };

/* Temperature EPOW */
enum OpalSysTemp {
     OPAL_SYSTEMP_AMB         = 0x0001, /* System over ambient temperature */
     OPAL_SYSTEMP_INT         = 0x0002, /* System over internal temperature */
     OPAL_SYSTEMP_HMD         = 0x0004, /* System over ambient humidity */
};
```

(continues on next page)

(continued from previous page)

```
/* Cooling EPOW */
enum OpalSysCooling {
     OPAL_SYSCOOL_INSF        = 0x0001, /* System insufficient cooling */
};

int64_t opal_get_epow_status(int16_t *out_epow, int16_t *length);
```

The *OPAL_GET_EPOW_STATUS* call gets the Environmental and Power Warnings state from OPAL. This can allow an OS to take action based on information from firmware / sensors.

On receipt of an *OPAL_MSG_EPOW* message, the OS can query the status using the *OPAL_GET_EPOW_STATUS* call. The OS allocates an array for the status bits, and passes in the length of this array. OPAL will return the maximum length it filled out. Thus, new classes can be added and backwards compatibility is maintained.

At time of writing, this call is only implemented on FSP based systems.

Returns

OPAL_SUCCESS Successfully retreived status. Note, success is returned even if only able to retreive a subset of the EPOW classes.

Other return codes may be returned in the future.

4.4.13 OPAL_GET_MSG

```
#define OPAL_GET_MSG                        85

int64_t opal_get_msg(uint64_t *buffer, uint64_t size);
```

OPAL_GET_MSG will get the next pending OPAL Message (see *OPAL_MESSAGE*).

The maximum size of an opal message is specified in the device tree passed to the host OS:

```
ibm,opal {
          opal-msg-size = <0x48>;
}
```

It is ALWAYS at least 72 bytes. In the future, OPAL may have messages larger than 72 bytes. Naturally, a HOST OS will only be able to interpret these if it correctly uses opal-msg-size. Any OPAL message > 72 bytes, a host OS may safely ignore.

A host OS *SHOULD* always supply a buffer to OPAL_GET_MSG of either 72 bytes or opal-msg-size. It MUST NOT supply a buffer of < 72 bytes.

Return values

OPAL_RESOURCE no available message.

OPAL_PARAMETER buffer is NULL or size is < 72 bytes. If buffer size < 72 bytes, the message will NOT be discarded by OPAL.

OPAL_PARTIAL If pending opal message is greater than supplied buffer. In this case the message is *DISCARDED* by OPAL. This is to keep compatibility with host Operating Systems with a hard coded opal-msg-size of 72 bytes. **NOT CURRENTLY IMPLEMENTED**. Specified so that host OS can prepare for the possible future with either a sensible error message or by gracefully ignoring such OPAL messages.

OPAL_SUCCESS message successfully copied to buffer.

4.4.14 OPAL_GET_MSI_32 and OPAL_GET_MSI_64

```
#define OPAL_GET_MSI_32                         39
#define OPAL_GET_MSI_64                         40

int64_t opal_get_msi_32(uint64_t phb_id, uint32_t mve_number,
                        uint32_t xive_num, uint8_t msi_range,
                        uint32_t *msi_address, uint32_t *message_data);

int64_t opal_get_msi_64(uint64_t phb_id, uint32_t mve_number,
                        uint32_t xive_num, uint8_t msi_range,
                        uint64_t *msi_address, uint32_t *message_data);
```

OPAL_GET_MSI_32

```
#define OPAL_GET_MSI_32                         39

int64_t opal_get_msi_32(uint64_t phb_id, uint32_t mve_number,
                        uint32_t xive_num, uint8_t msi_range,
                        uint32_t *msi_address, uint32_t *message_data);
```

See *OPAL_GET_MSI_64*.

OPAL_GET_MSI_64

```
#define OPAL_GET_MSI_64                         40

int64_t opal_get_msi_64(uint64_t phb_id, uint32_t mve_number,
                        uint32_t xive_num, uint8_t msi_range,
                        uint64_t *msi_address, uint32_t *message_data);
```

WARNING: the following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

OPAL PHBs encode MVE and XIVE specifiers in MSI DMA and message data values. The host calls these functions to determine the PHB MSI DMA address and message data to program into a PE PCIE function for a particular MVE and XIVE. The msi_address parameter returns the MSI DMA address and the msi_data parameter returns the MSI DMA message data value the PE uses to signal that interrupt.

`phb_id` The `phb_id` parameter is the value from the PHB node `ibm,opal-phbid` property.

`mve_number` The `mve_number` is the index of an MVE used to authorize this PE to this MSI. For `ibm,opal-ioda2` PHBs, the MVE number argument is ignored.

`xive_number` The `xive_number` is the index of an XIVE that corresponds to a particular DMA address and message data value this PE will signal as an MSI ro MSI-X.

`msi_range` The msi_range parameter specifies the number of MSIs associated with the in put MVE and XIVE, primarily for MSI-conventional Multiple Message Enable > 1 MSI. MSI requires consecutive MSIs per MSI address, and each MSI DMA address must be unique for any given consecutive power of 2 set of 32 message data values,. which in turn select particular PHB XIVEs. This value must be a power of 2 value in the range of 0 to 32. OPAL returns *OPAL_PARAMETER* for values outside of this range.

For MSI conventional, the MSI address and message data returned apply to a power of 2 sequential set of XIVRs starting from the xive_number for the power of 2 msi_range input argument. The message data returned represents the power of 2 aligned starting message data value of the first interrupt number in that sequential range. Valid msi_range input values are from 1 to 32. Non-power of 2 values result in a return code of *OPAL_PARAMETER*.

An msi_range value of 0 or 1 signifies that OPAL should return the message data and message address for exactly one MSI specified by the input XIVE number. For MSI conventional, the host should specify either a value of 0 or 1, for an MSI Capability MME value of 1 MSI. For MSI-X XIVRs, the host should specify a value of '1' for the msi_range argument and call this function for each MSI-X uniquely.

4.4.15 OPAL_GET_XIVE

```
#define OPAL_GET_XIVE                              20

int64_t opal_get_xive(uint32_t isn, uint16_t *server, uint8_t *priority);
```

The host calls this function to return the configuration of an interrupt source. See *OPAL_SET_XIVE* for details.

Parameters

isn The `isn` is the global interrupt number being queried

server_number the `server_number` returns the mangled server (processor) that is set to receive that interrupt.

priority the `priority` returns the current interrupt priority setting for that interrupt.

4.4.16 Hypervisor Maintenance Interrupt (HMI)

Hypervisor Maintenance Interrupt usually reports error related to processor recovery/checkstop, NX/NPU checkstop and Timer facility. Hypervisor then takes this opportunity to analyze and recover from some of these errors. Hypervisor takes assistance from OPAL layer to handle and recover from HMI. After handling HMI, OPAL layer sends the summary of error report and status of recovery action using HMI event. See ref:*opal-messages* for HMI event structure under *OPAL_MSG_HMI_EVT* section.

HMI is thread specific. The reason for HMI is available in a per thread Hypervisor Maintenance Exception Register (HMER). A Hypervisor Maintenance Exception Enable Register (HMEER) is per core. Bits from the HMER need to be enabled by the corresponding bits in the HMEER in order to cause an HMI.

Several interrupt reasons are routed in parallel to each of the thread specific copies. Each thread can only clear bits in its own HMER. OPAL handler from each thread clears the respective bit from HMER register after handling the error.

4.4.17 List of errors that causes HMI

- CPU Errors
  - Processor Core checkstop
  - Processor retry recovery
  - NX/NPU/CAPP checkstop.
- Timer facility Errors

  - ChipTOD Errors
  - ChipTOD sync check and parity errors
  - ChipTOD configuration register parity errors
  - ChiTOD topology failover
- Timebase (TB) errors
  - TB parity/residue error
  - TFMR parity and firmware control error
  - DEC/HDEC/PURR/SPURR parity errors

4.4.18 HMI handling

A core/NX/NPU checkstops are reported as malfunction alert (HMER bit 0). OPAL handler scans through Fault Isolation Register (FIR) for each core/nx/npu to detect the exact reason for checkstop and reports it back to the host alongwith the disposition.

A processor recovery is reported through HMER bits 2, 3 and 11. These are just an informational messages and no extra recovery is required.

Timer facility errors are reported through HMER bit 4. These are all recoverable errors. The exact reason for the errors are stored in Timer Facility Management Register (TFMR). Some of the Timer facility errors affects TB and some of them affects TOD. TOD is a per chip Time-Of-Day logic that holds the actual time value of the chip and communicates with every TOD in the system to achieve synchronized timer value within a system. TB is per core register (64-bit) derives its value from ChipTOD at startup and then it gets periodically incremented by STEP signal provided by the TOD. In a multi-socket system TODs are always configured as master/backup TOD under primary/secondary topology configuration respectively.

TB error generates HMI on all threads of the affected core. TB errors except DEC/HDEC/PURR/SPURR parity errors, causes TB to stop running making it invalid. As part of TB recovery, OPAL hmi handler synchronizes with all threads, clears the TB errors and then re-sync the TB with TOD value putting it back in running state.

TOD errors generates HMI on every core/thread of affected chip. The reason for TOD errors are stored in TOD ERROR register (0x40030). As part of the recovery OPAL hmi handler clears the TOD error and then requests new TOD value from another running chipTOD in the system. Sometimes, if a primary chipTOD is in error, it may need a TOD topology switch to recover from error. A TOD topology switch basically makes a backup as new active master.

4.4.19 OPAL_HANDLE_HMI

```
#define OPAL_HANDLE_HMI       98

int64_t opal_handle_hmi(void);
```

Superseded by *OPAL_HANDLE_HMI2*, meaning that *OPAL_HANDLE_HMI* should only be called if *OPAL_HANDLE_HMI2* is not available.

Since *OPAL_HANDLE_HMI2* has been available since the start of POWER9 systems being supported, if you only target POWER9 and above, you can assume the presence of *OPAL_HANDLE_HMI2*.

4.4.20 OPAL_HANDLE_HMI2

```
#define OPAL_HANDLE_HMI2      166

int64_t opal_handle_hmi2(__be64 *out_flags);
```

When OS host gets an Hypervisor Maintenance Interrupt (HMI), it must call *OPAL_HANDLE_HMI* or *OPAL_HANDLE_HMI2*. The *OPAL_HANDLE_HMI* is an old interface. *OPAL_HANDLE_HMI2* is newly introduced opal call that returns direct info to the OS. It returns a 64-bit flag mask currently set to provide info about which timer facilities were lost, and whether an event was generated. This information will help OS to take respective actions.

In case where opal hmi handler is unable to recover from TOD or TB errors, it would flag OPAL_HMI_FLAGS_TOD_TB_FAIL to indicate OS that TB is dead. This information then can be used by OS to make sure that the functions relying on TB value (e.g. udelay()) are aware of TB not ticking. This will avoid OS getting stuck or hang during its way to panic path.

Parameters

```
__be64 *out_flags;
```

Returns the 64-bit flag mask that provides info about which timer facilities were lost, and whether an event was generated.

```
/* OPAL_HANDLE_HMI2 out_flags */
enum {
     OPAL_HMI_FLAGS_TB_RESYNC         = (1ull << 0), /* Timebase has been resynced */
     OPAL_HMI_FLAGS_DEC_LOST          = (1ull << 1), /* DEC lost, needs to be␣
↪reprogrammed */
     OPAL_HMI_FLAGS_HDEC_LOST         = (1ull << 2), /* HDEC lost, needs to be␣
↪reprogrammed */
     OPAL_HMI_FLAGS_TOD_TB_FAIL       = (1ull << 3), /* TOD/TB recovery failed. */
     OPAL_HMI_FLAGS_NEW_EVENT         = (1ull << 63), /* An event has been created */
};
```

OPAL_HMI_FLAGS_TOD_TB_FAIL The Time of Day (TOD) / Timebase facility has failed. This is probably fatal for the OS, and requires the OS to be very careful to not call any function that may rely on it, usually as it heads down a *panic()* code path. This code path should be *OPAL_CEC_REBOOT2* with the OPAL_REBOOT_PLATFORM_ERROR option. Details of the failure are likely delivered as part of HMI events if *OPAL_HMI_FLAGS_NEW_EVENT* is set.

4.4.21 OPAL_HANDLE_INTERRUPT

The host OS must pass all interrupts in the *opal-interrupts* property of *ibm,opal* in the device tree to OPAL.

An example dt snippet is:

```
ibm,opal {
        opal-interrupts = <0x10 0x11 0x12 0x13 0x14 0x20010 0x20011 0x20012 0x20013␣
↪0x20014 0xffe 0xfff 0x17fe 0x17ff 0x2ffe 0x2fff 0x37fe 0x37ff 0x20ffe 0x20fff␣
↪0x217fe 0x217ff 0x22ffe 0x22fff 0x237fe 0x237ff>;
};
```

When the host OS gets any of these interrupts, it must call `OPAL_HANDLE_INTERRUPT`.

The `OPAL_HANDLE_INTERRUPT` call takes two parameters, one input and one output.

uint32_t isn the interrupt

uint64_t *outstanding_event_mask returns outstanding events for host OS to handle

The host OS should then handle any outstanding events.

See *OPAL_POLL_EVENTS* for documentation on events.

4.4.22 OPAL_I2C_REQUEST

```
#define OPAL_I2C_REQUEST                        109

/* OPAL I2C request */
struct opal_i2c_request {
             uint8_t type;
#define OPAL_I2C_RAW_READ     0
#define OPAL_I2C_RAW_WRITE    1
#define OPAL_I2C_SM_READ      2
#define OPAL_I2C_SM_WRITE     3
             uint8_t flags;
#define OPAL_I2C_ADDR_10      0x01    /* Not supported yet */
             uint8_t subaddr_sz;              /* Max 4 */
             uint8_t reserved;
             __be16 addr;                     /* 7 or 10 bit address */
             __be16 reserved2;
             __be32 subaddr;          /* Sub-address if any */
             __be32 size;                     /* Data size */
             __be64 buffer_ra;                /* Buffer real address */
};

int opal_i2c_request(uint64_t async_token, uint32_t bus_id,
                     struct opal_i2c_request *oreq);
```

Initiate I2C request using i2c master that OPAL controls.

Return Codes

Most return codes will come through as part of async completion.

OPAL_PARAMETER Invalid request pointer, or bus ID.

OPAL_UNSUPPORTED Unsupported operation. e.g. 10 bit addresses not yet supported.

OPAL_NO_MEM Not enough free memory in OPAL to initiate request.

OPAL_ASYNC_COMPLETION Operation will complete asynchronously.

OPAL_I2C_TIMEOUT I2C operation initiated successfully, but timed out.

OPAL_I2C_INVALID Invalid i2c Command.

OPAL_I2C_LBUS_PARITY I2C LBUS Parity error

OPAL_I2C_BKEND_OVERRUN I2C Backend overrun.

OPAL_I2C_BKEND_ACCESS I2C Backend Access error.

OPAL_I2C_ARBT_LOST I2C Bus Arbitration lost.

OPAL_I2C_NACK_RCVD I2C NACK received.

OPAL_I2C_STOP_ERR I2C STOP error.

OPAL_SUCCESS I2C operation completed successfully. Typically only as part of async completion.

4.4.23 OPAL_IMC_COUNTERS_INIT

OPAL call interface to initialize In-memory collection infrastructure. Call does multiple scom writes on each invocation for Core/Trace IMC initialization. And for the Nest IMC, at this point, call is a no-op and returns OPAL_SUCCESS. Incase of kexec, OS driver should first stop the engine via OPAL_IMC_COUNTER_STOP(and then free the memory if allocated, for nest memory is mmapped). Incase of kdump, OS driver should stop the engine via OPAL_IMC_COUNTER_STOP.

OPAL does sanity checks to detect unknown or unsupported IMC device type and nest units. check_imc_device_type() function removes unsupported IMC device type. disable_unavailable_units() removes unsupported nest units by the microcode. This way OPAL can lock down and advertise only supported device type and nest units.

Parameters

`uint32_t type` This parameter specifies the imc counter domain. The value can be 'OPAL_IMC_COUNTERS_NEST', 'OPAL_IMC_COUNTERS_CORE' or 'OPAL_IMC_COUNTERS_TRACE'.

`uint64_t addr` This parameter must have a non-zero value. This value must be a physical address of the core.

`uint64_t cpu_pir` This parameter specifices target cpu pir

Returns

OPAL_PARAMETER In case of unsupported `type`

OPAL_HARDWARE If any error in setting up the hardware.

OPAL_SUCCESS On succesfully initialized or even if init operation is a no-op.

4.4.24 OPAL_IMC_COUNTERS_START

OPAL call interface for starting the In-Memory Collection counters for a specified domain (NEST/CORE/TRACE).

Parameters

`uint32_t type` This parameter specifies the imc counter domain. The value can be 'OPAL_IMC_COUNTERS_NEST', 'OPAL_IMC_COUNTERS_CORE' or 'OPAL_IMC_COUNTERS_TRACE'.

`uint64_t cpu_pir` This parameter specifices target cpu pir

Returns

OPAL_PARAMETER In case of Unsupported `type`

OPAL_HARDWARE If any error in setting up the hardware.

OPAL_SUCCESS On successful execution of the operation for the given `type`.

4.4.25 OPAL_IMC_COUNTERS_STOP

OPAL call interface for stoping In-Memory Collection counters for a specified domain (NEST/CORE/TRACE). STOP should always be called after a related START. While STOP *may* run successfully without an associated START call, this is not gaurenteed.

Parameters

uint32_t type This parameter specifies the imc counter domain. The value can be 'OPAL_IMC_COUNTERS_NEST', 'OPAL_IMC_COUNTERS_CORE' or 'OPAL_IMC_COUNTERS_TRACE'

uint64_t cpu_pir This parameter specifices target cpu pir

Returns

OPAL_PARAMETER In case of Unsupported `type`

OPAL_HARDWARE If any error in setting up the hardware.

OPAL_SUCCESS On successful execution of the operation for the given `type`.

4.4.26 OPAL_INT_EOI

```
#define OPAL_INT_EOI                            124

static int64_t opal_int_eoi(uint32_t xirr);
```

Modelled on the `H_EOI` PAPR call.

This can return a positive value, which means more interrupts are queued for that CPU/priority and must be fetched as the XIVE is not guaranteed to assert the CPU external interrupt line again until the pending queue for the current priority has been emptied.

For P9 and above systems where host doesn't know about interrupt controller. An OS can instead make OPAL calls for XICS emulation.

For an OS to use this OPAL call, an `ibm,opal-intc` compatible device must exist in the device tree (see *I - Device-tree updates*). If OPAL does not create such a device, the host OS MUST NOT use this call.

4.4.27 OPAL_INT_GET_XIRR

```
#define OPAL_INT_GET_XIRR                       122

int64_t opal_int_get_xirr(uint32_t *out_xirr, bool just_poll);
```

Modelled on the PAPR call.

For P9 and above systems where host doesn't know about interrupt controller. An OS can instead make OPAL calls for XICS emulation.

For an OS to use this OPAL call, an `ibm,opal-intc` compatible device must exist in the device tree (see *I - Device-tree updates*). If OPAL does not create such a device, the host OS MUST NOT use this call.

4.4.28 OPAL_INT_SET_CPPR

```
#define      OPAL_INT_SET_CPPR                           123

static int64_t opal_int_set_cppr(uint8_t cppr);
```

Modelled on the `H_CPPR` PAPR call.

For P9 and above systems where host doesn't know about interrupt controller. An OS can instead make OPAL calls for XICS emulation.

For an OS to use this OPAL call, an `ibm,opal-intc` compatible device must exist in the device tree (see *I - Device-tree updates*). If OPAL does not create such a device, the host OS MUST NOT use this call.

4.4.29 OPAL_INT_SET_MFRR

```
#define OPAL_INT_SET_MFRR                           125

static int64_t opal_int_set_mfrr(uint32_t cpu, uint8_t mfrr);
```

Modelled on the `H_IPI` PAPR call.

For P9 and above systems where host doesn't know about interrupt controller. An OS can instead make OPAL calls for XICS emulation.

For an OS to use this OPAL call, an `ibm,opal-intc` compatible device must exist in the device tree (see *I - Device-tree updates*). If OPAL does not create such a device, the host OS MUST NOT use this call.

4.4.30 OPAL_INVALID_CALL

An OPAL call of `-1` will always return `OPAL_PARAMETER`. It is always ivalid.

It exists purely for testing.

4.4.31 OPAL_IPMI_SEND

```
#define OPAL_IPMI_SEND                                107

enum {
            OPAL_IPMI_MSG_FORMAT_VERSION_1 = 1,
};

struct opal_ipmi_msg {
            uint8_t version;
            uint8_t netfn;
            uint8_t cmd;
            uint8_t data[];
};

int64_t opal_ipmi_send(uint64_t interface,
                       struct opal_ipmi_msg *opal_ipmi_msg, uint64_t msg_len);
```

OPAL_IPMI_SEND call will send an IPMI message to the service processor.

Parameters

interface `interface` parameter is the value from the ipmi interface node `ibm,ipmi-interface-id`

opal_ipmi_msg `opal_ipmi_msg` is the pointer to a `struct opal_ipmi_msg` (see above)

msg_len ipmi message request size

Return Values

OPAL_SUCCESS `msg` queued successfully

OPAL_PARAMETER invalid ipmi message request length `msg_len`

OPAL_HARDWARE backend support is not present as block transfer/service processor ipmi routines are not initialized which are used for communication

OPAL_UNSUPPORTED in-correct opal ipmi message format version `opal_ipmi_msg->version`

OPAL_RESOURCE insufficient resources to create `ipmi_msg` structure

4.4.32 OPAL_IPMI_RECV

```
#define OPAL_IPMI_RECV                          108

enum {
              OPAL_IPMI_MSG_FORMAT_VERSION_1 = 1,
};

struct opal_ipmi_msg {
              uint8_t version;
              uint8_t netfn;
              uint8_t cmd;
              uint8_t data[];
};

int64_t opal_ipmi_recv(uint64_t interface,
                          struct opal_ipmi_msg *opal_ipmi_msg, uint64_t *msg_len)
```

`OPAL_IPMI_RECV` call reads an ipmi message of type `ipmi_msg` from ipmi message queue `msgq` into host OS structure `opal_ipmi_msg`.

Parameters

interface `interface` parameter is the value from the ipmi interface node `ibm,ipmi-interface-id`

opal_ipmi_msg `opal_ipmi_msg` is the pointer to a `struct opal_ipmi_msg` (see above)

msg_len `msg_len` is the pointer to ipmi message response size

Return Values

OPAL_SUCCESS ipmi message dequeued from `msgq` queue and memory taken by it got released successfully

OPAL_EMPTY `msgq` list is empty

ref:*OPAL_PARAMETER* invalid ipmi `interface` value

OPAL_UNSUPPORTED incorrect opal ipmi message format version `opal_ipmi_msg->version`

4.4.33 Service Indicators (LEDS)

The service indicator is one element of an overall hardware service strategy where end user simplicity is a high priority. The goal is system firmware or operating system code to isolate hardware failures to the failing FRU and automatically activate the fault indicator associated with the failing FRU. The end user then needs only to look for the FRU with the active fault indicator to know which part to replace.

Different types of indicators handled by LED code:

- **System attention indicator (Check log indicator)** Indicates there is a problem with the system that needs attention.
- **Identify** Helps the user locate/identify a particular FRU or resource in the system.
- **Fault** Indicates there is a problem with the FRU or resource at the location with which the indicator is associated.

All LEDs are defined in the device tree (see *Service Indicators (LEDS)*).

LED Design

When it comes to implementation we can classify LEDs into two categories:

1. Hypervisor (OPAL) controlled LEDs (All identify & fault indicators) During boot, we read/cache these LED details in OPAL (location code, state, etc). We use cached data to serve read request from FSP/Host. And we use SPCN passthrough MBOX command to update these LED state.
2. Service processor (FSP) controlled LEDs (System Attention Indicator) During boot, we read/cache this LED info using MBOX command. Later anytime FSP updates this LED, it sends update system parameter notification MBOX command. We use that data to update cached data. LED update request is sent via set/reset attn MBOX command.

LED update request: Both FSP and Host will send LED update requests. We have to serialize SPCN passthrough command. Hence we maintain local queue.

Note:

- For more information regarding service indicator refer to PAPR spec (Service Indicators chapter).

There are two OPAL calls relating to LED operations.

OPAL_LEDS_GET_INDICATOR

```
#define OPAL_LEDS_GET_INDICATOR                     114

int64_t opal_leds_get_indicator(char *loc_code, u64 *led_mask,
                                u64 *led_value, u64 *max_led_type);
```

Returns LED state for the given location code.

`loc_code` Location code of the LEDs.

`led_mask` LED types whose status is available (return by OPAL)

`led_value` Status of the available LED types (return by OPAL)

`max_led_type` Maximum number of supported LED types (Host/OPAL)

The host will pass the location code of the LED types (loc_code) and maximum number of LED types it understands (max_led_type). OPAL will update the 'led_mask' with set bits pointing to LED types whose status is available and updates the 'led_value' with actual status. OPAL checks the 'max_led_type' to understand whether the host is newer or older compared to itself. In the case where the OPAL is newer compared to host (OPAL's max_led_type > host's max_led_type), it will update led_mask and led_value according to max_led_type requested by the host. When the host is newer compared to the OPAL (host's max_led_type > OPAL's max_led_type), OPAL updates 'max_led_type' to the maximum number of LED type it understands and updates 'led_mask', 'led_value' based on that maximum value of LED types.

Currently this is only implemented on FSP basde machines, see hw/fsp/fsp-leds.c for more deatails.

OPAL_LEDS_SET_INDICATOR

```
#define OPAL_LEDS_SET_INDICATOR                       115

int64_t opal_leds_set_indicator(uint64_t async_token,
                                char *loc_code, const u64 led_mask,
                                const u64 led_value, u64 *max_led_type);
```

Sets LED state for the given location code.

loc_code Location code of the LEDs to be set.

led_mask LED types whose status will be updated

led_value Requested status of various LED types.

max_led_type Maximum number of supported LED types. If OPAL supports fewer LED types than requested, it will set `max_led_type` to the maximum it does support.

The host will pass the location code of the LED types, mask, value and maximum number of LED types it understands. OPAL will update LED status for all the LED types mentioned in the mask with their value mentioned. OPAL checks the 'max_led_type' to understand whether the host is newer or older compared to itself. In case where the OPAL is newer compared to the host (OPAL's max_led_type >

host's max_led_type), it updates LED status based on max_led_type

requested from the host. When the host is newer compared to the OPAL (host's max_led_type > OPAL's max_led_type), OPAL updates 'max_led_type' to the maximum number of LED type it understands and then it updates LED status based on that updated maximum value of LED types. Host needs to check the returned updated value of max_led_type to figure out which part of it's request got served and which ones got ignored.

Currently this is only implemented on FSP basde machines, see hw/fsp/fsp-leds.c for more deatails.

4.4.34 OPAL_LPC_READ

```
#define OPAL_LPC_READ                                 67

/*
 * Address cycle types for LPC accesses. These also correspond
 * to the content of the first cell of the "reg" property for
 * device nodes on the LPC bus
 */
 enum OpalLPCAddressType {
   OPAL_LPC_MEM      = 0,
   OPAL_LPC_IO       = 1,
   OPAL_LPC_FW       = 2,
```

(continues on next page)

(continued from previous page)

```
};

int64_t opal_lpc_read(uint32_t chip_id, enum OpalLPCAddressType addr_type,
                            uint32_t addr, uint32_t *data, uint32_t sz);
```

This function related to Low Pin Count (LPC) bus. This function reads the data from IDSEL register for `chip_id`, which has LPC information. From `addr` for `addr_type` with read size `sz` bytes in to a variable named `data`.

Parameters

`chip_id` The `chip_id` parameter contains value of the chip number identified at boot time.

`addr_type` The `addr_type` is one of the LPC supported address types. Supported address types are:

- LPC memory,
- LPC IO and
- LPC firmware.

`addr` The `addr` from which the data has to be read.

`data` The `data` will be used to store the read data.

`sz` How many `sz` bytes to be read in to `data`.

Return Codes

OPAL_PARAMETER Indicates either `chip_id` not found or `chip_id` doesn't contain LPC information.

OPAL_SUCCESS Indicates Success!

4.4.35 OPAL_LPC_WRITE

```
#define OPAL_LPC_WRITE                              68

/*
 * Address cycle types for LPC accesses. These also correspond
 * to the content of the first cell of the "reg" property for
 * device nodes on the LPC bus
 */
 enum OpalLPCAddressType {
   OPAL_LPC_MEM      = 0,
   OPAL_LPC_IO       = 1,
   OPAL_LPC_FW       = 2,
};

int64_t opal_lpc_write(uint32_t chip_id, enum OpalLPCAddressType addr_type,
                         uint32_t addr, uint32_t data, uint32_t sz);
```

This function related to Low Pin Count (LPC) bus. This function writes the `data` in to ECCB register for `chip_id`, which has LPC information. From `addr` for `addr_type` with write size `sz` bytes.

Parameters

chip_id The chip_id parameter contains value of the chip number identified at boot time.

addr_type The addr_type is one of the address types LPC supported. Supported address types are:

- LPC memory,
- LPC IO and
- LPC firmware.

addr The addr to where the data need to be written.

data The data for writing.

sz How many sz bytes to write.

Return Codes

OPAL_PARAMETER Indicates either chip_id not found or chip_id doesn't contain LPC information.

OPAL_SUCCESS Indicates Success!

4.4.36 OPAL_MESSAGE

The host OS can use OPAL_GET_MSG to retrive messages queued by OPAL. The messages are defined by enum opal_msg_type. The host is notified of there being messages to be consumed by the OPAL_EVENT_MSG_PENDING bit being set.

An opal_msg is:

```
struct opal_msg {
      __be32 msg_type;
      __be32 size;
      __be64 params[8];
};
```

The data structure is ALWAYS at least this size (4+4+8*8 = 72 bytes). Some messages define fewer than eight parameters. For messages that do not define all eight parameters, the value in the undefined parameters is undefined, although can safely be memcpy()d or otherwise moved.

In the device tree, there's an opal-msg-size property of the OPAL node that says the size of a struct opal-msg. Kernel will use this property to allocate memory for opal_msg structure. See OPAL_GET_MESSAGE documentation for details.

```
ibm,opal {
          opal-msg-size = <0x48>;
}
```

OPAL_MSG_ASYNC_COMP

```
params[0] = token
params[1] = rc
```

Additional parameters are function-specific.

OPAL_MSG_MEM_ERR

OPAL_MSG_EPOW

Used by OPAL to issue environmental and power warnings to host OS for conditions requiring an earlier poweroff. A few examples of these are high ambient temperature or system running on UPS power with low UPS battery. Host OS can query OPAL via *OPAL_GET_EPOW_STATUS* API to obtain information about EPOW conditions present. Refer include/opal-api.h for description of all supported EPOW events. OPAL_SYSPOWER_CHNG, OPAL_SYSPOWER_FAIL and OPAL_SYSPOWER_INC events don't require system poweroff.

Host OS should look for 'ibm,opal-v3-epow' string as compatible property for 'epow' node under OPAL device-tree to determine epow support.

OPAL_MSG_SHUTDOWN

Used by OPAL to inform the host OS it must imitate a graceful shutdown. Uses the first parameter to indicate weather the system is going down for shutdown or a reboot.

```
params[0] = 0x01 reboot, 0x00 shutdown
```

OPAL_MSG_HMI_EVT

Used by OPAL to sends the OPAL HMI Event to the host OS that reports a summary of HMI error and whether it was successfully recovered or not.

HMI is a Hypervisor Maintenance Interrupt usually reports error related to processor recovery/checkstop, NX checkstop and Timer facility. Hypervisor then takes this opportunity to analyze and recover from some of these errors. Hypervisor takes assistance from OPAL layer to handle and recover from HMI. After handling HMI, OPAL layer sends the summary of error report and status of recovery action using HMI event structure shown below.

The HMI event structure uses version numbering to allow future enhancement to accommodate additional members. The version start from V1 onward. Version 0 is invalid version and unsupported.

The current version of HMI event structure V2 and is backward compatible to V1 version.

Notes:

- When adding new structure to the union in future, the version number must be bumped.
- All future versions must be backward compatible to all its older versions.
- Size of this structure should not exceed that of struct opal_msg.

```
struct OpalHMIEvent {
     uint8_t          version;        /* 0x00 */
     uint8_t          severity;       /* 0x01 */
     uint8_t          type;           /* 0x02 */
     uint8_t          disposition;    /* 0x03 */
     uint8_t          reserved_1[4];  /* 0x04 */

     __be64           hmer;
     /* TFMR register. Valid only for TFAC and TFMR_PARITY error type. */
     __be64           tfmr;

     /* version 2 and later */
     union {
             /*
```

(continues on next page)

(continued from previous page)

```
                 * checkstop info (Core/NX).
                 * Valid for OpalHMI_ERROR_MALFUNC_ALERT.
                 */
                struct {
                        uint8_t xstop_type;     /* enum OpalHMI_XstopType */
                        uint8_t reserved_1[3];
                        __be32 xstop_reason;
                        union {
                                __be32 pir;       /* for CHECKSTOP_TYPE_CORE */
                                __be32 chip_id; /* for CHECKSTOP_TYPE_NX */
                        } u;
                } xstop_error;
        } u;
};
```

OPAL_MSG_DPO

Delayed poweroff where OPAL informs host OS that a poweroff has been requested and a forced shutdown will happen in future. Host OS can use OPAL_GET_DPO_STATUS API to query OPAL the number of seconds remaining before a forced poweroff will occur.

OPAL_MSG_PRD

This message is a OPAL-to-HBRT notification, and contains a struct opal_prd_msg:

```
enum opal_prd_msg_type {
        OPAL_PRD_MSG_TYPE_INIT = 0,     /* HBRT --> OPAL */
        OPAL_PRD_MSG_TYPE_FINI,         /* HBRT --> OPAL */
        OPAL_PRD_MSG_TYPE_ATTN,         /* HBRT <-- OPAL */
        OPAL_PRD_MSG_TYPE_ATTN_ACK,     /* HBRT --> OPAL */
        OPAL_PRD_MSG_TYPE_OCC_ERROR,    /* HBRT <-- OPAL */
        OPAL_PRD_MSG_TYPE_OCC_RESET,    /* HBRT <-- OPAL */
};

struct opal_prd_msg {
        uint8_t         type;
        uint8_t         pad[3];
        __be32          token;
        union {
                struct {
                        __be64  version;
                        __be64  ipoll;
                } init;
                struct {
                        __be64  proc;
                        __be64  ipoll_status;
                        __be64  ipoll_mask;
                } attn;
                struct {
                        __be64  proc;
                        __be64  ipoll_ack;
                } attn_ack;
                struct {
                        __be64  chip;
```

(continues on next page)

(continued from previous page)

```
                } occ_error;
                struct {
                        __be64  chip;
                } occ_reset;
        };
};
```

Responses from the kernel use the same message format, but are passed through the *OPAL_PRD_MSG* call.

OPAL_MSG_OCC

This is used by OPAL to inform host about OCC events like OCC reset, OCC load and throttle status change by OCC which can indicate the host the reason for frequency throttling/unthrottling.

```
#define OCC_RESET                     0
#define OCC_LOAD                      1
#define OCC_THROTTLE                  2
#define OCC_MAX_THROTTLE_STATUS               5
/*
 * struct opal_occ_msg:
 * type: OCC_RESET, OCC_LOAD, OCC_THROTTLE
 * chip: chip id
 * throttle status: Indicates the reason why OCC may have limited
 * the max Pstate of the chip.
 * 0x00 = No throttle
 * 0x01 = Power Cap
 * 0x02 = Processor Over Temperature
 * 0x03 = Power Supply Failure (currently not used)
 * 0x04 = Over current (currently not used)
 * 0x05 = OCC Reset (not reliable as some failures will not allow for
 * OCC to update throttle status)
 */
struct opal_occ_msg {
      __be64 type;
      __be64 chip;
      __be64 throttle_status;
};
```

Host should read opal_occ_msg.chip and opal_occ_msg.throttle_status only when `opal_occ_msg.type = OCC_THROTTLE`. If host receives `OCC_THROTTLE` after an `OCC_RESET` then this throttle message will have a special meaning which indicates that all the OCCs have become active after a reset. In such cases `opal_occ_msg.chip` and `opal_occ_msg.throttle_status` will be set to 0 and host should not use these values.

If `opal_occ_msg.type > 2` then host should ignore the message for now, new events can be defined for `opal_occ_msg.type` in the future versions of OPAL.

OPAL_MSG_PRD2

This message is a OPAL-to-HBRT notification. Its same as OPAL_MSG_PRD except this one supports passing more than 64bytes (8*8) of data.

4.4.37 OPAL_NMMU_SET_PTCR

```
#define OPAL_NMMU_SET_PTCR                      127

int64 opal_nmmu_set_ptcr(uint64 chip_id, uint64_t ptcr);
```

uint64 chip_id either the chip id containing the nest mmu who's ptcr should be set or alternatively -1ULL to indicate all nest mmu ptcr's should be set to the same value.

uint64 ptcr ptcr value pointing to either the radix tables or hash tables.

This OPAL call sets up the Nest MMU by pointing it at the radix page table base or the hash page table base (HTABORG).

Return Values

OPAL_SUCCESS the PTCR was updated successful

OPAL_PARAMETER a parameter was incorrect

4.4.38 OPAL NPU2 calls

There are three OPAL calls for interacting with NPU2 devices:

```
#define OPAL_NPU_INIT_CONTEXT                   146
#define OPAL_NPU_DESTROY_CONTEXT                147
#define OPAL_NPU_MAP_LPAR                       148
```

These are used to setup and configure address translation services (ATS) for a given NVLink2 device. Note that in some documentation this is also referred to as extended translation services (XTS).

Each NVLink2 supports multiple processes running on a GPU which issues requests for address translation. The NPU2 is responsible for completing the request by forwarding it to the Nest MMU (NMMU) along with the appropriate translation context (MSR/LPCR) bits. These bits are keyed off a 20-bit process ID (PASID/PID) which is identical to the PID used on the processor.

The OPAL calls documented here are used to setup/destroy the appropriate context for a given process on a given NVLink2 device.

OPAL_NPU_INIT_CONTEXT

Parameters:

```
uint64_t phb_id
int pasid
uint64_t msr
uint64_t lpid
```

Allocates a new context ID and sets up the given PASID/PID to be associated with the supplied MSR on for the given LPID. MSR should only contain bits set requried for NPU2 address lookups - ie. MSR DR/HV/PR/SF.

Returns the context ID on success or `OPAL_RESOURCE` if no more contexts are available or `OPAL_UNSUPPORTED` in the case of unsupported MSR bits.

OPAL_NPU_DESTROY_CONTEXT

Parameters:

```
uint64_t phb_id
uint64_t id
```

Destroys a previously allocated context ID. This may cause further translation requests from the GPU to fail.

OPAL_NPU_MAP_LPAR

Parameters:

```
uint64_t phb_id
uint64_t bdf
uint64_t lparid
uint64_t lpcr
```

Associates the given GPU BDF with a particular LPAR and LPCR bits. Hash mode ATS is currently unsupported so lpcr should be set to 0.

4.4.39 OPAL_NPU_SET_RELAXED_ORDER

Request that relaxed memory ordering be enabled or disabled for a device.

Parameters

```
uint64_t phb_id
uint16_t bdfn
bool request_enabled
```

phb_id OPAL ID of the PHB

bdfn Bus-Device-Function number of the device

request_enabled Requested state of relaxed memory ordering enablement

Return values

OPAL_SUCCESS Requested state set

OPAL_PARAMETER The given phb_id or bdfn is invalid or out of range

OPAL_CONSTRAINED Relaxed ordering can not be enabled until an enable request is made for every device on this PHB.

OPAL_RESOURCE No more relaxed ordering sources are available

4.4.40 OPAL_NPU_GET_RELAXED_ORDER

Query the relaxed memory ordering state of a device.

Parameters

```
uint64_t phb_id
uint64_t bdfn
```

phb_id OPAL ID of the PHB

bdfn Bus-Device-Function number of the device

Return values

On success, the current relaxed ordering state is returned.

OPAL_PARAMETER The given phb_id or bdfn is invalid.

4.4.41 OPAL_NPU_SPA_SETUP

OpenCAPI devices only.

Sets up a Shared Process Area (SPA) with the Shared Process Area Pointer (SPAP) set to the provided address *addr*, and sets the OTL PE mask (used for PASID to PE handle conversion) to *PE_mask*.

If *addr* is NULL, the SPA will be disabled. *addr* must be 4K aligned.

Parameters

```
uint64_t phb_id
int bdfn
uint64_t addr
uint64_t PE_mask
```

phb_id OPAL ID of PHB

bdfn Bus-Device-Function number of OpenCAPI AFU

addr Address of Shared Process Area, or NULL to disable SPA. Must be 4K aligned.

PE_mask Process Element mask for PASID to PE handle conversion

Return Values

OPAL_SUCCESS SPAP and PE mask were successfully set

OPAL_PARAMETER A provided parameter was invalid

OPAL_BUSY SPA is already enabled (or if addr is NULL, SPA is already disabled)

4.4.42 OPAL_NPU_SPA_CLEAR_CACHE

OpenCAPI devices only.

Invalidates the Process Element with the given *PE_handle* from the NPU's SPA cache.

Parameters

```
uint64_t phb_id
uint32_t bdfn
uint64_t PE_handle
```

phb_id OPAL ID of PHB

bdfn Bus-Device-Function number of OpenCAPI AFU

PE_handle Handle of Process Element being cleared from SPA cache

Return Values

OPAL_SUCCESS PE was successfully cleared from SPA cache

OPAL_PARAMETER A provided parameter was invalid

OPAL_BUSY XSLO is currently invalidating a previously requested entry

4.4.43 OPAL_NPU_TL_SET

OpenCAPI devices only.

Update the NPU OTL configuration with device capabilities.

Parameters

```
uint64_t phb_id
uint32_t bdfn
long capabilities
uint64_t rate_phys
int rate_sz
```

phb_id OPAL ID of PHB

bdfn Bus-Device-Function number of OpenCAPI AFU

capabilities Bitmap of TL templates the device can receive

rate_phys Physical address of rates buffer

rate_sz Size of rates buffer (must be equal to 32)

Return Values

OPAL_SUCCESS OTL configuration was successfully updated

OPAL_PARAMETER

A provided parameter was invalid

4.4.44 OPAL_NPU_MEM_ALLOC

OpenCAPI devices only.

Sets up the NPU memory BAR for Lowest Point of Coherency (LPC) memory.

At present, only one device per CPU can use LPC memory, and a maximum of 4TB can be allocated.

Parameters

```
uint64_t phb_id
uint32_t bdfn
uint64_t size
uint64_t *bar
```

phb_id OPAL ID of PHB

bdfn Bus-Device-Function number of OpenCAPI AFU

size Size of requested LPC memory area in bytes

bar Pointer to variable where base of LPC memory area will be returned

Return Values

OPAL_SUCCESS BAR setup completed successfully

OPAL_PARAMETER A provided parameter was invalid

OPAL_RESOURCE The BAR could not be assigned due to limitations

4.4.45 OPAL_NPU_MEM_RELEASE

OpenCAPI devices only.

Releases NPU memory BAR.

Parameters

```
uint64_t phb_id
uint32_t bdfn
```

phb_id OPAL ID of PHB

bdfn Bus-Device-Function number of OpenCAPI AFU

Return Values

OPAL_SUCCESS BAR setup completed successfully

OPAL_PARAMETER A provided parameter was invalid, or the specified device does not currently have LPC memory assigned

4.4.46 OPAL NVRAM

The NVRAM requirements for OPAL systems is derived from LoPAPR, and all requirements listed in it apply to OPAL with some exceptions. Note that Section 8.4.1.1.3 "OF Configuration Variables" does NOT apply to OPAL, neither does 8.4.1.2 "DASD Spin-up Control". Not that the RTAS calls of *nvram-fetch* and *nvram-store* are roughly equivalent to the *OPAL_READ_NVRAM* and *OPAL_WRITE_NVRAM* calls.

LoPAPR has a minimum requirement of 8KB of Non-Volatile Memory. While this requirement carries over, it's important to note that historically all OPAL systems have had roughly 500kb of NVRAM.

See *nvram Device Tree Node* for details on how NVRAM is represented in the device tree. It's fairly simple, it looks like this:

```
nvram {
      compatible = "ibm,opal-nvram";
      #bytes = <0x90000>;
};
```

OPAL_READ_NVRAM

```
#define OPAL_READ_NVRAM                         7

int64_t opal_read_nvram(uint64_t buffer, uint64_t size, uint64_t offset);
```

OPAL_READ_NVRAM call requests OPAL to read the data from system NVRAM memory into a memory buffer. The data at `offset` from nvram_image will be copied to memory `buffer` of size `size`.

This is a *synchronous* OPAL call, as OPAL will typically read the content of NVRAM from its storage (typically flash) during boot, so the call duration should be along the lines of a `memcpy()` operation rather than reading from storage.

Parameters

```
uint64_t buffer
uint64_t size
uint64_t offset
```

`buffer` the data from nvram will be copied to `buffer`

`size` the data of size `size` will be copied

`offset` the data will be copied from address equal to base `nvram_image` plus `offset`

Return Values

OPAL_SUCCESS data from nvram to memory `buffer` copied successfully

OPAL_PARAMETER a parameter `offset` or `size` was incorrect

OPAL_HARDWARE either nvram is not initialized or permanent error related to nvram hardware.

OPAL_WRITE_NVRAM

```
#define OPAL_WRITE_NVRAM                        8

int64_t opal_write_nvram(uint64_t buffer, uint64_t size, uint64_t offset);
```

OPAL_WRITE_NVRAM call requests OPAL to write the data to actual system NVRAM memory from memory `buffer` at `offset`, of size `size`

Parameters

```
uint64_t buffer
uint64_t size
uint64_t offset
```

`buffer` data from `buffer` will be copied to nvram

`size` the data of size `size` will be copied

`offset` the data will be copied to address which is equal to base `nvram_image` plus `offset`

Return Values

OPAL_SUCCESS data from memory `buffer` to actual nvram_image copied successfully

OPAL_PARAMETER a parameter `offset` or `size` was incorrect

OPAL_HARDWARE either nvram is not initialized or permanent error related to nvram hardware.

OPAL_BUSY OPAL is currently busy, retry the *OPAL_WRITE_NVRAM* call.

OPAL_BUSY_EVENT OPAL is currently busy, call *OPAL_POLL_EVENTS* and then retry *OPAL_WRITE_NVRAM*

4.4.47 Get/Set System Parameters

The usual way for setting system parameters is via IPMI for things controlled by the service processor, or through NVRAM for things controlled by host firmware. However, some platforms may have other options not easily (or possible to be) exposed over IPMI. These OPAL calls will read (and write) these parameters.

The list of parameters is set at boot time, and is represented in the device tree (see *sysparams* for details).

Currently only implemented on FSP based systems.

OPAL_GET_PARAM

```
#define OPAL_GET_PARAM                          89

int64_t fsp_opal_get_param(uint64_t async_token, uint32_t param_id,
                           uint64_t buffer, uint64_t length);
```

Get the current setting of *param_id*. This is an asynchronous call as OPAL may need to communicate with a service processor. The *param_id* and *length* are described in the device tree for each parameter (see *sysparams* for details).

Returns

OPAL_HARDWARE Hardware issue prevents retreiving parameter. e.g. FSP is offline or absent.

OPAL_PARAMETER Invalid *param_id*

OPAL_PERMISSION Not allowed to read parameter.

OPAL_NO_MEM Not enough free memory in OPAL to process request.

OPAL_INTERNAL_ERROR Other internal OPAL error

OPAL_ASYNC_COMPLETION Request is submitted.

OPAL_SET_PARAM

```
#define OPAL_SET_PARAM                          90

int64_t fsp_opal_set_param(uint64_t async_token, uint32_t param_id,
                           uint64_t buffer, uint64_t length);
```

Write a new setting for *param_id*. This is an asynchronous call as OPAL may need to communicate with a service processor. The *param_id* and *length* are described in the device tree for each parameter (see *sysparams* for details).

Returns

OPAL_HARDWARE Hardware issue prevents retreiving parameter. e.g. FSP is offline or absent.

OPAL_PARAMETER Invalid *param_id*

OPAL_PERMISSION Not allowed to write parameter.

OPAL_NO_MEM Not enough free memory in OPAL to process request.

OPAL_INTERNAL_ERROR Other internal OPAL error

OPAL_ASYNC_COMPLETION Request is submitted.

4.4.48 OPAL PCI Config Space Access

PCI Config space is read or written to through OPAL calls. All of these calls

OPAL_PCI_CONFIG_* Return codes

OPAL_SUCCESS Read/Write operation completed successfully.

OPAL_PARAMETER Invalid parameter. e.g. invalid *phb_id* or *bus_dev_func*.

OPAL_HARDWARE Invalid request for the hardware either permanently or in its current state. Can also be a hardware problem, e.g. fenced or config access is currently blocked.

OPAL_UNSUPPORTED Unsupported operation. For example, phb4 doesn't support ASB config space writes.

Other return codes Should be handled gracefully. For example, for any return code other than *OPAL_SUCCESS*, Linux will return all bits set for the specified size for a read, and will ignore the error on a write.

OPAL_PCI_CONFIG_READ_BYTE

```
#define OPAL_PCI_CONFIG_READ_BYTE            13

int64_t opal_pci_config_read_byte(uint64_t phb_id,
                                  uint64_t bus_dev_func,
                                  uint64_t offset,
                                  uint8_t *data);
```

Reads a single byte from PCI config space, see *OPAL_PCI_CONFIG_* Return codes*.

OPAL_PCI_CONFIG_READ_HALF_WORD

```
#define OPAL_PCI_CONFIG_READ_HALF_WORD       14

int64_t opal_pci_config_read_half_word(uint64_t phb_id,
                                       uint64_t bus_dev_func,
                                       uint64_t offset,
                                       uint16_t *data);
```

Reads a half word (16 bits) from PCI config space, see *OPAL_PCI_CONFIG_* Return codes*.

OPAL_PCI_CONFIG_READ_WORD

```
#define OPAL_PCI_CONFIG_READ_WORD            15

int64_t opal_pci_config_read_word(uint64_t phb_id,
                                  uint64_t bus_dev_func,
                                  uint64_t offset,
                                  uint32_t *data);
```

Reads a word (32 bits) from PCI config space, see *OPAL_PCI_CONFIG_* Return codes*.

OPAL_PCI_CONFIG_WRITE_BYTE

```
#define OPAL_PCI_CONFIG_WRITE_BYTE           16

int64_t opal_pci_config_write_byte(uint64_t phb_id,
                                   uint64_t bus_dev_func,
                                   uint64_t offset,
                                   uint8_t data);
```

Writes a byte (8 bits) to PCI config space, see *OPAL_PCI_CONFIG_* Return codes*.

OPAL_PCI_CONFIG_WRITE_HALF_WORD

```
#define OPAL_PCI_CONFIG_WRITE_HALF_WORD              17

int64_t opal_pci_config_read_half_word(uint64_t phb_id,
                                       uint64_t bus_dev_func,
```

(continues on next page)

(continued from previous page)

```
                                         uint64_t offset,
                                         uint16_t data);
```

Writes a half word (16 bits) to PCI config space, see *OPAL_PCI_CONFIG_* Return codes*.

OPAL_PCI_CONFIG_WRITE_WORD

```
#define OPAL_PCI_CONFIG_WRITE_WORD           18

int64_t opal_pci_config_read_word(uint64_t phb_id,
                                  uint64_t bus_dev_func,
                                  uint64_t offset,
                                  uint32_t data);
```

Writes a word (32 bits) to PCI config space, see *OPAL_PCI_CONFIG_* Return codes*.

4.4.49 OPAL_PCI_EEH_FREEZE_CLEAR

```
#define OPAL_PCI_EEH_FREEZE_CLEAR            26

enum OpalEehFreezeActionToken {
     OPAL_EEH_ACTION_CLEAR_FREEZE_MMIO = 1,
     OPAL_EEH_ACTION_CLEAR_FREEZE_DMA = 2,
     OPAL_EEH_ACTION_CLEAR_FREEZE_ALL = 3,

     OPAL_EEH_ACTION_SET_FREEZE_MMIO = 1,
     OPAL_EEH_ACTION_SET_FREEZE_DMA  = 2,
     OPAL_EEH_ACTION_SET_FREEZE_ALL  = 3
};

int64_t opal_pci_eeh_freeze_clear(uint64_t phb_id, uint64_t pe_number, uint64_t eeh_
↪action_token);
```

Returns

OPAL_SUCCESS Success!

OPAL_PARAMETER Invalid PHB

OPAL_UNSUPPORTED PHB doesn't support this operation.

OPAL_HARDWARE Hardware issue prevents completing operation. OPAL may have detected it being broken.

4.4.50 OPAL_PCI_EEH_FREEZE_SET

```
#define OPAL_PCI_EEH_FREEZE_SET                      97

enum OpalEehFreezeActionToken {
     OPAL_EEH_ACTION_CLEAR_FREEZE_MMIO = 1,
     OPAL_EEH_ACTION_CLEAR_FREEZE_DMA = 2,
     OPAL_EEH_ACTION_CLEAR_FREEZE_ALL = 3,
```

(continues on next page)

(continued from previous page)

```
     OPAL_EEH_ACTION_SET_FREEZE_MMIO = 1,
     OPAL_EEH_ACTION_SET_FREEZE_DMA  = 2,
     OPAL_EEH_ACTION_SET_FREEZE_ALL  = 3
};

int64_t opal_pci_eeh_freeze_set(uint64_t phb_id, uint64_t pe_number, uint64_t eeh_
↪action_token);
```

Returns

OPAL_PARAMETER Invalid parameter.

OPAL_UNSUPPORTED Unsupported operation

OPAL_HARDWARE Hardware in a bad state.

4.4.51 OPAL_PCI_EEH_FREEZE_STATUS

```
#define OPAL_PCI_EEH_FREEZE_STATUS           23

enum OpalFreezeState {
     OPAL_EEH_STOPPED_NOT_FROZEN = 0,
     OPAL_EEH_STOPPED_MMIO_FREEZE = 1,
     OPAL_EEH_STOPPED_DMA_FREEZE = 2,
     OPAL_EEH_STOPPED_MMIO_DMA_FREEZE = 3,
     OPAL_EEH_STOPPED_RESET = 4,
     OPAL_EEH_STOPPED_TEMP_UNAVAIL = 5,
     OPAL_EEH_STOPPED_PERM_UNAVAIL = 6
};

enum OpalPciStatusToken {
     OPAL_EEH_NO_ERROR        = 0,
     OPAL_EEH_IOC_ERROR       = 1,
     OPAL_EEH_PHB_ERROR       = 2,
     OPAL_EEH_PE_ERROR        = 3,
     OPAL_EEH_PE_MMIO_ERROR   = 4,
     OPAL_EEH_PE_DMA_ERROR    = 5
};

int64_t opal_pci_eeh_freeze_status(uint64_t phb_id, uint64_t pe_number,
                                   uint8_t *freeze_state,
                                   uint16_t *pci_error_type,
                                   uint64_t *phb_status);
```

Note: The `phb_status` parameter is deprecated as of *skiboot-6.3-rc1*. Linux only ever passed in NULL, and this was safe. Supplying a pointer was previously *unsafe*. Always pass NULL.

Note: There once was a *OPAL_PCI_EEH_FREEZE_STATUS2* call, but it was introduced in firmware and never used by any OS, so it has since been removed from OPAL.

Returns

OPAL_PARAMETER Invalid address or PHB.

OPAL_UNSUPPORTED PHB does not support this operation.

OPAL_HARDWARE Hardware prohibited getting status, OPAL maybe marked it as broken.

OPAL_SUCCESS Retreived status.

4.4.52 OPAL_PCI_EEH_FREEZE_STATUS2

```
#define OPAL_PCI_EEH_FREEZE_STATUS2           61
```

Use *OPAL_PCI_EEH_FREEZE_STATUS* instead of this (removed) call.

While you'd think that a call introduced in the first public OPAL release would have been used somewhere, it seems that all existing code has only ever used *OPAL_PCI_EEH_FREEZE_STATUS* over *OPAL_PCI_EEH_FREEZE_STATUS2*.

This call has been removed as of skiboot-6.4 as it has literally never been used.

4.4.53 OPAL_PCI_ERR_INJECT

```
#define OPAL_PCI_ERR_INJECT                   96

enum OpalErrinjectType {
     OPAL_ERR_INJECT_TYPE_IOA_BUS_ERR         = 0,
     OPAL_ERR_INJECT_TYPE_IOA_BUS_ERR64       = 1,
};

enum OpalErrinjectFunc {
     /* IOA bus specific errors */
     OPAL_ERR_INJECT_FUNC_IOA_LD_MEM_ADDR     = 0,
     OPAL_ERR_INJECT_FUNC_IOA_LD_MEM_DATA     = 1,
     OPAL_ERR_INJECT_FUNC_IOA_LD_IO_ADDR      = 2,
     OPAL_ERR_INJECT_FUNC_IOA_LD_IO_DATA      = 3,
     OPAL_ERR_INJECT_FUNC_IOA_LD_CFG_ADDR     = 4,
     OPAL_ERR_INJECT_FUNC_IOA_LD_CFG_DATA     = 5,
     OPAL_ERR_INJECT_FUNC_IOA_ST_MEM_ADDR     = 6,
     OPAL_ERR_INJECT_FUNC_IOA_ST_MEM_DATA     = 7,
     OPAL_ERR_INJECT_FUNC_IOA_ST_IO_ADDR      = 8,
     OPAL_ERR_INJECT_FUNC_IOA_ST_IO_DATA      = 9,
     OPAL_ERR_INJECT_FUNC_IOA_ST_CFG_ADDR     = 10,
     OPAL_ERR_INJECT_FUNC_IOA_ST_CFG_DATA     = 11,
     OPAL_ERR_INJECT_FUNC_IOA_DMA_RD_ADDR     = 12,
     OPAL_ERR_INJECT_FUNC_IOA_DMA_RD_DATA     = 13,
     OPAL_ERR_INJECT_FUNC_IOA_DMA_RD_MASTER   = 14,
     OPAL_ERR_INJECT_FUNC_IOA_DMA_RD_TARGET   = 15,
     OPAL_ERR_INJECT_FUNC_IOA_DMA_WR_ADDR     = 16,
     OPAL_ERR_INJECT_FUNC_IOA_DMA_WR_DATA     = 17,
     OPAL_ERR_INJECT_FUNC_IOA_DMA_WR_MASTER   = 18,
     OPAL_ERR_INJECT_FUNC_IOA_DMA_WR_TARGET   = 19,
};

int64_t opal_pci_err_inject(uint64_t phb_id, uint64_t pe_number,
                            uint32_t type, uint32_t func,
                            uint64_t addr, uint64_t mask);
```

Inject an error, used to test OS and OPAL EEH handling.

Returns

OPAL_SUCCESS Error injected successfully.

OPAL_PARAMETER Invalid argument.

OPAL_UNSUPPORTED PHB doesn't support error injection or the specific error attempting to be injected.

4.4.54 OPAL_PCI_GET_HUB_DIAG_DATA

```
#define OPAL_PCI_GET_HUB_DIAG_DATA           50

enum {
     OPAL_P7IOC_DIAG_TYPE_NONE       = 0,
     OPAL_P7IOC_DIAG_TYPE_RGC        = 1,
     OPAL_P7IOC_DIAG_TYPE_BI         = 2,
     OPAL_P7IOC_DIAG_TYPE_CI         = 3,
     OPAL_P7IOC_DIAG_TYPE_MISC       = 4,
     OPAL_P7IOC_DIAG_TYPE_I2C        = 5,
     OPAL_P7IOC_DIAG_TYPE_LAST       = 6
};

struct OpalIoP7IOCErrorData {
     __be16 type;

     /* GEM */
     __be64 gemXfir;
     __be64 gemRfir;
     __be64 gemRirqfir;
     __be64 gemMask;
     __be64 gemRwof;

     /* LEM */
     __be64 lemFir;
     __be64 lemErrMask;
     __be64 lemAction0;
     __be64 lemAction1;
     __be64 lemWof;

     union {
             struct OpalIoP7IOCRgcErrorData {
                     __be64 rgcStatus;       /* 3E1C10 */
                     __be64 rgcLdcp;         /* 3E1C18 */
             }rgc;
             struct OpalIoP7IOCBiErrorData {
                     __be64 biLdcp0;         /* 3C0100, 3C0118 */
                     __be64 biLdcp1;         /* 3C0108, 3C0120 */
                     __be64 biLdcp2;         /* 3C0110, 3C0128 */
                     __be64 biFenceStatus;   /* 3C0130, 3C0130 */

                     uint8_t biDownbound;    /* BI Downbound or Upbound */
             }bi;
             struct OpalIoP7IOCCiErrorData {
                     __be64 ciPortStatus;    /* 3Dn008 */
```

(continues on next page)

(continued from previous page)

```
                        __be64 ciPortLdcp;      /* 3Dn010 */

                        uint8_t ciPort;         /* Index of CI port: 0/1 */
                }ci;
        };
};

int64_t opal_pci_get_hub_diag_data(uint64_t hub_id, void *diag_buffer, uint64_t diag_
↪buffer_len);
```

Fetch diagnostic data for an IO hub. Currently, this is only implemented for p7ioc, which is specific to POWER7, something that was only ever available internally to IBM for development purposes.

If *OPAL_PCI_NEXT_ERROR* error type is *OPAL_EEH_IOC_ERROR* and severity is *OPAL_EEH_SEV_INF*, then the OS should call *OPAL_PCI_GET_HUB_DIAG_DATA* to retreive diagnostic data to log appropriately.

Returns

OPAL_SUCCESS Diagnostic data copied successfully

OPAL_PARAMETER Invalid address, invalid hub ID, or insufficient space in buffer for diagnostic data.

OPAL_UNSUPPORTED hub doesn't support retreiving diagnostic data.

OPAL_CLOSED No pending error.

OPAL_INTERNAL_ERROR Something went wrong.

4.4.55 OPAL_PCI_GET_PHB_DIAG_DATA2

```
#define OPAL_PCI_GET_PHB_DIAG_DATA2             64

/**
 * This structure defines the overlay which will be used to store PHB error
 * data upon request.
 */
 enum {
     OPAL_PHB_ERROR_DATA_VERSION_1 = 1,
 };

enum {
     OPAL_PHB_ERROR_DATA_TYPE_P7IOC = 1,
     OPAL_PHB_ERROR_DATA_TYPE_PHB3 = 2,
     OPAL_PHB_ERROR_DATA_TYPE_PHB4 = 3
};

enum {
     OPAL_P7IOC_NUM_PEST_REGS = 128,
     OPAL_PHB3_NUM_PEST_REGS = 256,
     OPAL_PHB4_NUM_PEST_REGS = 512
};

struct OpalIoPhbErrorCommon {
     __be32 version;
     __be32 ioType;
```

(continues on next page)

(continued from previous page)

```
     __be32 len;
};

struct OpalIoP7IOCPhbErrorData {
     struct OpalIoPhbErrorCommon common;

     __be32 brdgCtl;

     // P7IOC utl regs
     __be32 portStatusReg;
     __be32 rootCmplxStatus;
     __be32 busAgentStatus;

     // P7IOC cfg regs
     __be32 deviceStatus;
     __be32 slotStatus;
     __be32 linkStatus;
     __be32 devCmdStatus;
     __be32 devSecStatus;

     // cfg AER regs
     __be32 rootErrorStatus;
     __be32 uncorrErrorStatus;
     __be32 corrErrorStatus;
     __be32 tlpHdr1;
     __be32 tlpHdr2;
     __be32 tlpHdr3;
     __be32 tlpHdr4;
     __be32 sourceId;

     __be32 rsv3;

     // Record data about the call to allocate a buffer.
     __be64 errorClass;
     __be64 correlator;

     //P7IOC MMIO Error Regs
     __be64 p7iocPlssr;                // n120
     __be64 p7iocCsr;                  // n110
     __be64 lemFir;                    // nC00
     __be64 lemErrorMask;              // nC18
     __be64 lemWOF;                    // nC40
     __be64 phbErrorStatus;            // nC80
     __be64 phbFirstErrorStatus;       // nC88
     __be64 phbErrorLog0;              // nCC0
     __be64 phbErrorLog1;              // nCC8
     __be64 mmioErrorStatus;           // nD00
     __be64 mmioFirstErrorStatus;      // nD08
     __be64 mmioErrorLog0;             // nD40
     __be64 mmioErrorLog1;             // nD48
     __be64 dma0ErrorStatus;           // nD80
     __be64 dma0FirstErrorStatus;      // nD88
     __be64 dma0ErrorLog0;             // nDC0
     __be64 dma0ErrorLog1;             // nDC8
     __be64 dma1ErrorStatus;           // nE00
     __be64 dma1FirstErrorStatus;      // nE08
     __be64 dma1ErrorLog0;             // nE40
```

(continues on next page)

(continued from previous page)

```
    __be64 dma1ErrorLog1;              // nE48
    __be64 pestA[OPAL_P7IOC_NUM_PEST_REGS];
    __be64 pestB[OPAL_P7IOC_NUM_PEST_REGS];
};

struct OpalIoPhb3ErrorData {
    struct OpalIoPhbErrorCommon common;

    __be32 brdgCtl;

    /* PHB3 UTL regs */
    __be32 portStatusReg;
    __be32 rootCmplxStatus;
    __be32 busAgentStatus;

    /* PHB3 cfg regs */
    __be32 deviceStatus;
    __be32 slotStatus;
    __be32 linkStatus;
    __be32 devCmdStatus;
    __be32 devSecStatus;

    /* cfg AER regs */
    __be32 rootErrorStatus;
    __be32 uncorrErrorStatus;
    __be32 corrErrorStatus;
    __be32 tlpHdr1;
    __be32 tlpHdr2;
    __be32 tlpHdr3;
    __be32 tlpHdr4;
    __be32 sourceId;

    __be32 rsv3;

    /* Record data about the call to allocate a buffer */
    __be64 errorClass;
    __be64 correlator;

    /* PHB3 MMIO Error Regs */
    __be64 nFir;                       /* 000 */
    __be64 nFirMask;                   /* 003 */
    __be64 nFirWOF;            /* 008 */
    __be64 phbPlssr;                   /* 120 */
    __be64 phbCsr;             /* 110 */
    __be64 lemFir;             /* C00 */
    __be64 lemErrorMask;               /* C18 */
    __be64 lemWOF;             /* C40 */
    __be64 phbErrorStatus;     /* C80 */
    __be64 phbFirstErrorStatus;        /* C88 */
    __be64 phbErrorLog0;               /* CC0 */
    __be64 phbErrorLog1;               /* CC8 */
    __be64 mmioErrorStatus; /* D00 */
    __be64 mmioFirstErrorStatus;       /* D08 */
    __be64 mmioErrorLog0;              /* D40 */
    __be64 mmioErrorLog1;              /* D48 */
    __be64 dma0ErrorStatus; /* D80 */
    __be64 dma0FirstErrorStatus;       /* D88 */
```

(continues on next page)

(continued from previous page)

```
	__be64 dma0ErrorLog0;		/* DC0 */
	__be64 dma0ErrorLog1;		/* DC8 */
	__be64 dma1ErrorStatus;	/* E00 */
	__be64 dma1FirstErrorStatus;	/* E08 */
	__be64 dma1ErrorLog0;		/* E40 */
	__be64 dma1ErrorLog1;		/* E48 */
	__be64 pestA[OPAL_PHB3_NUM_PEST_REGS];
	__be64 pestB[OPAL_PHB3_NUM_PEST_REGS];
};

struct OpalIoPhb4ErrorData {
	struct OpalIoPhbErrorCommon common;

	__be32 brdgCtl;

	/* XXX missing UTL registers? */

	/* PHB4 cfg regs */
	__be32 deviceStatus;
	__be32 slotStatus;
	__be32 linkStatus;
	__be32 devCmdStatus;
	__be32 devSecStatus;

	/* cfg AER regs */
	__be32 rootErrorStatus;
	__be32 uncorrErrorStatus;
	__be32 corrErrorStatus;
	__be32 tlpHdr1;
	__be32 tlpHdr2;
	__be32 tlpHdr3;
	__be32 tlpHdr4;
	__be32 sourceId;

	/* PHB4 ETU Error Regs */
	__be64 nFir;				/* 000 */
	__be64 nFirMask;			/* 003 */
	__be64 nFirWOF;			/* 008 */
	__be64 phbPlssr;			/* 120 */
	__be64 phbCsr;				/* 110 */
	__be64 lemFir;				/* C00 */
	__be64 lemErrorMask;			/* C18 */
	__be64 lemWOF;				/* C40 */
	__be64 phbErrorStatus;			/* C80 */
	__be64 phbFirstErrorStatus;		/* C88 */
	__be64 phbErrorLog0;			/* CC0 */
	__be64 phbErrorLog1;			/* CC8 */
	__be64 phbTxeErrorStatus;		/* D00 */
	__be64 phbTxeFirstErrorStatus;		/* D08 */
	__be64 phbTxeErrorLog0;		/* D40 */
	__be64 phbTxeErrorLog1;		/* D48 */
	__be64 phbRxeArbErrorStatus;		/* D80 */
	__be64 phbRxeArbFirstErrorStatus;	/* D88 */
	__be64 phbRxeArbErrorLog0;		/* DC0 */
	__be64 phbRxeArbErrorLog1;		/* DC8 */
	__be64 phbRxeMrgErrorStatus;		/* E00 */
	__be64 phbRxeMrgFirstErrorStatus;	/* E08 */
```

(continues on next page)

(continued from previous page)

```
        __be64 phbRxeMrgErrorLog0;                /* E40 */
        __be64 phbRxeMrgErrorLog1;                /* E48 */
        __be64 phbRxeTceErrorStatus;              /* E80 */
        __be64 phbRxeTceFirstErrorStatus;         /* E88 */
        __be64 phbRxeTceErrorLog0;                /* EC0 */
        __be64 phbRxeTceErrorLog1;                /* EC8 */

        /* PHB4 REGB Error Regs */
        __be64 phbPblErrorStatus;                 /* 1900 */
        __be64 phbPblFirstErrorStatus;            /* 1908 */
        __be64 phbPblErrorLog0;                   /* 1940 */
        __be64 phbPblErrorLog1;                   /* 1948 */
        __be64 phbPcieDlpErrorLog1;               /* 1AA0 */
        __be64 phbPcieDlpErrorLog2;               /* 1AA8 */
        __be64 phbPcieDlpErrorStatus;             /* 1AB0 */
        __be64 phbRegbErrorStatus;                /* 1C00 */
        __be64 phbRegbFirstErrorStatus;           /* 1C08 */
        __be64 phbRegbErrorLog0;                  /* 1C40 */
        __be64 phbRegbErrorLog1;                  /* 1C48 */

        __be64 pestA[OPAL_PHB4_NUM_PEST_REGS];
        __be64 pestB[OPAL_PHB4_NUM_PEST_REGS];
};

int64_t opal_pci_get_phb_diag_data2(uint64_t phb_id, void *diag_buffer, uint64_t diag_
↪buffer_len);
```

Get PCI diagnostic data from a given PHB. Each PHB present in the device tree has a `ibm,phb-diag-data-size` property which is the size of the diagnostic data structure that can be returned.

Each PHB generation has a different structure for diagnostic data, and the small common structure will allow the OS to work out what format the data is coming in.

In future, it's possible that the format will change to be more flexible, and require less OS support.

Parameters

`uint64_t phb_id` the ID of the PHB you want to retrieve data from

`void *diag_buffer` an allocated buffer to store diag data in

`uint64_t diag_buffer_len` size in bytes of the diag buffer

Calling

Retrieve the PHB's diagnostic data. The diagnostic data is stored in the buffer pointed by @diag_buffer. Different PHB versions will store different diagnostics, defined in include/opal-api.h as `struct OpalIo<PHBVer>ErrorData`.

OPAL_PCI_GET_PHB_DIAG_DATA is deprecated and *OPAL_PCI_GET_PHB_DIAG_DATA2* should be used instead.

Return Codes

OPAL_SUCCESS Diagnostic data has been retrieved and stored successfully

OPAL_PARAMETER The given buffer is too small to store the diagnostic data

OPAL_HARDWARE The PHB is in a broken state and its data cannot be retreived

OPAL_UNSUPPORTED Diagnostic data is not implemented for this PHB type

4.4.56 OPAL_PCI_GET_POWER_STATE

```
#define OPAL_PCI_GET_POWER_STATE                120

int64_t opal_pci_get_power_state(uint64_t id, uint64_t data);
```

Get PCI slot power state

Parameter

`uint64_t id` PCI slot ID

`uint64_t data` memory buffer pointer for power state

Calling

Retrieve PCI slot's power state. The retrieved power state is stored in buffer pointed by @data.

Return Codes

OPAL_SUCCESS PCI slot's power state is retrieved successfully

OPAL_PARAMETER The indicated PCI slot isn't found

OPAL_UNSUPPORTED Power state retrieval not supported on the PCI slot

4.4.57 OPAL_PCI_GET_PRESENCE_STATE

Get PCI slot presence state

Parameters

`uint64_t id` PCI slot ID

`uint64_t data` memory buffer pointer for presence state

Calling

Retrieve PCI slot's presence state. The detected presence means there are adapters inserted to the PCI slot. Otherwise, the PCI slot is regarded as an empty one. The typical use is to ensure there are adapters existing before probing the PCI slot in PCI hot add path. The retrieved presence state is stored in buffer pointed by @data.

Return Codes

OPAL_SUCCESS PCI slot's presence state is retrieved successfully

OPAL_PARAMETER The indicated PCI slot isn't found

OPAL_UNSUPPORTED Presence retrieval not supported on the PCI slot

4.4.58 OPAL_PCI_GET_PBCQ_TUNNEL_BAR

```
#define OPAL_PCI_GET_PBCQ_TUNNEL_BAR 164

int64_t opal_pci_get_pbcq_tunnel_bar(uint64_t phb_id, uint64_t *addr);
```

The host calls this function to read the address out of the PBCQ Tunnel Bar register.

Parameters

phb_id The value from the PHB node ibm,opal-phbid property for the device.

addr A pointer to where the address stored in the PBCQ Tunnel Bar register will be copied.

Return Values

OPAL_SUCCESS Operation was successful

OPAL_PARAMETER Invalid PHB or addr parameter

OPAL_UNSUPPORTED Not supported by hardware

4.4.59 OPAL_PCI_SET_PBCQ_TUNNEL_BAR

```
#define OPAL_PCI_SET_PBCQ_TUNNEL_BAR 165

int64_t opal_pci_set_pbcq_tunnel_bar(uint64_t phb_id, uint64_t addr);
```

The host calls this function to set the PBCQ Tunnel Bar register.

Parameters

phb_id The value from the PHB node ibm,opal-phbid property for the device.

addr The value of the address chosen for the PBCQ Tunnel Bar register. If the address is 0, then the PBCQ Tunnel Bar register will be reset. It the address is non-zero, then the PBCQ Tunnel Bar register will be set with

```
Bit[0:42]      Bit[8:50] of the address
```

Return Values

OPAL_SUCCESS Operation was successful

OPAL_PARAMETER Invalid PHB or addr parameter

OPAL_UNSUPPORTED Not supported by hardware

4.4.60 OPAL_PCI_MAP_PE_DMA_WINDOW

```
#define OPAL_PCI_MAP_PE_DMA_WINDOW          44

static int64_t opal_pci_map_pe_dma_window(uint64_t phb_id,
                                          uint64_t pe_number,
                                          uint16_t window_id,
                                          uint16_t tce_levels,
                                          uint64_t tce_table_addr,
                                          uint64_t tce_table_size,
                                          uint64_t tce_page_size);
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to create a DMA window and map it to a PE. This call returns the address in PCI memory that corresponds to the specified DMA window, which in part may depend on the particular PHB DMA window used. An address that is all zeros in the upper 32 bits reflects a DMA window enabled for 32-bit DMA addresses.

The overall size of the DMA window in PCI memory is determined by the number of tce_levels times the tce_table_size times the tce_page_size.

`phb_id` is the value from the PHB node `ibm,opal-phbid` property.

`dma_window_number` specifies the DMA window

For ibm,opal-ioda PHBs the dma_window_number is an index from 0 to the PHB total number of windows minus 1. For ibm,opal-ioda2 PHBs the DMA window_number is an index from 0 to n-1, where n is the number of windows per window set, within the window set associated with the specified PE number.

`pe_number` is the index of the PE that is authorized to DMA to this window address space in PCI memory,

`tce_levels` is the number of TCE table levels in the translation hiearchy, from 1 to ibm,opal-dmawins property <translation levels>.

`tce_table_addr` is the 64-bit system real address of the first level (root, for mult-level) TCE table in the translation hiearchy.

`tce_table_size` is the size, in bytes, of each TCE table in the translation hierarchy. A value of '0' indicates to disable this DMA window.

For ibm,opal-ioda, this must be a value in the range from 128MB / tce_page_size to 256TB / tce_page_size, and must be in the format and matching a value in the tce_table ranges property that is minimally 256KB for 4K pages.

A particular PE may be mapped to multiple DMA windows, each spanning a DMA window size corresponding to the win_size32 or win_size_64 specified in the ibm,opal-dmawins<> property. However, the TCE table base address must be unique for each window unless it is intended that the same page address in each DMA window is mapped through the same TCE table entry. Generally, when mapping the same PE to multiple DMA windows, so as to create a larger overall DMA window, it is recommended to use consecutive DMA windows and each DMA window should use a TCE table address that is offset by the win_size value of predecessor DMA window.

tce_page_size is the size of PCI memory pages mapped to system real pages through all TCE tables in the translation hierarchy. This must be the same format as and match a value from the ibm,opal-dmawins property <dma-page-sizes>. This page size applies to all TCE tables in the translation hierarchy.

pci_start_addr returns the starting address in PCI memory that corresponds to this DMA window based on the input translation parameter values.

pci_mem_type selects whether this DMA window should be created in 32-bit or 64-bit PCI memory. The input values correspond to the same PCI memory space locators as MMIO spaces in the ranges<> property – 0x2 indicated 32-bit PCI memory and 0x3 indicates 64-bit memory.

Window 0 for both ibm,opal-ioda and ibm,opal-ioda2 PHBs must be within 32-bit PCI memory and this call return opal_parameter for calls that specify window 0 in 64-bit PCI memory.

The DMA win_size property for 32 bit DMA windows limits the number of ibm,opal-ioda PHB windows that can map32-bit address space. For example, with a win_size_32 = 256MB, only 16 DMA windows (and therefore no more than 16 distinct PEs) can map the 4GB of 32-bit PCI memory for DMA. OPAL does not police this limitation.

Return value:

```
if (!phb)
        return OPAL_PARAMETER;
if (!phb->ops->map_pe_dma_window)
        return OPAL_UNSUPPORTED;
```

4.4.61 OPAL_PCI_MAP_PE_DMA_WINDOW_REAL

```
#define OPAL_PCI_MAP_PE_DMA_WINDOW_REAL              45

int64_t opal_pci_map_pe_dma_window_real(uint64_t phb_id,
                                        uint64_t pe_number,
                                        uint16_t window_id,
                                        uint64_t pci_start_addr,
                                        uint64_t pci_mem_size);
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to initialize the specified DMA window for untranslated DMA addresses. This allows a PE to DMA directly to system memory without TCE translation. The DMA window PCI memory address is equal to the system memory real address. The PHB passes PCI address bits 04:63 directly to system real address bits 04:63 when PCI address bits 04:39 are within the region specified by mem_addr t0 mem_addr + window_size.

The addresses must be 16MB aligned and a multiple of 16MB in size.

phb_id is the value from the PHB node ibm,opal-phbid property.

dma_window_number specifies the DMA window

For ibm,opal-ioda PHBs the dma_window_number is an index from 0 to the PHB total number of windows minus 1. For ibm,opal-ioda2 PHBs the DMA window_number is an index from 0 to n-1, where n is the number of windows per window set, within the window set associated with the specified PE number.

pe_number is the index of the PE that is authorized to DMA to this window address space in PCI memory,

mem_addr is the starting 64-bit system real address mapped directly to the starting address in PCI memory. Addresses below 4GB are zero in bits above bit 32. This value must be aligned on a 16MB boundary; OPAL returns OPAL_PARAMETER for any value that is not a multiple of 16MB.

window_size is the size, in bytes, of the address range defined by this window. This value must be a multiple of 16MB; OPAL returns *OPAL_PARAMETER* for any value that is not a multiple of 16MB. A value of '0' indicates to disable this DMA window.

4.4.62 OPAL_PCI_MAP_PE_MMIO_WINDOW

```
#define OPAL_PCI_MAP_PE_MMIO_WINDOW          29

int64_t opal_pci_map_pe_mmio_window(uint64_t phb_id,
                                    uint64_t pe_number,
                                    uint16_t window_type,
                                    uint16_t window_num,
                                    uint16_t segment_num);
```

Note: Appears to be POWER7 p7ioc specific. Likely to be removed soon.

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to map a segment of MMIO address space to a PE.

phb_id is the value from the PHB node ibm,opal-phbid property.

window_type specifies 32-bit or 64-bit PCI memory

'0' selects PCI IO Space. ibm,opal-ioda2 PHBs do not support IO space, and OPAL returns opal_unsupported if called for IO windows.

'1' selects 32-bit PCI memory space

'2' selects 64 bit PCI memory space

window_num is the MMIO window number within the specified PCI memory space

segment_num is an index from 0 to the number of segments minus 1 defined or this window, and selects a particular segment within the specified window.

Return value:

```
if (!phb)
        return OPAL_PARAMETER;
if (!phb->ops->map_pe_mmio_window)
        return OPAL_UNSUPPORTED;
```

4.4.63 OPAL_PCI_MSI_EOI

```
#define OPAL_PCI_MSI_EOI                     63

int64_t opal_pci_msi_eoi(uint64_t phb_id, uint32_t hwirq);
```

Only required on PHB3 (POWER8) based systems.

Returns

OPAL_SUCCESS Success!

OPAL_PARAMETER Invalid PHB id or hwirq.

OPAL_HARDWARE Hardware or configuration issue.

OPAL_UNSUPPORTED Unsupported on this PHB.

4.4.64 OPAL_PCI_NEXT_ERROR

```
#define OPAL_PCI_NEXT_ERROR                    60

enum OpalPciStatusToken {
     OPAL_EEH_NO_ERROR        = 0,
     OPAL_EEH_IOC_ERROR       = 1,
     OPAL_EEH_PHB_ERROR       = 2,
     OPAL_EEH_PE_ERROR        = 3,
     OPAL_EEH_PE_MMIO_ERROR   = 4,
     OPAL_EEH_PE_DMA_ERROR    = 5
};

enum OpalPciErrorSeverity {
     OPAL_EEH_SEV_NO_ERROR    = 0,
     OPAL_EEH_SEV_IOC_DEAD    = 1,
     OPAL_EEH_SEV_PHB_DEAD    = 2,
     OPAL_EEH_SEV_PHB_FENCED  = 3,
     OPAL_EEH_SEV_PE_ER       = 4,
     OPAL_EEH_SEV_INF         = 5
};

int64_t opal_pci_next_error(uint64_t phb_id, uint64_t *first_frozen_pe,
                            uint16_t *pci_error_type, uint16_t *severity);
```

Retreives details of a PCIe error.

Returns

OPAL_SUCCESS Successfully filled *pci_error_type* and *severity* with error details.

OPAL_UNSUPPORTED Unsupported operation on this PHB.

OPAL_PARAMETER Invalid phb_id, or address for other arguments.

4.4.65 OPAL_PCI_PHB_MMIO_ENABLE

```
#define OPAL_PCI_PHB_MMIO_ENABLE               27

int64_t opal_pci_phb_mmio_enable(uint64_t phb_id, uint16_t window_type,
                                 uint16_t window_num, uint16_t enable);
```

Note: Appears to be POWER7 p7ioc specific. Likely to be removed soon.

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to enable or disable PHB decode of the PCI IO and Memory address spaces below that PHB. Window_num selects an mmio window within that address space. Enable set to '1' enables the PHB to decode and forward system real addresses to PCI memory, while enable set to '0' disables PHB decode and forwarding for the address range defined in a particular MMIO window.

Not all PHB hardware may support disabling some or all MMIO windows. OPAL returns *OPAL_UNSUPPORTED* if called to disable an MMIO window for which hardware does not support disable. KVM may call this function for all MMIO windows and ignore the opal_unsuppsorted return code so long as KVM has disabled MMIO to all downstream PCI devices and assured that KVM and OS guest partitions cannot issue CI loads/stores to these address spaces from the processor (e.g.,via HPT).

OPAL returns *OPAL_SUCCESS* for calls to OPAL to enable them for PHBs that do not support disable.

phb_id is the value from the PHB node ibm,opal-phbid property.

window_type specifies 32-bit or 64-bit PCI memory

> '0' selects PCI IO Space
>
> '1' selects 32-bit PCI memory space
>
> '2' selects 64 bit PCI memory space

window_num is the MMIO window number within the specified PCI memory space

enable specifies to enable or disable this MMIO window.

4.4.66 OPAL_PCI_POLL

```
#define OPAL_PCI_POLL                                     62

int64_t opal_pci_poll(uint64_t id);
```

Crank the state machine for the PHB id. Returns how many milliseconds for the caller to sleep.

Returns

Milliseconds for the caller to sleep for, error code, or *OPAL_SUCCESS*.

4.4.67 OPAL_PCI_REINIT

```
#define OPAL_PCI_REINIT                                   53

enum OpalPciReinitScope {
     /*
      * Note: we chose values that do not overlap
      * OpalPciResetScope as OPAL v2 used the same
      * enum for both
      */
     OPAL_REINIT_PCI_DEV = 1000
};

int64_t opal_pci_reinit(uint64_t phb_id, uint64_t reinit_scope, uint64_t data);
```

Note: Much glory awaits the one who fills in this documentation.

Returns

OPAL_PARAMETER Invalid PHB, scope, or device.

OPAL_UNSUPPORTED Operation unsupported

OPAL_HARDWARE Some hardware issue prevented the reinit.

4.4.68 OPAL_PCI_RESET

```
#define OPAL_PCI_RESET                              49

enum OpalPciResetScope {
     OPAL_RESET_PHB_COMPLETE         = 1,
     OPAL_RESET_PCI_LINK             = 2,
     OPAL_RESET_PHB_ERROR            = 3,
     OPAL_RESET_PCI_HOT              = 4,
     OPAL_RESET_PCI_FUNDAMENTAL      = 5,
     OPAL_RESET_PCI_IODA_TABLE       = 6
};

enum OpalPciResetState {
     OPAL_DEASSERT_RESET = 0,
     OPAL_ASSERT_RESET   = 1
};

int64_t opal_pci_reset(uint64_t id, uint8_t reset_scope, uint8_t assert_state);
```

Kick off the requested PCI reset operation. This starts a state machine off to perform the requested operation. This call will return how many milliseconds to wait before calling back into *OPAL_PCI_POLL*. An OS can call *OPAL_PCI_POLL* earlier, but it is unlikely any progress will have been made.

Returns

OPAL_PARAMETER Invalid `id`, `reset_scope`, or `assert_state`.

OPAL_UNSUPPORTED Operation is unsupported on `id`.

value > 0 How many ms to wait for the state machine to crank. Call *OPAL_PCI_POLL* to crank the state machine further.

4.4.69 OPAL_PCI_SET_MVE

```
#define OPAL_PCI_SET_MVE                        33

int64_t opal_pci_set_mve(uint64_t phb_id, uint32_t mve_number, uint64_t pe_number);
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to bind a PE to an MSI Validation Table Entry (MVE) in the PHB. The MVE compares the MSI requester (RID) to a PE RID, including within the XIVE, to validate that the requester is authorized to signal an interrupt to the associated DMA address for a message value that selects a particular XIVE.

`phb_id` is the value from the PHB node ibm,opal-phbid property.

mve_number is the index, from 0 to ibm,opal,ibm-num-msi-ports minus1

pe_number is the index of a PE, from 0 to ibm,opal-num-pes minus 1.

This call maps an MVE to a PE and PE RID domain. OPAL uses the PELT to determine the PE domain. OPAL treats this call as a NOP for IODA2 PHBs and returns a status of OPAL_SUCCESS.

Return value:

```
if (!phb)
        return OPAL_PARAMETER;
if (!phb->ops->set_mve)
        return OPAL_UNSUPPORTED;
```

4.4.70 OPAL_PCI_SET_MVE_ENABLE

```
#define OPAL_PCI_SET_MVE_ENABLE                       34

int64_t opal_pci_set_mve_enable(uint64_t phb_id, uint32_t mve_number, uint32_t state);

enum OpalMveEnableAction {
     OPAL_DISABLE_MVE = 0,
     OPAL_ENABLE_MVE = 1
};
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to enable or disable an MVE to respond to an MSI DMA address and message data value.

phb_id is the value from the PHB node ibm,opal-phbid property.

mve_number is the index, from 0 to ibm,opal,ibm-num-msi-ports minus1

state A ‘1’ value of the state parameter indicates to enable the MVE and a ‘0’ value indicates to disable the MVE.

This call sets the MVE to an enabled (1) or disabled (0) state.

Return value:

```
if (!phb)
        return OPAL_PARAMETER;
if (!phb->ops->set_mve_enable)
        return OPAL_UNSUPPORTED;
```

4.4.71 OPAL_PCI_SET_P2P

```
#define OPAL_PCI_SET_P2P                         157

int64_t opal_pci_set_p2p(uint64_t phbid_init, uint64_t phbid_target,
                              uint64_t desc, uint16_t pe_number);

/* PCI p2p descriptor */
#define OPAL_PCI_P2P_ENABLE           0x1
#define OPAL_PCI_P2P_LOAD             0x2
#define OPAL_PCI_P2P_STORE            0x4
```

The host calls this function to enable PCI peer-to-peer on the PHBs.

Parameters

phbid_init is the value from the PHB node ibm,opal-phbid property for the device initiating the p2p operation

phbid_target is the value from the PHB node ibm,opal-phbid property for the device targeted by the p2p operation

desc tells whether the p2p operation is a store (OPAL_PCI_P2P_STORE) or load (OPAL_PCI_P2P_LOAD). Can be both. OPAL_PCI_P2P_ENABLE enables/disables the setting

pe_number PE number for the initiating device

Return Values

OPAL_SUCCESS Configuration was successful

OPAL_PARAMETER Invalid PHB or mode parameter

OPAL_UNSUPPORTED Not supported by hardware

4.4.72 OPAL_PCI_SET_PE

```
#define OPAL_PCI_SET_PE                          31

int64_t opal_pci_set_pe(uint64_t phb_id, uint64_t pe_number,
                        uint64_t bus_dev_func, uint8_t bus_compare,
                        uint8_t dev_compare, uint8_t func_compare,
                        uint8_t pe_action);
```

NOTE: The following two paragraphs come from some old documentation and have not been checked for accuracy. Same goes for bus_compare, dev_compare and func_compare documentation. Do *NOT* assume this documentation is correct without checking the source.

A host OS calls this function to map a PCIE function (RID), or range of function bus/dev/funcs (RIDs), to a PHB PE. The bus, device, func, and compare parameters define a range of bus, device, or function numbers to define a range of RIDs within this domain. A value of "7" for the bus_compare, and non-zero for the dev_compare and func_compare, define exactly one function RID to be a PE (within a PE number domain).

This must be called prior to ALL other OPAL calls that take a PE number argument, for OPAL to correlate the RID (bus/dev/func) domain of the PE. If a PE domain is changed, the host must call this to reset the PE bus/dev/func domain and then call all other OPAL calls that map PHB IODA resources to update those domains within PHB facilities.

phb_id is the value from the PHB node ibm,opal-phbid property.

pe_number is the index of a PE, from 0 to ibm,opal-num-pes minus 1.

bus_compare is a value from 0 to 7 indicating which bus number bits define the range of buses in a PE domain:

0 = do not validate against RID bus number (PE = all bus numbers)

2 = compare high order 3 bits of RID bus number to high order 3 bits of PE bus number

3 = compare high order 4 bits of RID bus number to high order 4 bits of PE bus number

6 = compare high order 7 bits of RID bus number to high order 7 bits of PE bus number

7 = compare all bits of RID bus number to all bits of PE bus number

dev_compare indicates to compare the RID device number to the PE device number or not. '0' signifies that the RID device number is not compared – essentially all device numbers within the bus and function number range of this PE are also within this PE. Non-zero signifies to compare the RID device number to the PE device number, such that only that device number is in the PE domain, for all buses and function numbers in the PE domain.

func_compare indicates to compare the RID function number to the PE function number or not. '0' signifies that the RID function number is not compared – essentially all function numbers within the bus and device number range of this PE are also within this PE. Non-zero signifies to compare the RID function number to the PE function number, such that only that function number is in the PE domain, for all buses and device numbers in the PE domain.

pe_action is one of:

```
enum OpalPeAction {
    OPAL_UNMAP_PE = 0,
    OPAL_MAP_PE = 1
};
```

Returns

OPAL_PARAMETER If one of the following:

- invalid phb
- invalid pe_action
- invalid bus_dev_func
- invalid bus_compare

OPAL_UNSUPPORTED PHB does not support set_pe operation

OPAL_SUCCESS if operation was successful

4.4.73 OPAL_PCI_SET_PELTV

```
#define OPAL_PCI_SET_PELTV                     32

int64_t opal_pci_set_peltv(uint64_t phb_id, uint32_t parent_pe,
                           uint32_t child_pe, uint8_t state);
```

WARNING: This documentation comes from an old source and is possibly not up to date with OPALv3. Rely on this documentation only as a starting point, use the source (and update the docs).

This call sets the PELTV of a parent PE to add or remove a PE number as a PE within that parent PE domain. The host must call this function for each child of a parent PE.

phb_id is the value from the PHB node ibm,opal-phbid property

parent_pe is the PE number of a PE that is higher in the PCI hierarchy to other PEs, such that an error involving this parent PE should cause a collateral PE freeze for PEs below this PE in the PCI hierarchy. For example a switch upstream bridge is a PE that is parent to PEs reached through that upstream bridge such that an error involving the upstream bridge (e.g, ERR_FATAL) should cause the PHB to freeze all other PEs below that upstream bridge (e.g., a downstream bridge, or devices below a downstream bridge).

child_pe is the PE number of a PE that is lower in the PCI hierarchy than another PE, such that an error involving that other PE should cause a collateral PE freeze for this child PE. For example a device below a downstream bridge of a PCIE switch is a child PE that downstream bridge PE and the upstream bridge PE of that switch – an ERR_Fatal from either bridge should result in a collateral freeze of that device PE.

```
enum OpalPeltvAction {
     OPAL_REMOVE_PE_FROM_DOMAIN = 0,
     OPAL_ADD_PE_TO_DOMAIN = 1
};
```

OPAL Implementation Note: WARNING TODO: *CHECK IF THIS IS CORRECT FOR skiboot:* For ibm,opal-ioda2, OPAL sets the PELTV bit in all RTT entries for the parent PE when the state argument is '1'. OPAL clears the PELTV bit in all RTT entries for the parent PE when the state argument is '0' and setting the child PE bit in the parent PELTV results in an all-zeros value for that PELTV.

Return value:

```
if (!phb)
        return OPAL_PARAMETER;
if (!phb->ops->set_peltv)
        return OPAL_UNSUPPORTED;
```

4.4.74 OPAL_PCI_SET_PHB_CAPI_MODE

```
#define OPAL_PCI_SET_PHB_CAPI_MODE           93

/* CAPI modes for PHB */
enum {
  OPAL_PHB_CAPI_MODE_PCIE             = 0,
  OPAL_PHB_CAPI_MODE_CAPI             = 1,
  OPAL_PHB_CAPI_MODE_SNOOP_OFF    = 2,
  OPAL_PHB_CAPI_MODE_SNOOP_ON         = 3,
  OPAL_PHB_CAPI_MODE_DMA              = 4,
  OPAL_PHB_CAPI_MODE_DMA_TVT1         = 5,
};

int64_t opal_pci_set_phb_capi_mode(uint64_t phb_id, uint64_t mode, uint64_t pe_
↪number);
```

Switch the CAPP attached to the given PHB in one of the supported CAPI modes.

Parameters

`uint64_t phb_id` the ID of the PHB which identifies attached CAPP to perform mode switch on

`uint64_t mode` A mode id as described below

`pe_number` PE number for the initiating device

Calling

Switch CAPP attached to the given PHB in one of the following supported modes:

```
OPAL_PHB_CAPI_MODE_PCIE             = 0
OPAL_PHB_CAPI_MODE_CAPI             = 1
OPAL_PHB_CAPI_MODE_SNOOP_OFF    = 2
```

(continues on next page)

(continued from previous page)

```
OPAL_PHB_CAPI_MODE_SNOOP_ON    = 3
OPAL_PHB_CAPI_MODE_DMA                  = 4
OPAL_PHB_CAPI_MODE_DMA_TVT1    = 5
```

Modes *OPAL_PHB_CAPI_MODE_PCIE* and *OPAL_PHB_CAPI_MODE_CAPI* are used to enable/disable CAPP attached to the PHB.

Modes *OPAL_PHB_CAPI_MODE_SNOOP_OFF* and *OPAL_PHB_CAPI_MODE_SNOOP_ON* are used to enable/disable CAPP snooping of Powerbus traffic for cache line invalidates.

Mode *OPAL_PHB_CAPI_MODE_DMA* and *OPAL_PHB_CAPI_MODE_DMA_TVT1* are used to enable CAPP DMA mode.

Presently Mode *OPAL_PHB_CAPI_MODE_DMA_TVT1* is exclusively used by the Mellanox CX5 adapter. Requesting this mode will also indicate to opal that the card requests maximum number of DMA read engines allocated to improve DMA read performance at cost of reduced bandwidth available to other traffic including CAPP-PSL transactions.

Notes

- If PHB is in PEC2 then requesting mode *OPAL_PHB_CAPI_MODE_DMA_TVT1* will allocate extra 16/8 dma read engines to the PHB depending on its stack (stack 0/ stack 1). This is needed to improve the Direct-GPU DMA read performance for the Mellanox CX5 card.
- Mode *OPAL_PHB_CAPI_MODE_PCIE* not yet supported on Power-9.
- Requesting mode *OPAL_PHB_CAPI_MODE_CAPI* on Power-9 will disable fast-reboot.
- Modes *OPAL_PHB_CAPI_MODE_DMA*, *OPAL_PHB_CAPI_MODE_SNOOP_OFF* are not supported on Power-9 yet.

Return Codes

OPAL_SUCCESS Switch to the requested capi mode performed successfully.

OPAL_PARAMETER The requested value of mode or phb_id parameter is not valid.

OPAL_HARDWARE An error occurred while switching the CAPP to requested mode.

OPAL_UNSUPPORTED Switching to requested capi mode is not possible at the moment

OPAL_RESOURCE CAPP ucode not available hence activating CAPP not supported.

OPAL_BUSY CAPP is presently in recovery-mode and mode switch cannot be performed.

4.4.75 OPAL_PCI_SET_PHB_MEM_WINDOW

```
#define OPAL_PCI_SET_PHB_MEM_WINDOW             28

int64_t opal_pci_set_phb_mem_window(uint64_t phb_id,
                                    uint16_t window_type,
                                    uint16_t window_num,
                                    uint64_t addr,
                                    uint64_t pci_addr,
                                    uint64_t size);
```

Note: Appears to be POWER7 p7ioc specific. Likely to be removed soon.

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to set the PHB PCI memory window parameters for PHBs. OPAL sets IO space for P7IOC and KVM cannot relocate this. KVM should changes these windows only while all devices below the PHB are disabled for PCI memory ops, and with the target window in disabled state (where supported by PHB hardware).

phb_id is the value from the PHB node ibm,opal-phbid property.

window_type specifies 32-bit or 64-bit PCI memory

‘0’ selects IO space, and is not supported for relocation. OPAL returns OPAL_UNSUPPORTED for this value.

‘1’ selects 32-bit PCI memory space

‘2’ selects 64 bit PCI memory space

window_num is the MMIO window number within the specified PCI memory space

starting_real_address specifies the location within sytsem (processor)real address space this MMIO window starts. This must be a location within the IO Hub or PHB node ibm,opal-mmio-real property.

starting_pci_address specifies the location within PCI 32 or 64-bit address space that this MMIO window starts. For 64-bit PCI memory, this must be within the low order 60 bit (1 Exabyte) region of PCI memory. Addresses above 1EB are reserved to IODA definitions.

segment_size defines the segment size of this window, in the same format as and a matching value from the ibm,opal-memwin32/64 <segment_size> property. The window total size, in bytes, is the segment_size times the ibm,opal-memwin32/64 <num_segments> property and must not extend beyond the ibm,opal-mmio-real property range within system real address space. The total MMIO window size is the segment_size times the num_segments supported for the specific window. The host must assure that the cumulative address space for all enabled windows does not exceed the total PHB 32-bit or 64-bit real address window space, or extend outside these address ranges, and that no windows overlap each other in real or PCI address space. OPAL does not validate those conditions.

A segment size of ‘0’ indicates to disable this MMIO window. If the PHB hardware does not support disabling a window, OPAL returns OPAL_UNSUPPORTED status.

The size of the system real and PCI memory spaces are equal and defined by segment_size times the number of segments within this MMIO window.

The host must set PHB memory windows to be within the system real address ranges indicated in the PHB parent HDT hub node ibm,opal-mmio-real property.

Return value:

```
if (!phb)
        return OPAL_PARAMETER;
if (!phb->ops->set_phb_mem_window)
        return OPAL_UNSUPPORTED;
```

4.4.76 OPAL_PCI_SET_POWER_STATE

```
#define OPAL_PCI_SET_POWER_STATE                121

int64_t opal_pci_set_power_state(uint64_t async_token, uint64_t id, uint64_t data);
```

Set PCI slot power state

Parameters

`uint64_t async_token` Token of asynchronous message to be sent on completion of OPAL_PCI_SLOT_POWER_{OFF, ON}. It is ignored when @data is OPAL_PCI_SLOT_{OFFLINE, ONLINE}.

`uint64_t id` PCI slot ID

`uint64_t data` memory buffer pointer for the power state which can be one of OPAL_PCI_SLOT_POWER_{OFF, ON, OFFLINE, ONLINE}.

Calling

Set PCI slot's power state. The power state is stored in buffer pointed by @data. The typical use is to hot add or remove adapters behind the indicated PCI slot (by @id) in PCI hotplug path.

User will receive an asychronous message after calling the API. The message contains the API completion status: event (Power off or on), device node's phandle identifying the PCI slot, errcode (e.g. *OPAL_SUCCESS*). The API returns *OPAL_ASYNC_COMPLETION* for the case.

The states OPAL_PCI_SLOT_OFFLINE and OPAL_PCI_SLOT_ONLINE are used for removing or adding devices behind the slot. The device nodes in the device tree are removed or added accordingly, without actually changing the slot's power state. The API call will return OPAL_SUCCESS immediately and no further asynchronous message will be sent.

Return Codes

OPAL_SUCCESS PCI hotplug on the slot is completed successfully

OPAL_ASYNC_COMPLETION PCI hotplug needs further message to confirm

OPAL_PARAMETER The indicated PCI slot isn't found

OPAL_UNSUPPORTED Setting power state not supported on the PCI slot

4.4.77 OPAL_PCI_SET_XIVE_PE

```
#define OPAL_PCI_SET_XIVE_PE                    37

int64_t opal_pci_set_xive_pe(uint64_t phb_id, uint64_t pe_number, uint32_t xive_num);
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to bind a PE to an XIVE. Only that PE may then signal an MSI that selects this XIVE.

`phb_id` is the value from the PHB node ibm,opal-phbid property.

`pe_number` is the index of a PE, from 0 to ibm,opal-num-pes minus 1.

`xive_number` is the index, from 0 to ibm,opal,ibm-num-msis minus (num_lsis+1)

This call maps the XIVR indexed by xive_num to the PE specified by pe_number. For ibm,opal-ioda HW, the pe_number must match the pe_number set in the MVE.

Return value:

```
if (!phb)
        return OPAL_PARAMETER;
if (!phb->ops->set_xive_pe)
        return OPAL_UNSUPPORTED;
```

4.4.78 OPAL_PCI_TCE_KILL

```
int64_t opal_pci_tce_kill(uint64_t phb_id,
                          uint32_t kill_type,
                          uint64_t pe_number,
                          uint32_t tce_size,
                          uint64_t dma_addr,
                          uint32_t npages);
```

An abstraction around TCE kill. This allows host OS kernels to use an OPAL call if they don't know the model specific invalidation method.

Where kill_type is one of:

```
enum {
   OPAL_PCI_TCE_KILL_PAGES,
   OPAL_PCI_TCE_KILL_PE,
   OPAL_PCI_TCE_KILL_ALL,
};
```

Not all PHB types currently support this abstraction. It is supported in PHB4, which means from POWER9 onwards it will be present.

Returns

OPAL_PARAMETER if `phb_id` is invalid (or similar)

OPAL_UNSUPPORTED if PHB model doesn't support this call. This is likely true for systems before POWER9/PHB4. Do *NOT* rely on this call existing for systems prior to POWER9 (i.e. PHB4).

Example code (from linux/arch/powerpc/platforms/powernv/pci-ioda.c)

```
static inline void pnv_pci_ioda2_tce_invalidate_pe(struct pnv_ioda_pe *pe)
{
        struct pnv_phb *phb = pe->phb;

        if (phb->model == PNV_PHB_MODEL_PHB3 && phb->regs)
           pnv_pci_phb3_tce_invalidate_pe(pe);
        else
           opal_pci_tce_kill(phb->opal_id, OPAL_PCI_TCE_KILL_PE,
                             pe->pe_number, 0, 0, 0);
}
```

and

```
struct pnv_phb *phb = pe->phb;
unsigned int shift = tbl->it_page_shift;

if (phb->model == PNV_PHB_MODEL_PHB3 && phb->regs)
        pnv_pci_phb3_tce_invalidate(pe, rm, shift,
```

(continues on next page)

(continued from previous page)

```
                                   index, npages);
else
      opal_pci_tce_kill(phb->opal_id,
                         OPAL_PCI_TCE_KILL_PAGES,
                         pe->pe_number, 1u << shift,
                         index << shift, npages);
```

4.4.79 OPAL_POLL_EVENTS

Poll for outstanding events.

Fills in a bitmask of pending events.

Current events are:

OPAL_EVENT_OPAL_INTERNAL = 0x1

Currently unused.

OPAL_EVENT_NVRAM = 0x2

Unused

OPAL_EVENT_RTC = 0x4

TODO: clean this up, this is just copied from hw/fsp/fsp-rtc.c:

```
* Because the RTC calls can be pretty slow, these functions will shoot
* an asynchronous request to the FSP (if none is already pending)
*
* The requests will return OPAL_BUSY_EVENT as long as the event has
* not been completed.
*
* WARNING: An attempt at doing an RTC write while one is already pending
* will simply ignore the new arguments and continue returning
* OPAL_BUSY_EVENT. This is to be compatible with existing Linux code.
*
* Completion of the request will result in an event OPAL_EVENT_RTC
* being signaled, which will remain raised until a corresponding call
* to opal_rtc_read() or opal_rtc_write() finally returns OPAL_SUCCESS,
* at which point the operation is complete and the event cleared.
*
* If we end up taking longer than rtc_read_timeout_ms millieconds waiting
* for the response from a read request, we simply return a cached value (plus
* an offset calculated from the timebase. When the read request finally
* returns, we update our cache value accordingly.
*
* There is two separate set of state for reads and writes. If both are
* attempted at the same time, the event bit will remain set as long as either
* of the two has a pending event to signal.
```

OPAL_EVENT_CONSOLE_OUTPUT = 0x8

TODO

OPAL_EVENT_CONSOLE_INPUT = 0x10

TODO

OPAL_EVENT_ERROR_LOG_AVAIL = 0x20

TODO

OPAL_EVENT_ERROR_LOG = 0x40

TODO

OPAL_EVENT_EPOW = 0x80

TODO

OPAL_EVENT_LED_STATUS = 0x100

TODO

OPAL_EVENT_PCI_ERROR = 0x200

TODO

OPAL_EVENT_DUMP_AVAIL = 0x400

Signifies that there is a pending system dump available. See *OPAL Dumps* suite of calls for details.

OPAL_EVENT_MSG_PENDING = 0x800

4.4.80 OPAL Power Shift Ratio

Sometimes power management firmware needs to throttle power availability to system components in order to keep within power cap or thermal limits. It's possible to set a preference as to what trade-offs power management firmware will make. For example, certain workloads may heavily prefer throttling CPU over GPUs or vice-versa.

OPAL_GET_POWER_SHIFT_RATIO

OPAL call to read the power-shifting-ratio using a handle to identify the type (e.g CPU vs. GPU, CPU vs. MEM) which is exported via device-tree.

The call can be asynchronus, where the token parameter is used to wait for the completion.

Parameters

u32	handle
int	token
u32	*ratio

Returns

OPAL_SUCCESS Success

OPAL_PARAMETER Invalid ratio pointer

OPAL_UNSUPPORTED No support for reading psr

OPAL_HARDWARE Unable to procced due to the current hardware state

OPAL_ASYNC_COMPLETION Request was sent and an async completion message will be sent with token and status of the request.

OPAL_SET_POWER_SHIFT_RATIO

OPAL call to set power-shifting-ratio using a handle to identify the type of PSR which is exported in device-tree. This call can be asynchronus where the token parameter is used to wait for the completion.

Parameters

u32	handle
int	token
u32	ratio

Returns

OPAL_SUCCESS Success

OPAL_PARAMETER Invalid ratio requested

OPAL_UNSUPPORTED No support for changing the ratio

OPAL_PERMISSION Hardware cannot take the request

OPAL_ASYNC_COMPLETION Request was sent and an async completion message will be sent with token and status of the request.

OPAL_HARDWARE Unable to procced due to the current hardware state

OPAL_BUSY Previous request in progress

OPAL_INTERNAL_ERROR Error in request response

OPAL_TIMEOUT Timeout in request completion

4.4.81 OPAL Power Caps

Each entity that can be power capped is described in the device tree, see *power-mgt/powercap*. The values for each power cap aren't in the device tree, but rather fetched using the *OPAL_GET_POWERCAP* OPAL call. This is because there may be other entities such as a service processor that can change the nature of the power cap asynchronously to OPAL.

OPAL_GET_POWERCAP

The OPAL_GET_POWERCAP call retreives current information on the power cap.

For each entity that can be power capped, the device tree binding indicates what handle should be passed for each of the power cap properties (minimum possible, maximum possible, current powercap).

The current power cap must be between the minimium possible and maximum possible power cap. The minimum and maximum values are dynamic to allow for them possibly being changed by other factors or entities (e.g. service processor).

The call can be asynchronus, where the token parameter is used to wait for the completion.

Parameters

u32	handle
int	token
u32	*pcap

Returns

OPAL_SUCCESS Success

OPAL_PARAMETER Invalid pcap pointer

OPAL_UNSUPPORTED No support for reading powercap sensor

OPAL_HARDWARE Unable to procced due to the current hardware state

OPAL_ASYNC_COMPLETION Request was sent and an async completion message will be sent with token and status of the request.

OPAL_SET_POWERCAP

The OPAL_SET_POWERCAP call sets a power cap.

For each entity that can be power capped, the device tree binding indicates what handle should be passed for each of the power cap properties (minimum possible, maximum possible, current powercap).

The current power cap must be between the minimium possible and maximum possible power cap.

You cannot currently set the minimum or maximum power cap, and thus OPAL_PERMISSION will be returned if it is attempted to set. In the future, this may change - but for now, the correct behaviour for an Operating System is to not attempt to set them.

Parameters

:: u32 handle int token u32 pcap

Returns

OPAL_SUCCESS Success

OPAL_PARAMETER Invalid powercap requested beyond powercap limits

OPAL_UNSUPPORTED No support for changing the powercap

OPAL_PERMISSION Hardware cannot take the request

OPAL_ASYNC_COMPLETION Request was sent and an async completion message will be sent with token and status of the request.

OPAL_HARDWARE Unable to procced due to the current hardware state

OPAL_BUSY Previous request in progress

OPAL_INTERNAL_ERROR Error in request response

OPAL_TIMEOUT Timeout in request completion

4.4.82 OPAL_PRD_MSG

```
#define OPAL_PRD_MSG                            113

int64_t opal_prd_msg(struct opal_prd_msg *msg);
```

The OPAL_PRD_MSG call is used to pass a struct opal_prd_msg from the HBRT code into opal, and is paired with the *OPAL_PRD_MSG* message type.

Parameters

`struct opal_msg *msg` Passes an opal_msg, of type OPAL_PRD_MSG, from the OS to OPAL.

4.4.83 OPAL_QUERY_CPU_STATUS

```
#define OPAL_QUERY_CPU_STATUS                           42

enum OpalThreadStatus {
      OPAL_THREAD_INACTIVE = 0x0,
      OPAL_THREAD_STARTED = 0x1,
      OPAL_THREAD_UNAVAILABLE = 0x2 /* opal-v3 */
};

int64_t opal_query_cpu_status(uint64_t server_no, uint8_t *thread_status);
```

Sets *thread_status* to be the state of the *server_no* CPU thread. CPU threads can be owned by OPAL or the OS. Ownership changes based on *OPAL_START_CPU* and *OPAL_RETURN_CPU*.

`OPAL_THREAD_INACTIVE` Active in skiboot, not in OS. Skiboot owns the CPU thread.

`OPAL_THREAD_STARTED` CPU has been started by OS, not owned by OPAL.

OPAL_THREAD_UNAVAILABLE CPU is unavailable. e.g. is guarded out.

Returns

OPAL_PARAMETER Invalid address for *thread_status*, invalid CPU, or CPU not in OPAL or OS.

OPAL_SUCCESS Successfully retreived status.

4.4.84 OPAL_QUIESCE

```
#define OPAL_QUIESCE                          158

int64_t opal_quiesce(uint32_t quiesce_type, int32_t cpu_target);
```

The host OS can use *OPAL_QUIESCE* to ensure CPUs under host control are not executing OPAL. This is useful in crash or shutdown scenarios to try to ensure that CPUs are not holding locks, and is intended to be used with *OPAL_SIGNAL_SYSTEM_RESET*, for example.

Arguments

quiesce_type

QUIESCE_HOLD Wait for all target(s) currently executing OPAL to return to the host. Any new OPAL call that is made will be held off until QUIESCE_RESUME.

QUIESCE_REJECT Wait for all target(s) currently executing OPAL to return to the host. Any new OPAL call that is made will fail with OPAL_BUSY until QUIESCE_RESUME.

QUIESCE_LOCK_BREAK After QUIESCE_HOLD or QUIESCE_REJECT is successful, the CPU can call QUIESCE_LOCK_BREAK to skip all locking in OPAL to give the best chance of making progress in the crash/debug paths. The host should ensure all other CPUs are stopped (e.g., with OPAL_SIGNAL_SYSTEM_RESET) before this call is made, to avoid concurrency.

QUIESCE_RESUME Undo the effects of QUIESCE_HOLD/QUIESCE_REJECT and QUIESCE_LOCK_BREAK calls.

QUIESCE_RESUME_FAST_REBOOT As above, but also reset the tracking of OS calls into firmware as part of fast reboot (secondaries will never return to OS, but instead be released into a new OS boot).

target_cpu

`cpu_nr >= 0` The cpu server number of the target cpu to reset.

`-1` All cpus except the current one should be quiesced.

Returns

OPAL_SUCCESS The quiesce call was successful.

OPAL_PARTIAL Some or all of the CPUs executing OPAL when the call was made did not return to the host after a timeout of 1 second. This is a best effort at quiescing OPAL, and QUIESCE_RESUME must be called to resume normal firmware operation.

OPAL_PARAMETER A parameter was incorrect.

OPAL_BUSY This CPU was not able to complete the operation, either because another has concurrently started quiescing the system, or because it has not successfully called QUIESCE_HOLD or QUIESCE_REJECT before attempting QUIESCE_LOCK_BREAK or QUIESCE_RESUME.

4.4.85 OPAL Timed Power On and Delayed Power Off

```
#define OPAL_WRITE_TPO                          103
#define OPAL_READ_TPO                           104
#define OPAL_GET_DPO_STATUS                     105
```

TPO is a Timed Power On facility, and DPO is Delayed Power Off.

It is an OPTIONAL part of the OPAL spec.

If a platform supports Timed Power On (TPO), the RTC node in the device tree (itself under the "ibm,opal" node will have the has-tpo property:

```
rtc {
   compatible = "ibm,opal-rtc";
   has-tpo;
};
```

If the "has-tpo" proprety is *NOT* present then OPAL does *NOT* support TPO.

OPAL_READ_TPO

```
#define OPAL_READ_TPO                           104

static int64_t opal_read_tpo(uint64_t async_token, uint32_t *y_m_d, uint32_t *hr_min);
```

OPAL_WRITE_TPO

```
#define OPAL_WRITE_TPO                          103

int64_t fsp_opal_tpo_write(uint64_t async_token, uint32_t y_m_d, uint32_t hr_min);
```

OPAL_GET_DPO_STATUS

```
#define OPAL_GET_DPO_STATUS                     105

static int64_t opal_get_dpo_status(int64_t *dpo_timeout);
```

A *OPAL_MSG_DPO* message may be sent to indicate that there will shortly be a forced system shutdown. In this case, an OS can call *OPAL_GET_DPO_STATUS* to find out how many seconds it has before power is cut to the system.

This call could be present on systems where the service processor is integrated with a UPS or similar.

Returns zero if Delayed Power Off is not active, positive value indicating number of seconds remaining for a forced system shutdown. This will enable the host to schedule for shutdown voluntarily before timeout occurs.

Returns

OPAL_SUCCESS `dpo_timeout` is set to the number of seconds remaining before power is cut.

OPAL_WRONG_STATE A Delayed Power Off is not pending, `dpo_timeout` is set to zero.

4.4.86 OPAL_REINIT_CPUS

```
#define OPAL_REINIT_CPUS                        70

static int64_t opal_reinit_cpus(uint64_t flags);
```

This OPAL call reinitializes some bit of CPU state across *ALL* CPUs. Consequently, all CPUs must be in OPAL for this call to succeed (either at boot time or after OPAL_RETURN_CPU is called).

Arguments

Currently, possible flags are:

```
enum {
      OPAL_REINIT_CPUS_HILE_BE         = (1 << 0),
      OPAL_REINIT_CPUS_HILE_LE         = (1 << 1),
      OPAL_REINIT_CPUS_MMU_HASH        = (1 << 2),
      OPAL_REINIT_CPUS_MMU_RADIX       = (1 << 3),
      OPAL_REINIT_CPUS_TM_SUSPEND_DISABLED = (1 << 4),
};
```

Extra flags may be added in the future, so other bits *must* be 0.

On POWER7 CPUs, only OPAL_REINIT_CPUS_HILE_BE is supported. All other flags will return OPAL_UNSUPPORTED.

On POWER8 CPUs, only OPAL_REINIT_CPUS_HILE_BE and OPAL_REINIT_CPUS_HILE_LE are support and other bits *MUST NOT* be set.

On POWER9 CPUs, all options including OPAL_REINIT_CPUS_MMU_HASH and OPAL_REINIT_CPUS_MMU_RADIX.

OPAL_REINIT_CPUS_TM_SUSPEND_DISABLED

This flag requests that CPUs be configured with TM (Transactional Memory) suspend mode disabled. This may only be supported on some CPU versions.

Returns

OPAL_SUCCESS Success!

OPAL_UNSUPPORTED Processor does not suport reinit flags.

4.4.87 OPAL_RESYNC_TIMEBASE

```
#define OPAL_RESYNC_TIMEBASE                     79

int64_t opal_resync_timebase(void);
```

Resynchronises the timebase for all threads in a core to the timebase from chiptod.

Returns

OPAL_SUCCESS Successfully resynced timebases (or it's a no-op on this platform).

OPAL_HARDWARE Failed to resync timebase.

4.4.88 OPAL Real Time Clock (RTC) APIs

OPAL_RTC_READ

Read the Real Time Clock.

Parameters

`uint32_t* year_month_day` the year, month and day formatted as follows:

- bits 0-15 is bcd formatted year (0100-9999)
- bits 16-23 is bcd formatted month (01-12)
- bits 24-31 is bcd formatted day (01-31)

`uint64_t* hour_minute_second_millisecond` the hour, minute, second and millisecond formatted as follows:

- bits 0-16 is reserved
- bits 17-24 is bcd formatted hour (00-23)
- bits 25-31 is bcd formatted minute (00-59)
- bits 32-39 is bcd formatted second (00-60)
- bits 40-63 is bcd formatted milliseconds (000000-999999)

Calling

Since RTC calls can be pretty slow, *OPAL_RTC_READ* is likely to first return *OPAL_BUSY_EVENT*, requiring the caller to wait until the *OPAL_EVENT_RTC = 0x4* event has been signaled. Once the event has been signaled, a subsequent *OPAL_RTC_READ* call will retrieve the time. Since the *OPAL_EVENT_RTC = 0x4* event is used for both reading and writing the RTC, callers must be able to handle the event being signaled for a concurrent in flight *OPAL_RTC_WRITE* rather than this read request.

The following code is one way to correctly issue and then wait for a response:

```
int rc = OPAL_BUSY_EVENT;
while (rc == OPAL_BUSY_EVENT) {
      rc = opal_rtc_read(&y_m_d, &h_m_s_ms);
      if (rc == OPAL_BUSY_EVENT)
```

(continues on next page)

(continued from previous page)

```
        opal_poll_events(NULL);
}
```

Although as of writing all *OPAL_RTC_READ* backends are asynchronous, there is no requirement for them to be - it is valid for *OPAL_RTC_READ* to immediately return the retreived value rather than *OPAL_BUSY_EVENT*.

TODO: describe/document format of arguments.

Return codes

OPAL_SUCCESS parameters now contain the current time, or one read from cache.

OPAL_HARDWARE error in retrieving the time. May be transient error, may be permanent.

OPAL_PARAMETER year_month_day or hour_minute_second_millisecond parameters are NULL

OPAL_INTERNAL_ERROR something went wrong, Possibly reported in error log. This can be a transient error

OPAL_BUSY_EVENT request is in flight

OPAL_BUSY request may be in flight

OPAL_RTC_WRITE

OPAL_RTC_WRITE is much like *OPAL_RTC_READ* in that it can be asynchronous.

If multiple WRITES are issued before the first one completes, subsequent writes are ignored. There can only be one write in flight at any one time.

Format of the time is the same as for *OPAL_RTC_READ*.

4.4.89 OPAL Sensor Groups

See *ibm,opal/sensor-groups* for device tree layout.

OPAL_SENSOR_GROUP_ENABLE

```
#define OPAL_SENSOR_GROUP_ENABLE             163

int opal_sensor_group_enable(u32 group_hndl, int token, bool enable);
```

OPAL call to enable/disable the sensor group using a handle to identify the type of sensor group provided in the device tree.

For example this call is used to disable/enable copying of sensor group by OCC to main memory.

The call can be asynchronus, where the token parameter is used to wait for the completion.

Returns

OPAL_SUCCESS Success

OPAL_UNSUPPORTED No support to enable/disable the sensor group

OPAL_HARDWARE Unable to procced due to the current hardware state

OPAL_PERMISSION Hardware cannot take the request

OPAL_ASYNC_COMPLETION Request was sent and an async completion message will be sent with token and status of the request.

OPAL_BUSY Previous request in progress

OPAL_INTERNAL_ERROR Error in request response

OPAL_TIMEOUT Timeout in request completion

OPAL_SENSOR_GROUP_CLEAR

```
int opal_sensor_group_clear(u32 group_hndl, int token);

#define OPAL_SENSOR_GROUP_CLEAR                      156
```

OPAL call to clear the sensor groups data using a handle to identify the type of sensor group which is exported via DT.

The call can be asynchronus, where the token parameter is used to wait for the completion.

Returns

OPAL_SUCCESS Success

OPAL_UNSUPPORTED No support for clearing the sensor group

OPAL_HARDWARE Unable to procced due to the current hardware state

OPAL_PERMISSION Hardware cannot take the request

OPAL_ASYNC_COMPLETION Request was sent and an async completion message will be sent with token and status of the request.

OPAL_BUSY Previous request in progress

OPAL_INTERNAL_ERROR Error in request response

OPAL_TIMEOUT Timeout in request completion

4.4.90 OPAL_SENSOR_READ

```
#define OPAL_SENSOR_READ                    88

int64_t opal_sensor_read(uint32_t sensor_hndl, int token, uint32_t *sensor_data);
```

The OPAL sensor call reads a sensor data using a unique handler to identity the targeted sensor. The *sensor_handle* is provided via the device tree and is opaque to the OS (although we currently do use an encoding scheme).

This call can be asynchronous, when a message needs to be sent to a service processor for example. In this case, the call will return OPAL_ASYNC_COMPLETION and the token parameter will be used to wait for the completion of the request.

The OPAL API doesn't enforce alimit on the number of sensor calls that can be in flight.

Internally, *OPAL_SENSOR_READ* is implemented as a wrapper around *OPAL_SENSOR_READ_U64*. Any code targeting processor generations prior to POWER9 will need to use *OPAL_CHECK_TOKEN* to ensure *OPAL_SENSOR_READ_U64* is present and gracefully fall back to *OPAL_SENSOR_READ* if it is not.

Parameters

```
uint32_t sensor_handle
int      token
uint32_t *sensor_data
```

Return values

OPAL_SUCCESS Success!

OPAL_PARAMETER invalid sensor handle

OPAL_UNSUPPORTED platform does not support reading sensors.

OPAL_ASYNC_COMPLETION a request was sent and an async completion will be triggered with the @token argument

OPAL_PARTIAL the request completed but the data returned is invalid

OPAL_BUSY_EVENT a previous request is still pending

OPAL_NO_MEM allocation failed

OPAL_INTERNAL_ERROR communication failure with the FSP

OPAL_HARDWARE FSP is not available

4.4.91 OPAL_SENSOR_READ_U64

```
#define OPAL_SENSOR_READ_U64                  162

s64 opal_sensor_read_u64(u32 sensor_hndl, int token, u64 *sensor_data);
```

The OPAL sensor call to read sensor data of type u64. Unlike opal_sensor_read which reads upto u32 this call can be used to read values of sensors upto 64bits. The calling conventions and return values are same as *OPAL_SENSOR_READ*.

All sensors can be read through the *OPAL_SENSOR_READ_U64* call that can be read using the *OPAL_SENSOR_READ* call. Internally, *OPAL_SENSOR_READ* is a wrapper around *OPAL_SENSOR_READ_U64*. Any code targeting processor generations prior to POWER9 will need to use *OPAL_CHECK_TOKEN* to ensure *OPAL_SENSOR_READ_U64* is present and gracefully fall back to *OPAL_SENSOR_READ* if it is not.

4.4.92 OPAL_SET_XIVE

```
#define OPAL_SET_XIVE                                 19

int64_t opal_set_xive(uint32_t isn, uint16_t server, uint8_t priority);
```

The host calls this function to set the server (target processor) and priority parameters of an interrupt source.

This can be also used to mask or unmask the interrupt (by changing the priority to 0xff one masks an interrupt).

WARNINGS:

- For MSIs or generally edge sensitive interrupts, OPAL provides no guarantee as to whether the interrupt will be latched if it occurs while masked and replayed on unmask. It may or may not. The OS needs to be aware of this. The current implementation will *not* replay, neither on P8 nor on P9 XICS emulation.

- When masking, there is no guarantee that the interrupt will not still occur after this call returns. The reason is that it might already be on its way past the source controller and latched into one of the presenters. There is however a guarantee that it won't replay indefinitely so it's acceptable for the OS to simply ignore it.

Parameters

`isn` This is a global interrupt number as obtained from the device-tree "interrupts" or "interrupt-map" properties.

`server_number` is the mangled server (processor) that is to receive the interrupt request. The mangling means that the actual processor number is shifted left by 2 bits, the bottom bits representing the "link". However links aren't supported in OPAL so the bottom 2 bits should be 0.

`priority` is the interrupt priority value applied to the interrupt (0=highest, 0xFF = lowest/disabled).

4.4.93 OPAL_SIGNAL_SYSTEM_RESET

```
int64_t signal_system_reset(int32_t cpu_nr);
```

This OPAL call causes the specified cpu(s) to be reset to the system reset exception handler (0x100).

The SRR1 register will indicate a power-saving wakeup when appropriate, and the wake reason will be System Reset (see Power ISA).

This interrupt may not be recoverable in some cases (e.g., if it is raised when the target has MSR[RI]=0), so it should not be used in normal operation, but only for crashing, debugging, and similar exceptional cases.

OPAL_SIGNAL_SYSTEM_RESET can pull CPUs out of OPAL, which may be undesirable in a crash or shutdown situation (e.g., because they may hold locks which are required to access the console, or may be halfway through setting hardware registers), so OPAL_QUIESCE can be used before OPAL_SIGNAL_SYSTEM_RESET to (attempt to) ensure all CPUs are out of OPAL before being interrupted.

Arguments

```
int32_t cpu_nr
  cpu_nr >= 0         The cpu server number of the target cpu to reset.
  SYS_RESET_ALL (-1) All cpus should be reset.
  SYS_RESET_ALL_OTHERS (-2) All but the current cpu should be reset.
```

Returns

OPAL_SUCCESS The system reset requests to target CPU(s) was successful. This returns asynchronously without acknowledgement from targets that system reset interrupt processing has completed or even started.

OPAL_PARAMETER A parameter was incorrect.

OPAL_HARDWARE Hardware indicated failure during reset, some or all of the target CPUs may have the system reset delivered.

OPAL_CONSTRAINED Platform does not support broadcast operations.

OPAL_PARTIAL Platform can not reset sibling threads on the same core as requested. None of the specified CPUs are reset in this case.

OPAL_UNSUPPORTED This processor/platform is not supported.

4.4.94 OPAL_SLW_SET_REG

```
#define OPAL_SLW_SET_REG                        100

int64_t opal_slw_set_reg(uint64_t cpu_pir, uint64_t sprn, uint64_t val);
```

OPAL_SLW_SET_REG is used to inform low-level firmware to restore a given value of SPR when there is a state loss. The actual set of SPRs that are supported is platform dependent.

In Power 8, it uses p8_pore_gen_cpufreq_fixed(), api provided by pore engine, to inform the spr with their corresponding values with which they must be restored.

In Power 9, it uses p9_stop_save_cpureg(), api provided by self restore code, to inform the spr with their corresponding values with which they must be restored.

Parameters

`uint64_t cpu_pir` This parameter specifies the pir of the cpu for which the call is being made.

`uint64_t sprn` This parameter specifies the spr number as mentioned in p9_stop_api.H for Power9 and p8_pore_table_gen_api.H for Power8.

`uint64_t val` This parameter specifices value with which the spr should be restored.

Returns

OPAL_INTERNAL_ERROR On failure. The actual error code from the platform specific code is logged in the OPAL logs

OPAL_UNSUPPORTED If spr restore is not supported by pore engine.

OPAL_SUCCESS On success

4.4.95 Starting and stopping secondary CPUs

In this context, each thread is a CPU. That is, you start and stop threads of CPUs.

OPAL_START_CPU

```
#define OPAL_START_CPU                                  41

int64_t opal_start_cpu_thread(uint64_t server_no, uint64_t start_address);
```

Returns

OPAL_SUCCESS The CPU was instructed to start executing instructions from the specified *start_address*. This is an *asynchronous* operation, so it may take a short period of time before the CPU actually starts at that address.

OPAL_PARAMETER Invalid CPU.

OPAL_WRONG_STATE If the CPU thread is not in OPAL, or is being re-initialized through *OPAL_REINIT_CPUS*

OPAL_INTERNAL_ERROR Something else went horribly wrong.

OPAL_RETURN_CPU

```
#define OPAL_RETURN_CPU                         69

int64_t opal_return_cpu(void);
```

When OPAL first starts the host, all secondary CPUs are spinning in OPAL. To start them, one must call OPAL_START_CPU (you may want to OPAL_REINIT_CPUS to set the HILE bit first).

In cases where you need OPAL to do something for you across all CPUs, such as OPAL_REINIT_CPUS, (on some platforms) a firmware update or get the machine back into a similar state as to when the host OS was started (e.g. for kexec) you may also need to return control of the CPU to OPAL.

Returns

This call does **not return**. You need to OPAL_START_CPU.

4.4.96 OPAL_SYNC_HOST_REBOOT

```
#define OPAL_SYNC_HOST_REBOOT                   87

static int64_t opal_sync_host_reboot(void);
```

This OPAL call halts asynchronous operations in preparation for something like kexec. It will halt DMA as well notification of some events (such as a new error log being available for retreival).

It's meant to be called in a loop until *OPAL_SUCCESS* is returned.

Returns

OPAL_SUCCESS Success!

OPAL_BUSY_EVENT not yet complete, call opal_sync_host_reboot() again, possibly with a short delay.

OPAL_BUSY Call opal_poll_events() and then retry opal_sync_host_reboot

4.4.97 OPAL_TEST

OPAL_TEST is a REQUIRED call for OPAL and conforming implementations MUST have it.

It is designed to test basic OPAL call functionality.

Token:

```
#define OPAL_TEST                          0
```

Arguments

```
uint64_t     arg
```

Returns

```
0xfeedf00d
```

Function

OPAL_TEST MAY print a string to the OPAL log with the value of argument.

For example, the reference implementation (skiboot) implements *OPAL_TEST* as:

```
static uint64_t opal_test_func(uint64_t arg)
{
        printf("OPAL: Test function called with arg 0x%llx\n", arg);

        return 0xfeedf00d;
}
```

4.4.98 OPAL_WRITE_OPPANEL_ASYNC

```
#define OPAL_WRITE_OPPANEL_ASYNC                95

typedef struct oppanel_line {
     __be64 line;
     __be64 line_len;
} oppanel_line_t;

int64_t opal_write_oppanel_async(uint64_t async_token,
                                 oppanel_line_t *lines,
                                 uint64_t num_lines);
```

Writes to a (possibly physical) Operator Panel. An Operator Panel contains a small LCD screen (or similar) displaying a small amount of ASCII text. It can be used to report on boot progress, failure, or witty messages from a systems administrator.

A typical panel, as present on IBM FSP based machines, is two lines of 16 characters each.

See *Operator Panel (oppanel)* for how the panel is described in the device tree. Not all systems have an operator panel.

Pass in an array of oppanel_line_t structs defining the ASCII characters to display on each line of the oppanel. If there are two lines on the physical panel, and you only want to write to the first line, you only need to pass in one line. If you only want to write to the second line, you need to pass in both lines, and set the line_len of the first line to zero.

Returns

OPAL_SUCCESS Success! Typically this is async operation, so immediate success is unlikely.

OPAL_ASYNC_COMPLETION Request submitted asynchronously.

OPAL_PARAMETER Invalid *lines* or *num_lines*

OPAL_NO_MEM Not enough free memory in OPAL to complete the request.

OPAL_INTERNAL_ERROR Other internal error.

4.4.99 OPAL_XSCOM_READ

```
#define OPAL_XSCOM_READ                                 65

int xscom_read(uint32_t partid, uint64_t pcb_addr, uint64_t *val);
```

This low level call will read XSCOM values directly.

They should only be used by low level manufacturing/debug tools. "Normal" host OS kernel code should not know about XSCOM.

This is also needed by HBRT/*opal-prd*.

Returns

OPAL_SUCCESS Success!

OPAL_HARDWARE if operation failed

OPAL_WRONG_STATE if CPU is asleep

OPAL_XSCOM_BUSY Alias for *OPAL_BUSY*.

OPAL_XSCOM_CHIPLET_OFF Alias for *OPAL_WRONG_STATE*

OPAL_XSCOM_PARTIAL_GOOD XSCOM Partial Good

OPAL_XSCOM_ADDR_ERROR XSCOM Address Error

OPAL_XSCOM_CLOCK_ERROR XSCOM Clock Error

OPAL_XSCOM_PARITY_ERROR XSCOM Parity Error

OPAL_XSCOM_TIMEOUT XSCOM Timeout

OPAL_XSCOM_CTR_OFFLINED XSCOM Controller Offlined due to too many errors.

4.4.100 OPAL_XSCOM_WRITE

```
#define OPAL_XSCOM_WRITE                        66

int xscom_write(uint32_t partid, uint64_t pcb_addr, uint64_t val);
```

This low level call will write an XSCOM value directly.

They should only be used by low level manufacturing/debug tools. "Normal" host OS kernel code should not know about XSCOM.

This is also needed by HBRT/*opal-prd*.

Returns

OPAL_SUCCESS Success!

OPAL_HARDWARE if operation failed

OPAL_WRONG_STATE if CPU is asleep

OPAL_XSCOM_BUSY Alias for *OPAL_BUSY*.

OPAL_XSCOM_CHIPLET_OFF Alias for *OPAL_WRONG_STATE*

OPAL_XSCOM_PARTIAL_GOOD XSCOM Partial Good

OPAL_XSCOM_ADDR_ERROR XSCOM Address Error

OPAL_XSCOM_CLOCK_ERROR XSCOM Clock Error

OPAL_XSCOM_PARITY_ERROR XSCOM Parity Error

OPAL_XSCOM_TIMEOUT XSCOM Timeout

OPAL_XSCOM_CTR_OFFLINED XSCOM Controller Offlined due to too many errors.

4.4.101 OPAL_NX_COPROC_INIT

This OPAL call resets read offset and queued entries in high and normal priority receive FIFO control registers. The kernel initializes read offset entry in RXFIFO that it maintains during initialization. So this register reset is needed for NX module reload or in kexec boot to make sure read offset value matches with kernel entries. Otherwise NX reads requests with wrong offset in RxFIFO which could cause NX request failures.

The kernel initiates this call for each coprocessor type such as 842 and GZIP per NX instance.

Arguments

```
``uint32_t chip_id``
  Contains value of the chip number identified at boot time.

``uint32_t pid``
  Contains NX coprocessor type (pid from the device tree).
```

Returns

OPAL_SUCCESS The call to reset readOffset and queued entries for high and normal FIFOs was successful.

OPAL_PARAMETER Indicates invalid chip ID or NX coprocessor type.

OPAL_UNSUPPORTED Not supported on P7 and P8.

4.4.102 POWER9 Changes to OPAL API

This document is a summary of POWER9 changes to the OPAL API over what it was for POWER7 and POWER8. As the POWER series of processors (at least up to POWER9) require changes in the hypervisor to work on a new processor generation, this gives us an opportunity with POWER9 to clean up several parts of the OPAL API.

Eventually, when the kernel drops support for POWER8 and before, we can then remove the associated kernel code too.

OPAL_REINIT_CPUS

Can now be extended beyond HILE BE/LE bits. If invalid flags are set on POWER9, OPAL_UNSUPPORTED will be returned.

Device Tree

- `/ibm,opal/` compatible property now just lists `ibm,opal-v3` and no longer `ibm,opal-v2` (power9 and above only)
- Use only `stdout-path` property from POWER9 and above as usage of `linux,stdout-path` is deprecated
- Rename `fsp-ipl-side` as `sp-ipl-side` in `/ipl-params`
- Add interrupt-parent property for `/ibm,opal/ipmi` node on POWER9 and above to make use of OPAL irqchip rather than event interface in linux.

TODO

Things we still have to do for POWER9:

- PCI to use async API rather than returning delays
- deprecate/remove v1 APIs where there's a V2
- Fix this FWTS warning:

```
FAILED [MEDIUM] DeviceTreeBaseDTCWarnings: Test 3, dtc reports warnings from
device tree: Warning (reg_format): "reg" property in /ibm,opal/flash@0 has
invalid length (8 bytes) (#address-cells == 0, #size-cells == 0)
```

- Remove mi-version / ml-version from `/ibm,opal/firmware` and replace with something better and more portable

4.4.103 OPAL API Return Codes

All OPAL calls return an integer relaying the success/failure of the OPAL call.

Success is typically indicated by OPAL_SUCCESS. Failure is always indicated by a negative return code.

Conforming host Operating Systems MUST handle return codes other than those listed here. In future OPAL versions, additional return codes may be added.

In the reference implementation (skiboot) these are all in include/opal-api.h

There have been additions to the return codes from OPAL over time. A conforming host OS should gracefully handle receiving a new error code for existing calls.

An OS running on a POWER8 system only has to know about error codes that existed when POWER8 with OPAL was introduced (indicated by YES in the POWER8 column below). Additional OPAL error codes *may be returned on POWER8 systems* and as such OSs need to gracefully handle unknown error codes.

An OS running on POWER9 or above must handle all error codes as they were when POWER9 was introduced. We use the placeholder "v1.0" version for "since the dawn of time" even though there never was a skiboot v1.0

Name	Return Code	POWER8 GA	POWER9 GA	skiboot version where introduced
OPAL_SUCCESS	0	YES	YES	v1.0 (initial release)
OPAL_PARAMETER	-1	YES	YES	v1.0 (initial release)
OPAL_BUSY	-2	YES	YES	v1.0 (initial release)
OPAL_PARTIAL	-3	YES	YES	v1.0 (initial release)
OPAL_CONSTRAINED	-4	YES	YES	v1.0 (initial release)

Continued on next page

Table 2 – continued from previous page

OPAL_CLOSED	-5	YES	YES	v1.0 (initial release)
OPAL_HARDWARE	-6	YES	YES	v1.0 (initial release)
OPAL_UNSUPPORTED	-7	YES	YES	v1.0 (initial release)
OPAL_PERMISSION	-8	YES	YES	v1.0 (initial release)
OPAL_NO_MEM	-9	YES	YES	v1.0 (initial release)
OPAL_RESOURCE	-10	YES	YES	v1.0 (initial release)
OPAL_INTERNAL_ERROR	-11	YES	YES	v1.0 (initial release)
OPAL_BUSY_EVENT	-12	YES	YES	v1.0 (initial release)
OPAL_HARDWARE_FROZEN	-13	YES	YES	v1.0 (initial release)
OPAL_WRONG_STATE	-14	YES	YES	v1.0 (initial release)
OPAL_ASYNC_COMPLETION	-15	YES	YES	v1.0 (initial release)
OPAL_EMPTY	-16	NO	YES	v4.0
OPAL_I2C_TIMEOUT	-17	NO	YES	*skiboot-5.1.0*
OPAL_I2C_INVALID	-18	NO	YES	*skiboot-5.1.0*
OPAL_I2C_LBUS_PARITY	-19	NO	YES	*skiboot-5.1.0*
OPAL_I2C_BKEND_OVERRUN	-20	NO	YES	*skiboot-5.1.0*
OPAL_I2C_BKEND_ACCESS	-21	NO	YES	*skiboot-5.1.0*
OPAL_I2C_ARBT_LOST	-22	NO	YES	*skiboot-5.1.0*
OPAL_I2C_NACK_RCVD	-23	NO	YES	*skiboot-5.1.0*
OPAL_I2C_STOP_ERR	-24	NO	YES	*skiboot-5.1.0*
OPAL_XSCOM_BUSY	OPAL_BUSY	NO	YES	*skiboot-5.4.0*
OPAL_XSCOM_CHIPLET_OFF	OPAL_WRONG_STATE	NO	YES	*skiboot-5.4.0*
OPAL_XSCOM_PARTIAL_GOOD	-25	NO	YES	*skiboot-5.4.0*
OPAL_XSCOM_ADDR_ERROR	-26	NO	YES	*skiboot-5.4.0*
OPAL_XSCOM_CLOCK_ERROR	-27	NO	YES	*skiboot-5.4.0*
OPAL_XSCOM_PARITY_ERROR	-28	NO	YES	*skiboot-5.4.0*
OPAL_XSCOM_TIMEOUT	-29	NO	YES	*skiboot-5.4.0*
OPAL_XSCOM_CTR_OFFLINED	-30	NO	YES	*skiboot-5.4.0*
OPAL_XIVE_PROVISIONING	-31	NO	YES	*skiboot-5.5.0*
OPAL_XIVE_FREE_ACTIVE	-32	NO	YES	*skiboot-5.5.0*
OPAL_TIMEOUT	-33	NO	YES	*skiboot-5.8*

The core set of return codes are:

OPAL_SUCCESS

```
#define OPAL_SUCCESS            0
```

Success!

OPAL_PARAMETER

```
#define OPAL_PARAMETER          -1
```

A parameter was invalid. This will also be returned if you call an invalid OPAL call. To determine if a specific OPAL call is supported or not, OPAL_CHECK_TOKEN should be called rather than relying on OPAL_PARAMETER being returned for an invalid token.

OPAL_BUSY

```
#define OPAL_BUSY              -2
```

Try again later. Related to *OPAL_BUSY_EVENT*, but *OPAL_BUSY* indicates that the caller need not call *OPAL_POLL_EVENTS* itself. **TODO** Clarify current situation.

OPAL_PARTIAL

```
#define OPAL_PARTIAL           -3
```

The operation partially succeeded.

OPAL_CONSTRAINED

```
#define OPAL_CONSTRAINED       -4
```

FIXME

OPAL_CLOSED

```
#define OPAL_CLOSED            -5
```

FIXME document these

OPAL_HARDWARE

```
#define OPAL_HARDWARE                  -6
```

FIXME document these

OPAL_UNSUPPORTED

```
#define OPAL_UNSUPPORTED       -7
```

Unsupported operation. Non-fatal.

OPAL_PERMISSION

```
#define OPAL_PERMISSION                -8
```

Inadequate permission to perform the operation.

OPAL_NO_MEM

```
#define OPAL_NO_MEM          -9
```

Indicates a temporary or permanent lack of adequate memory to perform the operation. Ideally, this should never happen. Skiboot reserves a small amount of memory for its heap and some operations (such as I2C requests) are allocated from this heap.

If this is ever hit, you should likely file a bug.

OPAL_RESOURCE

```
#define OPAL_RESOURCE                -10
```

When trying to use a limited resource, OPAL found that there were none free. While OPAL_BUSY indicates that OPAL may soon be able to proces the requent, OPAL_RESOURCE is a more permanent error and while the resource *may* become available again in the future, it is not certain that it will.

OPAL_INTERNAL_ERROR

```
#define OPAL_INTERNAL_ERROR  -11
```

Something has gone wrong inside OPAL. This is likely a bug somewhere and we return OPAL_INTERNAL_ERROR for safety.

OPAL_BUSY_EVENT

```
#define OPAL_BUSY_EVENT              -12
```

The same as *OPAL_BUSY* but signals that the OS should call *OPAL_POLL_EVENTS* as that may be required to get into a state where the call will succeed.

OPAL_HARDWARE_FROZEN

```
#define OPAL_HARDWARE_FROZEN -13
```

OPAL_WRONG_STATE

```
#define OPAL_WRONG_STATE     -14
```

The requested operation requires a (hardware or software) component to be in a different state. For example, you cannot call OPAL_START_CPU on a CPU that is not currently in OPAL.

OPAL_ASYNC_COMPLETION

```
#define OPAL_ASYNC_COMPLETION        -15
```

For asynchronous calls, successfully queueing/starting executing the command is indicated by the OPAL_ASYNC_COMPLETION return code. pseudo-code for an async call:

```
token = opal_async_get_token();
rc = opal_async_example(foo, token);
if (rc != OPAL_ASYNC_COMPLETION)
    handle_error(rc);
rc = opal_async_wait(token);
// handle result here
```

OPAL_EMPTY

```
#define OPAL_EMPTY              -16
```

The call was successful and the correct result is empty. For example, the OPAL_IPMI_RECV call can succeed and return that there is no waiting IPMI message.

OPAL_I2C_TIMEOUT

```
#define OPAL_I2C_TIMEOUT      -17
```

OPAL_I2C_INVALID

```
#define OPAL_I2C_INVALID_CMD  -18
```

OPAL_I2C_LBUS_PARITY

```
#define OPAL_I2C_LBUS_PARITY  -19
```

OPAL_I2C_BKEND_OVERRUN

```
#define OPAL_I2C_BKEND_OVERRUN        -20
```

OPAL_I2C_BKEND_ACCESS

```
#define OPAL_I2C_BKEND_ACCESS -21
```

OPAL_I2C_ARBT_LOST

```
#define OPAL_I2C_ARBT_LOST    -22
```

OPAL_I2C_NACK_RCVD

```
#define OPAL_I2C_NACK_RCVD    -23
```

OPAL_I2C_STOP_ERR

```
#define OPAL_I2C_STOP_ERR     -24
```

OPAL_XSCOM_BUSY

An alias for *OPAL_BUSY*

OPAL_XSCOM_CHIPLET_OFF

An alias for *OPAL_WRONG_STATE*

OPAL_XSCOM_PARTIAL_GOOD

```
#define OPAL_XSCOM_PARTIAL_GOOD -25
```

OPAL_XSCOM_ADDR_ERROR

```
#define OPAL_XSCOM_ADDR_ERROR -26
```

OPAL_XSCOM_CLOCK_ERROR

```
#define OPAL_XSCOM_CLOCK_ERROR       -27
```

OPAL_XSCOM_PARITY_ERROR

```
#define OPAL_XSCOM_PARITY_ERROR      -28
```

OPAL_XSCOM_TIMEOUT

```
#define OPAL_XSCOM_TIMEOUT   -29
```

OPAL_XSCOM_CTR_OFFLINED

```
#define OPAL_XSCOM_CTR_OFFLINED      -30
```

OPAL_XIVE_PROVISIONING

```
#define OPAL_XIVE_PROVISIONING       -31
```

OPAL_XIVE_FREE_ACTIVE

```
#define OPAL_XIVE_FREE_ACTIVE        -32
```

OPAL_TIMEOUT

```
#define OPAL_TIMEOUT         -33
```

4.4.104 Removed Calls

Under **very** specific and careful circumstances, an OPAL call has been removed and no longer supported.

Name	API Token	Introduced	Removed
OPAL_GET_COMPLETION_TOKEN_STATUS	21	Never	
OPAL_PCI_SHPC	24	Never	
OPAL_PCI_SET_PHB_TABLE_MEMORY	30	Never	
OPAL_PCI_GET_XIVE_REISSUE	35	Never	
OPAL_PCI_GET_XIVE_REISSUE	36	Never	
OPAL_PCI_FENCE_PHB	52	Never	
OPAL_PCI_MASK_PE_ERROR	54	Never	
OPAL_SET_SLOT_LED_STATUS	55	Never	
OPAL_SET_SYSTEM_ATTENTION_LED	57	Never	
OPAL_RESERVED1	58	Never	
OPAL_RESERVED2	59	Never	
OPAL_ELOG_SEND	92	pre-v1.0	pre-v1.0
OPAL_PCI_GET_PHB_DIAG_DATA	51	pre-v1.0	pre-v1.0, with last remnants removed in skiboot-6.4
OPAL_GET_XIVE_SOURCE	38	v1.0 Initial Release	skiboot-6.4
OPAL_WRITE_OPPANEL	43	pre-v1.0	pre-v1.0
OPAL_OLD_I2C_REQUEST	106	v4.0	v4.0
OPAL_REGISTER_OPAL_EXCEPTION_HANDLER	22	v1.0 Initial Release	*skiboot 5.0*
OPAL_PCI_SET_HUB_TCE_MEMORY	11	pre-v1.0	skiboot-5.2.0
OPAL_PCI_SET_PHB_TCE_MEMORY	12	pre-v1.0	skiboot-5.2.0
OPAL_PCI_EEH_FREEZE_STATUS2	61	v1.0 Initial Release	skiboot-6.4

OPAL_GET_COMPLETION_TOKEN_STATUS

In the before time, long-long ago, there existed something called OPAL before the incarnation we know today. Presumably, this long forgotten incarnation had a call called this.

This call has never been implemented, and never will be.

OPAL_PCI_SHPC

A remnant of a long forgotten incarnation of OPAL. Never implemented, never will be.

OPAL_PCI_SET_PHB_TABLE_MEMORY

A remnant of an old API design. Never implemented, never used. Only ever returned *OPAL_UNSUPPORTED*, now the call is not implemented at all.

OPAL_PCI_GET_XIVE_REISSUE

A remnant of something prior to OPALv3. Never implemented in skiboot and never used by anyone. Returend *OPAL_UNSUPPORTED* until skiboot-6.4, where it was removed.

OPAL_PCI_SET_XIVE_REISSUE

A remnant of something prior to OPALv3. Never implemented in skiboot and never used by anyone. Returend *OPAL_UNSUPPORTED* until skiboot-6.4, where it was removed.

OPAL_PCI_FENCE_PHB

Never implemented.

OPAL_PCI_MASK_PE_ERROR

Never implemented.

OPAL_SET_SLOT_LED_STATUS

Never implemented.

OPAL_SET_SYSTEM_ATTENTION_LED

Never implemented.

OPAL_RESERVED1

Reserved for future use, but never used.

OPAL_RESERVED2

Reserved for future use, but never used.

OPAL_ELOG_SEND

Briefly present prior to the first public release of OPAL. Never used in any public kernel tree. If this functionality were to ever be implemented, it'd appear as *OPAL_ELOG_WRITE* rather than this call.

OPAL_PCI_GET_PHB_DIAG_DATA

This call was introduced and functionally removed (all backends for it were) before the first public opal release. It has not been used since Linux 3.11-rc1. Considering the state of EEH in such old kernels and firmware, removing the remnants of this call is considered safe. If for some bizarre reason such an old kernel is run on skiboot-6.4 or later, an *OPAL_PARAMETER* error will be returned instead of *OPAL_UNSUPPORTED*.

It is replaced by *OPAL_PCI_GET_PHB_DIAG_DATA2* instead.

OPAL_GET_XIVE_SOURCE

While this call was technically implemented by skiboot, no code has ever called it, and it was only ever implemented for the p7ioc-phb back-end (i.e. POWER7). Since this call was unused in Linux, and that POWER7 with OPAL was only ever available internally, it was determined that it was safe to remove this call as of skiboot-6.4.

OPAL_WRITE_OPPANEL

Never in a released version, use *OPAL_WRITE_OPPANEL_ASYNC*.

OPAL_OLD_I2C_REQUEST

Never used. Only existing briefly in the *skiboot 4.0* development cycle.

OPAL_REGISTER_OPAL_EXCEPTION_HANDLER

```
#define OPAL_REGISTER_OPAL_EXCEPTION_HANDLER 22


int64_t opal_register_exc_handler(uint64_t opal_exception __unused,
                                  uint64_t handler_address __unused,
                                  uint64_t glue_cache_line __unused);
```

This call existed for a very short period of time and only ever worked with Big Endian host operating systems. The idea was that OPAL would handle HMIs and an OS could (if it chose to) register a handler for them. This call is not required since the introduction of *OPAL_HANDLE_HMI* and all machines that ever shipped without *OPAL_HANDLE_HMI* have a firmware update that supports it. For IBM Tuleta machines, this was FW810.20 (released Oct 2014) that had *OPAL_HANDLE_HMI* support.

This call was removed in *skiboot 5.0* and now just returns *OPAL_UNSUPPORTED*.

Use of the *OPAL_HANDLE_HMI* call was introduced in Linux 3.17.

OPAL_PCI_SET_HUB_TCE_MEMORY

```
#define OPAL_PCI_SET_HUB_TCE_MEMORY          11

int64_t opal_pci_set_hub_tce_memory(uint64_t hub_id,
                                    uint64_t tce_mem_addr __unused,
                                    uint64_t tce_mem_size __unused);
```

This call was only ever relevant for p5ioc based POWER7 systems. These were never available with OPAL outside of IBM development.

Support for POWER7 systems with p5ioc was dropped in skiboot-5.2.0, and these systems were only ever used with OPAL inside IBM for development and bring-up purposes.

Support for p5ioc was removed from the Linux kernel in v4.6-rc1.

OPAL_PCI_SET_PHB_TCE_MEMORY

```
#define OPAL_PCI_SET_PHB_TCE_MEMORY          12

int64_t opal_pci_set_phb_tce_memory(uint64_t phb_id,
                                    uint64_t tce_mem_addr,
                                    uint64_t tce_mem_size);
```

This call was only ever relevant for p5ioc based POWER7 systems. These were never available with OPAL outside of IBM development.

Support for POWER7 systems with p5ioc was dropped in skiboot-5.2.0, and these systems were only ever used with OPAL inside IBM for development and bring-up purposes.

Support for p5ioc was removed from the Linux kernel in v4.6-rc1.

4.4.105 Future Calls

OPAL_ELOG_WRITE

May be implemented in the future to complement the *OPAL_ELOG_READ* call.

CHAPTER

FIVE

SKIBOOT RELEASE NOTES

5.1 Release Notes

5.1.1 skiboot 4.0

Skiboot 4.0 was released 19th November 2014. It was the first release to obtain an independent version number and numbering scheme. Previous releases were identified either purely by a GIT SHA1 hash or the associated PowerKVM release number.

This release introduced the following OPAL calls:

- *OPAL_IPMI_SEND*
- *OPAL_IPMI_RECV*
- *OPAL_I2C_REQUEST*

5.1.2 skiboot 4.1

Skiboot 4.1 was released 10th December 2014. It was a release where more development transitioned to the open source mailing list rather than internal mailing lists.

Changes include:

- We now build with -fstack-protector and -Werror
- Stack checking extensions when built with STACK_CHECK=1
- Reduced stack usage in some areas, -Wstack-usage=1024 now.
  - Some functions could use 2kb stack, now all are <1kb
- Unsafe libc functions such as sprintf() have been removed
- Symbolic backtraces
- expose skiboot symbol map to OS (via device-tree)
- removed machine check interrupt patching in OPAL
- occ/hbrt: Call stopOCC() for implementing reset OCC command from FSP
- occ: Fix the low level ACK message sent to FSP on receiving {RESET/LOAD}_OCC
- hardening to errors of various FSP code
  - fsp: Avoid NULL dereference in case of invalid class_resp bits- abort if device tree parsing fails
  - FSP: Validate fsp_msg in fsp_queue_msg

  - fsp-elog: Add various NULL checks
- Finessing of when to use error log vs prerror()
- More i2c work
- Can now run under Mambo simulator (see external/mambo/skiboot.tcl) (commonly known as "POWER8 Functional Simulator")
- Document skiboot versioning scheme
- opal: Handle more TFAC errors.
  - TB_RESIDUE_ERR, FW_CONTROL_ERR and CHIP_TOD_PARITY_ERR
- ipmi: populate FRU data
- rtc: Add a generic rtc cache
- ipmi/rtc: use generic cache
- Error Logging backend for bmc based machines
- PSI: Drive link down on HIR
- occ: Fix clearing of OCC interrupt on remote fix

5.1.3 skiboot 4.1.1

Skiboot 4.1 was released 30th January 2015.

- fsp: Avoid NULL dereference in case of invalid class_resp bits CQ: SW288484
- Makefile: Support CROSS_COMPILE as well as CROSS
- Additional unit testing:
  - Tiny hello_world kernel
  - Will run boot tests with hello_world and (if present) petitboot image in the POWER8 Functional simulator (mambo) (if present)
  - Run CCAN unit tests as part of 'make check'
  - Increased testing of PEL code
  - unit test console-log
  - skeleton libc unit tests
- Fix compatible match for palmetto & habanero The strings should be "tyan,..." not "ibm,..." (N/A for IBM systems)
- i2c: Unify the frequencies to calculate bit rate divisor
- Unlock rtc cache lock when cache isn't valid Could cause IPL crash on POWER7
- Initial documentation for OPAL API, ABI and Specification
- Add Firestone platform
- Fix crash when one socket wasn't populated with a CPU LTC-Bugzilla: 120562
- Bug fix in RTC state machine which possibly led to RTC not working
- Makefile fixes for running with some GCC 4.9 compilers
- Add device tree properties for pstate vdd and vcs values

- cpuidle: Add validated metrics for idle states Export residency times in device tree
- Revert “platforms/astbmc: Temporary reboot workaround” (N/A for IBM systems)
- Fix buffer overrun in print_* functions. This could cause IPL failures or conceivably other runtime problems

5.1.4 skiboot 5.0

Skiboot 5.0 was released Friday 10th April 2015.

Changes in 5.0 (since rc3):

- Fix chip id for nx coprocessors.
- hw/ipmi: Fix FW Boot Progress sensor
- bt: Add a temporary workaround for bmc dropping messages
- FSP/CUPD: Fix lock issue

Changes in rc3 (since rc2):

- add support for cec_power_down on mambo
- external/opal-prd: Use link register for cross-endian branch
- opal header file rework, Linux and skiboot now very closely match (API in opal-api.h)
- libflash: don’t use the low level interface if it doesn’t exist
- libflash/file: add file abstraction for libflash
- external: create a GUARD partition parsing utility

Changes in rc2 (since rc1):

- opal: Fix an issue where partial LID load causes opal to hang.
- nx: use proc_gen instead of param
- use chip id for NX engine Coproc Instance num
- Fix (hopefully) missing dot symbols in skiboot.map
- exceptions: Catch exceptions at boot time
- exceptions: Remove deprecated exception patching stuff
- mambo: Make mambo_utils.tcl optional
- mambo: Exit mambo when the simulation is stopped
- add NX register defines
- set NX crb input queues to 842 only
- core: Catch attempts to branch through a NULL pointer
- plat/firestone: Add missing platform hooks
- plat/firestone: Add missing platform hooks
- elog: Don’t call uninitialized platform elog_commit
- external/opal-prd: Use “official” switch-endian syscall
- hw/ipmi: Rework sensors and fix boot count sensor

Changes in rc1 (since 4.1.1):

General:

- big OPAL API documentation updates We now document around 19 OPAL calls. There's still ~100 left to doc though :)
- skiboot can load FreeBSD kernel payload (thanks to Nathan Whitehorn)
- You can now run sparse by setting C=1 when building
- PSI: Revert the timeout for PSI link recovery to architected value now 30mins (prev 15)
- cpuidle: Add validated metrics for idle states
- core/flash: Add flash API OPAL_FLASH_(READ|WRITE|ERASE)
- capi: Dynamically calculate which CAPP port to use no longer hardwired to PHB0
- vpd: Use slca parent-child relationship to create vpd tree
- opal: Do not overwrite same HMI event for multiple HMI errors. Now Linux will get a HMI event for each HMI error
- HMI event v2 now includes information about checkstop
- HMI improvements, handle more conditions gracefully:
  - TB residue error
  - TFMR firmware control error
  - TFMR parity
  - TFMR HDEC parity error
  - TFMR DEC parity error
  - TFMR SPURR/PURR parity error
  - TB residue and HDEC parity HMI errors on split core
- hostservices: Cache lids prior to first load request
- Warn when pollers are called with a lock held and keep track of lock depth.

  NOTE: This means we will get backtraces in skiboot msglog on FSP machines This is a KNOWN ISSUE and is largely harmless. There's still a couple that we haven't yet cleaned, these messages can be thought of as a TODO list for developers.
- Don't run pollers in time_wait if lock held
- pci: Don't hang if we have only one CPU
- Detect recursive poller entry
- General cleanup
- Cleanup of opal.h so that we can have Linux and skiboot match
- add sparse annotations to opal.h
- Platform hooks for loading and preloading resources (LIDs) This lays the groundwork for cutting 4-20 seconds off boot in a future skiboot release.
- Fix potential race when clearing OCC interrupt status
- Add platform operation for reading sensors
  - add support to read core and memory buffer temperatures

Mambo/POWER8 Functional Simulator:

- Replace is_mambo_chip() with a better quirks mechanism.
- Don't hang if we only have one CPU and PCI.

BMC systems:

- BMC can load payload from flash
- IPMI on BMC systems: graceful poweroff and reboot
- IPMI on BMC systems: watchdog timer support
- IPMI on BMC systems: PNOR locking
- Support for IPMI progress sensor
- IPMI boot count sensor
- capi: Rework microcode flash download and CAPP upload load microcode on non-fsp systems
- NEW opal-prd userspace tool that handles PRD on non-FSP systems. and OPAL PRD calls to support it.
- Improvements to opal-prd, libflash, and ipmi
- ECC support in libflash
- Load CAPI micro code, enabling CAPI on OpenPower systems.
- Dynamically calculate which CAPP port to use, don't hardcode to PHB0
- memboot flash backend

POWER8

- add nx-842 coproc support

FSP systems:

- Make abort() update sp attn area (like assert does) On FSP systems this gives better error logs/dumps when abort() is hit
- FSP/LEDS: Many improvements and bug fixes
- LED support for FSP machines Adds OPAL_LEDS_(GET|SET)_INDICATOR and device-tree bindings
- Refactor of fsp-rtc
- OCC loading fixes, including possible race condition where we would fail to IPL.

POWER7

- Fix unsupported return code of OPAL_(UN)REGISTER_DUMP_REGION on P7
- occ: Don't do bad XSCOMs on P7 The OCC interrupt register only exists on P8, accessing it on P7 causes not only error logs but also causes PRD to eventually gard chips.
- cpu: Handle opal_reinit_cpus() more gracefully on P7 no longer generate error logs
- libflash updates for openpower
- misc code cleanup
- add nx-842 coproc support

5.1.5 skiboot-5.1.0

skiboot-5.1.0 was released on August 17th, 2015.

skiboot-5.1.0 is the first stable release of 5.1.0 following two beta releases. This new stable release replaces skiboot-5.0 as the current stable skiboot release (5.0 was released April 14th 2015).

Skiboot 5.1.0 contains all fixes from skiboot-5.0 stable branch up to skiboot-5.0.5 and everything from 5.1.0-beta1 and 5.1.0-beta2.

Over skiboot-5.1.0-beta2, we have the following changes:

- opal_prd now supports multiple socket systems
- fix compiler warnings in gard and libflash

Below are the changes introduced in previous skiboot-5.1.0 releases over the previous stable release, skiboot-5.0:

New features

- Add Naples chip (CPU, PHB, LPC serial interrupts) support
- Added qemu platform
- improvements to FSI error handling
- improvements in chip TOD failover (some only on FSP systems)
- Set Relative Priority Register (RPR) to recommended value
  – this affects thread priority in SMT modes
- greatly reduce memory consumption by CPU stacks for non-present CPUs
  – Previously we would reserve enough memory for max PIR for each CPU type.
  – This fix frees up 77MB of RAM on a typical P8 system.
- increased OPAL API documentation
- Asynchronous preloading of resources from FSP/flash
  – improves boot time on some systems
- Basic Garrison platform support
- Add Mambo platform (P8 Functional Simulator, systemsim)
  – includes fake NVRAM, RTC
- Support building with GCOV, increasing memory for skiboot binary to 2MB
  – includes boot code coverage testing
- Increased skiboot HEAP size.
  – We are not aware of any system where you would run out, but on large systems it was getting closer than we liked.
- add boot_tests.sh for helping automate boot testing on FSP and BMC machines
- Versioning of pflash and gard utilities to help Linux (or other OS) distributions with packaging.
- OCC throttle status messages to host
- CAPP timebase sync (“ibm,capp-timebase-sync” in DT to indicate CAPP timebase was synced by OPAL)
- opal-api: Add OPAL call to handle abnormal reboots.

`OPAL_CEC_REBOOT2` currently supports two reboot types:

0. normal reboot, that will behave similar to that of opal_cec_reboot() call
1. platform error reboot.

Long term, this is designed to replace OPAL_CEC_REBOOT.

New features for FSP based machines

- in-band IPMI support
- ethernet adaptor location codes
- add DIMM frequency information to device tree
- improvements in FSP error log code paths
- fix some boot time memory leaks
  - harmless to end user

New features for AMI BMC based machines

- PCIe power workaround for K80
- Added support for Macronix 128Mbit flash chips
- Initial PRD support for Firestone platform
- improved reliability when BMC reboots

The following bugs have been fixed

- Increase PHB3 timeout for electrical links coming up to 2 seconds.
  - fixes issues with some Mellanox cards
- Hang in opal_reinit_cpus() that could prevent kdump from functioning
- PHB3: fix crash in phb3_init
- PHB3: fix crash with fenced PHB in phb3_init_hw()
- Fix bugs in hw/bt.c (interface for IPMI on BMC machines) that could possibly lead to a crash (dereferencing invalid address, deadlock)
- ipmi/sel: fix use-after-free
- Bug fixes in EEH handling
  - opal_pci_next_error() cleared OPAL_EVENT_PCI_ERROR unconditionally, possibly leading to missed errors.
- external/opal-prd: Only map each PRD range once
  - could eventually lead to failing to map PRD ranges
- On skiboot crash, don't try to print symbol when we didn't find one
  - makes backtrace prettier
- On skiboot crash, dump hssr0 and hsrr1 registers correctly.

- Better support old and biarch compilers
  - test "new" compiler flags before using them
  - Specify -mabi=elfv1 if supported (which means it's needed)
- fix boot-coverage-report makefile target
- ipmi: Fix the opal_ipmi_recv() call to handle the error path
- Could make kernel a sad panda when in continues with other IPMI commands
- IPMI: truncate SELs at 2kb
  - it's the limit of the astbmc. We think.
- IPMI/SEL/PEL:
  - As per PEL spec, we should log events with severity >= 0x22 and "service action flag" is "on". But in our case, all logs OPAL originagted logs are makred as report externally. We now only report logs with severity >= 0x22
- IPMI: fixes to eSEL logging
- hw/phb3: Change reserved PE to 255
  - Currently, we have reserved PE#0 to which all RIDs are mapped prior to PE assignment request from kernel. The last M64 BAR is configured to have shared mode. So we have to cut off the first M64 segment, which corresponds to reserved PE#0 in kernel. If the first BAR (for example PF's IOV BAR) requires huge alignment in kernel, we have to waste huge M64 space to accommodate the alignment. If we have reserved PE#256, the waste of M64 space will be avoided.

FSP-specific bugs fixed

- (also fixed in skiboot-5.0.2) Fix race in firenze_get_slot_info() leading to assert() with many PCI cards

  With many PCI cards, we'd hit a race where calls to firenze_add_pcidev_to_fsp_inventory would step on each other leading to memory corruption and finally an assert() in the allocator being hit during boot.
- PCIe power workaround for K80 cards
- /ibm,opal/led renamed to /ibm,opal/leds in Device Tree
  - compatible change as no FSP based systems shipped with skiboot-5.0

General improvements

- Preliminary Centaur i2c support
  - lays framework for supporting Centaur i2c
- don't run pollers on non-boot CPUs in time_wait
- improvements to opal-prd, pflash, libflash
  - including new blocklevel interface in libflash
- many minor fixes to issues found by static analysis
- improvements in FSP error log code paths
- code cleanup in memory allocator
- Don't expose individual nvram partitions in the device tree, just the whole flash device.

- build improvements for building on ppc64el host
- improvements in cpu_relax() for idle threads, needed for GCOV on large machines.
- Optimized memset() for POWER8, greatly reducing number of instructions executed for boot, which helps boot time in simulators.
- Major improvements in hello_world kernel
  - Bloat of huge 17 instruction test case reduced to 10.
- Disable bust_locks for general calls of abort()
  - Should enable better error messages during abort() when other users of LPC bus exist (e.g. flash)
- unified version numbers for bundled utilities
- external/boot_test/boot_test.sh
  - better usable for automated boot testing

Contributors

Since skiboot-5.0, we've had the following changesets:

Processed 372 csets from 27 developers 2 employers found A total of 15868 lines added, 3359 removed (delta 12509)

Developers with the most changesets

Developer	Changesets
Stewart Smith	117 (31.5%)
Jeremy Kerr	37 (9.9%)
Cyril Bur	33 (8.9%)
Vasant Hegde	32 (8.6%)
Benjamin Herrenschmidt	32 (8.6%)
Kamalesh Babulal	22 (5.9%)
Joel Stanley	12 (3.2%)
Mahesh Salgaonkar	12 (3.2%)
Alistair Popple	12 (3.2%)
Neelesh Gupta	9 (2.4%)
Gavin Shan	8 (2.2%)
Cédric Le Goater	8 (2.2%)
Ananth N Mavinakayanahalli	8 (2.2%)
Vipin K Parashar	6 (1.6%)
Michael Neuling	6 (1.6%)
Samuel Mendoza-Jonas	3 (0.8%)
Frederic Bonnard	3 (0.8%)
Andrew Donnellan	2 (0.5%)
Vaidyanathan Srinivasan	2 (0.5%)
Philippe Bergheaud	1 (0.3%)
Shilpasri G Bhat	1 (0.3%)
Daniel Axtens	1 (0.3%)
Hari Bathini	1 (0.3%)
Michael Ellerman	1 (0.3%)
Andrei Warkentin	1 (0.3%)
Dan Horák	1 (0.3%)
Anton Blanchard	1 (0.3%)

Developers with the most changed lines

Stewart Smith	4499 (27.3%)
Benjamin Herrenschmidt	3782 (22.9%)
Jeremy Kerr	1887 (11.4%)
Cyril Bur	1654 (10.0%)
Vasant Hegde	959 (5.8%)
Mahesh Salgaonkar	886 (5.4%)
Neelesh Gupta	473 (2.9%)
Samuel Mendoza-Jonas	387 (2.3%)
Vipin K Parashar	332 (2.0%)
Philippe Bergheaud	171 (1.0%)
Shilpasri G Bhat	165 (1.0%)
Alistair Popple	151 (0.9%)
Joel Stanley	105 (0.6%)
Cédric Le Goater	89 (0.5%)
Gavin Shan	83 (0.5%)
Frederic Bonnard	76 (0.5%)
Kamalesh Babulal	65 (0.4%)
Michael Neuling	46 (0.3%)
Daniel Axtens	31 (0.2%)
Andrew Donnellan	22 (0.1%)
Ananth N Mavinakayanahalli	20 (0.1%)
Anton Blanchard	3 (0.0%)
Vaidyanathan Srinivasan	2 (0.0%)
Hari Bathini	2 (0.0%)
Michael Ellerman	1 (0.0%)
Andrei Warkentin	1 (0.0%)
Dan Horák	1 (0.0%)

Developers with the most lines removed

Michael Neuling	24 (0.7%)
Hari Bathini	1 (0.0%)

Developers with the most signoffs (total 253)

Stewart Smith	249 (98.4%)
Mahesh Salgaonkar	4 (1.6%)

Developers with the most reviews (total 24)

Vasant Hegde	9 (37.5%)
Joel Stanley	3 (12.5%)
Gavin Shan	2 (8.3%)
Kamalesh Babulal	2 (8.3%)
Samuel Mendoza-Jonas	2 (8.3%)
Alistair Popple	2 (8.3%)
Stewart Smith	1 (4.2%)
Andrei Warkentin	1 (4.2%)
Preeti U Murthy	1 (4.2%)
Ananth N Mavinakayanahalli	1 (4.2%)

Developers with the most test credits (total 1)

Chad Larson	1 (100.0%)

Developers who gave the most tested-by credits (total 1)

Gavin Shan	1 (100.0%)

Developers with the most report credits (total 4)

Benjamin Herrenschmidt	2 (50.0%)
Chad Larson	1 (25.0%)
Andrei Warkentin	1 (25.0%)

Developers who gave the most report credits (total 4)

Stewart Smith	3 (75.0%)
Gavin Shan	1 (25.0%)

Top changeset contributors by employer

IBM	369 (99.2%)
(Unknown)	3 (0.8%)

Top lines changed by employer

IBM	16497 (100.0%)
(Unknown)	3 (0.0%)

Employers with the most signoffs (total 253)

IBM	253 (100.0%)

Employers with the most hackers (total 27)

IBM	24 (88.9%)
(Unknown)	3 (11.1%)

5.1.6 skiboot-5.1.0-beta1

skiboot-5.1.0-beta1 was released on July 21st, 2015.

skiboot-5.1.0-beta1 is the first beta release of skiboot 5.1, which will become a new stable release, replacing skiboot-5.0 (released April 14th 2015)

Skiboot 5.1-beta1 contains all fixes from skiboot-5.0 stable branch up to skiboot-5.0.5.

New features

Over skiboot-5.0, the following features have been added:

- Centaur i2c support
- Add Naples chip (CPU, PHB, LPC serial interrupts) support
- Added qemu platform
- improvements to FSI error handling
- improvements in chip TOD failover (some only on FSP systems)
- Set Relative Priority Register (RPR) to recommended value
  - this affects thread priority in SMT modes
- greatly reduce memory consumption by CPU stacks for non-present CPUs
  - Previously we would reserve enough memory for max PIR for each CPU type.
  - This fix frees up 77MB of RAM on a typical P8 system.
- increased OPAL API documentation
- Asynchronous preloading of resources from FSP/flash
  - improves boot time on some systems
- Basic Garrison platform support
- Add Mambo platform (P8 Functional Simulator, systemsim)
  - includes fake NVRAM, RTC
- Support building with GCOV, increasing memory for skiboot binary to 2MB
  - includes boot code coverage testing
- Increased skiboot HEAP size.
  - We are not aware of any system where you would run out, but on large systems it was getting closer than we liked.

- add boot_tests.sh for helping automate boot testing on FSP and BMC machines
- Versioning of pflash and gard utilities to help Linux (or other OS) distributions with packaging.
- OCC throttle status messages to host
- CAPP timebase sync ("ibm,capp-timebase-sync" in DT to indicate CAPP timebase was synced by OPAL)

New features for FSP based machines

- in-band IPMI support
- ethernet adaptor location codes
- add DIMM frequency information to device tree
- improvements in FSP error log code paths
- fix some boot time memory leaks
  - harmless to end user

New features for AMI BMC based machines

- PCIe power workaround for K80
- Added support for Macronix 128Mbit flash chips
- Initial PRD support for Firestone platform
- improved reliability when BMC reboots

Bug Fixes

The following bugs have been fixed:

- Increase PHB3 timeout for electrical links coming up to 2 seconds.
  - fixes issues with some Mellanox cards
- Hang in opal_reinit_cpus() that could prevent kdump from functioning
- PHB3: fix crash in phb3_init
- PHB3: fix crash with fenced PHB in phb3_init_hw()
- Fix bugs in hw/bt.c (interface for IPMI on BMC machines) that could possibly lead to a crash (dereferencing invalid address, deadlock)
- ipmi/sel: fix use-after-free
- Bug fixes in EEH handling
  - opal_pci_next_error() cleared OPAL_EVENT_PCI_ERROR unconditionally, possibly leading to missed errors.

FSP-specific bugs fixed:

- (also fixed in skiboot-5.0.2) Fix race in firenze_get_slot_info() leading to assert() with many PCI cards

  With many PCI cards, we'd hit a race where calls to firenze_add_pcidev_to_fsp_inventory would step on each other leading to memory corruption and finally an assert() in the allocator being hit during boot.
- PCIe power workaround for K80 cards
- /ibm,opal/led renamed to /ibm,opal/leds in Device Tree
  - compatible change as no FSP based systems shipped with skiboot-5.0

General improvements:

- don't run pollers on non-boot CPUs in time_wait
- improvements to opal-prd, pflash, libflash
  - including new blocklevel interface in libflash
- many minor fixes to issues found by static analysis
- improvements in FSP error log code paths
- code cleanup in memory allocator
- Don't expose individual nvram partitions in the device tree, just the whole flash device.
- build improvements for building on ppc64el host
- improvements in cpu_relax() for idle threads, needed for GCOV on large machines.
- Optimized memset() for POWER8, greatly reducing number of instructions executed for boot, which helps boot time in simulators.
- Major improvements in hello_world kernel
  - Bloat of huge 17 instruction test case reduced to 10.
- Disable bust_locks for general calls of abort()
  - Should enable better error messages during abort() when other users of LPC bus exist (e.g. flash)

Contributors

Thanks to everyone who has made skiboot-5.1.0-beta1 happen!

Processed 321 csets from 25 developers 3 employers found A total of 13696 lines added, 2754 removed (delta 10942)

Developers with the most changesets

Developer	Changesets
Stewart Smith	101 (31.5%)
Benjamin Herrenschmidt	32 (10.0%)
Cyril Bur	31 (9.7%)
Vasant Hegde	28 (8.7%)
Jeremy Kerr	27 (8.4%)
Kamalesh Babulal	19 (5.9%)
Alistair Popple	12 (3.7%)
Mahesh Salgaonkar	12 (3.7%)
Neelesh Gupta	8 (2.5%)
Cédric Le Goater	8 (2.5%)
Joel Stanley	8 (2.5%)
Ananth N Mavinakayanahalli	8 (2.5%)
Gavin Shan	6 (1.9%)
Michael Neuling	6 (1.9%)
Frederic Bonnard	3 (0.9%)
Vipin K Parashar	2 (0.6%)
Vaidyanathan Srinivasan	2 (0.6%)
Philippe Bergheaud	1 (0.3%)
Shilpasri G Bhat	1 (0.3%)
Daniel Axtens	1 (0.3%)
Hari Bathini	1 (0.3%)
Michael Ellerman	1 (0.3%)
Andrei Warkentin	1 (0.3%)
Dan Horák	1 (0.3%)
Anton Blanchard	1 (0.3%)

Developers with the most changed lines

Developer	Changed Lines
Stewart Smith	3987 (27.9%)
Benjamin Herrenschmidt	3811 (26.6%)
Cyril Bur	1918 (13.4%)
Jeremy Kerr	1307 (9.1%)
Mahesh Salgaonkar	886 (6.2%)
Vasant Hegde	764 (5.3%)
Neelesh Gupta	473 (3.3%)
Vipin K Parashar	176 (1.2%)
Alistair Popple	175 (1.2%)
Philippe Bergheaud	171 (1.2%)
Shilpasri G Bhat	165 (1.2%)
Cédric Le Goater	89 (0.6%)
Frederic Bonnard	78 (0.5%)
Gavin Shan	73 (0.5%)
Joel Stanley	65 (0.5%)
Kamalesh Babulal	63 (0.4%)
Michael Neuling	47 (0.3%)
Daniel Axtens	31 (0.2%)
Ananth N Mavinakayanahalli	22 (0.2%)
Anton Blanchard	3 (0.0%)
Vaidyanathan Srinivasan	2 (0.0%)
Hari Bathini	2 (0.0%)
Michael Ellerman	1 (0.0%)
Andrei Warkentin	1 (0.0%)
Dan Horák	1 (0.0%)

Developers with the most lines removed:

Vipin K Parashar	105 (3.8%)
Michael Neuling	24 (0.9%)
Hari Bathini	1 (0.0%)

Developers with the most signoffs (total 214)

Stewart Smith	214 (100.0%)

Developers with the most reviews (total 21)

Vasant Hegde	7 (33.3%)
Joel Stanley	3 (14.3%)
Gavin Shan	2 (9.5%)
Kamalesh Babulal	2 (9.5%)
Alistair Popple	2 (9.5%)
Stewart Smith	1 (4.8%)
Andrei Warkentin	1 (4.8%)
Preeti U Murthy	1 (4.8%)
Samuel Mendoza-Jonas	1 (4.8%)
Ananth N Mavinakayanahalli	1 (4.8%)

Developers with the most test credits (total 1)

Chad Larson	1 (100.0%)

Developers who gave the most tested-by credits (total 1)

Gavin Shan	1 (100.0%)

Developers with the most report credits (total 4)

Benjamin Herrenschmidt	2 (50.0%)
Chad Larson	1 (25.0%)
Andrei Warkentin	1 (25.0%)

Developers who gave the most report credits (total 4)

Stewart Smith	3 (75.0%)
Gavin Shan	1 (25.0%)

Top changeset contributors by employer

IBM	319 (99.4%)
dan@danny.cz	1 (0.3%)
andrey.warkentin@gmail.com	1 (0.3%)

Top lines changed by employer

IBM	14309 (100.0%)
dan@danny.cz	1 (0.0%)
andrey.warkentin@gmail.com	1 (0.0%)

Employers with the most signoffs (total 214)

IBM	214 (100.0%)

Employers with the most hackers (total 25)

IBM	23 (92.0%)
dan@danny.cz	1 (4.0%)
andrey.warkentin@gmail.com	1 (4.0%)

5.1.7 skiboot-5.1.0-beta2

skiboot-5.1.0-beta2 was released on August 14th, 2015.

skiboot-5.1.0-beta2 is the second beta release of skiboot 5.1, which will become a new stable release, replacing skiboot-5.0 (released April 14th 2015)

Skiboot 5.1.0-beta2 contains all fixes from skiboot-5.0 stable branch up to skiboot-5.0.5 and everything from 5.1.0-beta1.

New Features

Over skiboot-5.1.0-beta1, the following features have been added:

- **opal-api: Add OPAL call to handle abnormal reboots.** OPAL_CEC_REBOOT2 Currently it will support two reboot types (0). normal reboot, that will behave similar to that of opal_cec_reboot() call, and (1). platform error reboot.

  Long term, this is designed to replace OPAL_CEC_REBOOT.

Bug fixes

Over skiboot-5.1.0-beta1, the following bugs have been fixed:

- external/opal-prd: Only map each PRD range once
  - could eventually lead to failing to map PRD ranges
- On skiboot crash, don't try to print symbol when we didn't find one
  - makes backtrace prettier
- On skiboot crash, dump hssr0 and hsrr1 registers correctly.
- Better support old and biarch compilers
  - test "new" compiler flags before using them
  - Specify -mabi=elfv1 if supported (which means it's needed)
- fix boot-coverage-report makefile target
- ipmi: Fix the opal_ipmi_recv() call to handle the error path
  - Could make kernel a sad panda when in continues with other IPMI commands
- IPMI: truncate SELs at 2kb
  - it's the limit of the astbmc. We think.
- IPMI/SEL/PEL:
  - As per PEL spec, we should log events with severity >= 0x22 and "service action flag" is "on". But in our case, all logs OPAL originagted logs are makred as report externally. We now only report logs with severity >= 0x22
- IPMI: fixes to eSEL logging
- hw/phb3: Change reserved PE to 255
  - Currently, we have reserved PE#0 to which all RIDs are mapped prior to PE assignment request from kernel. The last M64 BAR is configured to have shared mode. So we have to cut off the first M64 segment, which corresponds to reserved PE#0 in kernel. If the first BAR (for example PF's IOV BAR) requires huge

alignment in kernel, we have to waste huge M64 space to accommodate the alignment. If we have reserved PE#256, the waste of M64 space will be avoided.

Other changes

- unified version numbers for bundled utilities
- external/boot_test/boot_test.sh
  - better usable for automated boot testing

5.1.8 skiboot-5.1.1

skiboot-5.1.1 was released on August 18th, 2015.

skiboot-5.1.1 is the send stable release of 5.1, it follows skiboot-5.1.0.

Skiboot 5.1.1 contains all fixes from skiboot-5.1.0 and is a minor bugfix release.

Changes

Over skiboot-5.1.0, we have the following changes:

- Fix detection of compiler options on ancient GCC (e.g. gcc 4.4, shipped with RHEL6)
- ensure the GNUC version defines for GCOV are coming from target CC rather than host CC for extract-gcov
- phb3: Continue CAPP setup even if PHB is already in CAPP mode This fixes a critical bug in CAPI support.

  CAPI requires that all faults are escalated into a fence, not a freeze. This is done by setting bits in a number of MMIO registers. phb3_set_capi_mode() calls phb3_init_capp_errors() to do this. However, if the PHB is already in CAPP mode - for example in the recovery case - phb3_set_capi_mode() will bail out early, and those registers will not be set.

  This is quite easy to verify. PCI config space access errors, for example, normally cause a freeze. On a CAPI-mode PHB, they should cause a fence. Say we have a CAPI card on PHB 0, and we inject a PCI config space error:

  ```
  echo 0x8000000000000000 > /sys/kernel/debug/powerpc/PCI0000/err_injct_inboundA;
  lspci;
  ```

  The first time we inject this, the PHB will fence and recover, but won't reset the registers. Therefore, the second time we inject it, we will incorrectly freeze, not fence.

  Worse, the recovery for the resultant EEH freeze event interacts poorly with the CAPP, triggering an EEH recovery of the PHB. The combination of the two attempted recoveries will get the PHB into an inoperable state.

5.1.9 skiboot-5.1.10

skiboot-5.1.10 was released on Friday November 13th, 2015.

skiboot-5.1.10 is the 11th stable release of 5.1, it follows skiboot-5.1.9 (which was released October 30th, 2015).

Skiboot 5.1.10 contains all fixes from skiboot-5.1.9 and is a minor bug fix release.

Over skiboot-5.1.9, we have the following change:

IBM FSP machines

- FSP: Handle Delayed Power Off initiated CEC shutdown with FSP in Reset/Reload

  In a scenario where the DPO has been initiated, but the FSP then went into reset before the CEC power down came in, OPAL may not give up the link since it may never see the PSI interrupt. So, if we are in dpo_pending and an FSP reset is detected via the DISR, give up the PSI link voluntarily.

Generic

- sensor: add a compatible property OPAL needs an extra compatible property "ibm,opal-sensor" to make module autoload work smoothly in Linux for ibmpowernv driver.
- console: Completely flush output buffer before power down and reboot Completely flush the output buffer of the console driver before power down and reboot. Implements the flushing function for uart consoles, which includes the astbmc and rhesus platforms.

  This fixes an issue where some console output is sometimes lost before power down or reboot in uart consoles. If this issue is also prevalent in other console types then it can be fixed later by adding a .flush to that driver's con_ops.

5.1.10 skiboot-5.1.11

skiboot-5.1.11 was released on Friday November 13th, 2015.

Since it was Friday 13th, we had to find a bug right after we tagged and released skiboot-5.1.10.

skiboot-5.1.11 is the 12th stable release of 5.1, it follows skiboot-5.1.10 (which was released November 13th, 2015).

Skiboot 5.1.11 contains one additional bug fix over skiboot-5.1.10.

It is:

- On IBM FSP machines, if IPMI/Serial console is not connected during shutdown or reboot, machine would enter termination state rather than shut down.

5.1.11 skiboot-5.1.12

skiboot-5.1.12 was released on Friday December 4th, 2015.

skiboot-5.1.12 is the 13th stable release of 5.1, it follows skiboot-5.1.11 (which was released November 13th, 2015).

Skiboot 5.1.12 contains bug fixes and a performance improvement.

opal-prd

- Display an explict and obvious message if running on a system that does not support opal-prd, such as an IBM FSP based POWER system, where the FSP takes on the role of opal-prd.

pflash

- Fix a missing (C) header - cherry-picked from master.

General

- Don't link with libgcc - On some toolchains, we don't have libgcc available.

POWER8 PHB (PCIe) specific

- **hw/phb3: Flush cache line after updating P/Q bits** When doing an MSI EOI, we update the P and Q bits in the IVE. That causes the corresponding cache line to be dirty in the L3 which will cause a subsequent update by the PHB (upon receiving the next MSI) to get a few retries until it gets flushed.

  We improve the situation (and thus performance) by doing a dcbf instruction to force a flush of the update we do in SW.

  This improves interrupt performance, reducing latency per interrupt. The improvement will vary by workload.

IBM FSP based machines

- FSP: Give up PSI link on shutdown This clears up some erroneous SRCs (error logs) in some situations.
- **Correctly report back Real Time Clock errors to host** Under certain rare error conditions, we could return an error code to the host OS that would cause current Linux kernels to get stuck in an infinite loop during boot. This was introduced in skiboot-5.0-rc1.

5.1.12 skiboot-5.1.13

skiboot-5.1.13 was released on Wed January 27th, 2016.

skiboot-5.1.13 is the 14th stable release of 5.1, it follows skiboot-5.1.12 (which was released December 4th, 2015). This release contains bug fixes.

General

- core/device.c: Sort nodes with name@unit names by unit
  - This gives predictable device tree ordering to the payload (usually petitboot)
  - This means that utilities such as "lspci" will always return the same ordering.
- Add OPAL_CONSOLE_FLUSH to the OPAL API uart consoles only flush output when polled. The Linux kernel calls these pollers frequently, except when in a panic state. As such, panic messages are not fully printed unless the system is configured to reboot after panic.

  This patch adds a new call to the OPAL API to flush the buffer. If the system has a uart console (i.e. BMC machines), it will incrementally flush the buffer, returning if there is more to be flushed or not. If the system has a different console, the function will have no effect. This will allow the Linux kernel to ensure that panic message have been fully printed out.

CAPI

- hmi: Identify the phb upon CAPI malfunction alert Previously, any error on a CAPI adapter would assume PHB0. This could cause issues on Firestone machines.

gard utility

- Fix displaying 'cleared' gard records When a garded component is replaced hostboot detects this and updates the gard partition.

  Previously, there was ambiguity on if the gard record ID or the whole gard record needed to be erased. This fix makes gard and hostboot agree.

firestone platform

- fix spacing in slot name The other SlotN names have no space.

5.1.13 skiboot-5.1.14

skiboot-5.1.14 was released on Wed March 9th, 2016.

skiboot-5.1.14 is the 15th stable release of 5.1, it follows skiboot-5.1.13 (which was released January 27th, 2016). This release contains a spelling fix in a log message and an added device tree property to enable older kernels (with bootloader support) to use a framebuffer that is redirected to the BMC VGA port.

As such, skiboot-5.1.14 has no advantage over skiboot-5.1.13 unless you are wanting the neat offb framebuffer trick.

Changes are:

- fsp: fix spelling of "advertise" in log message See: https://www.youtube.com/watch?v=8Gv0H-vPoDc
- Explicit 1:1 mapping in ranges properties have been added to PCI bridges. This allows a neat trick with offb and VGA ports that should probably not be told to young children.

5.1.14 skiboot-5.1.15

skiboot-5.1.15 was released on Wed March 16th, 2016.

skiboot-5.1.15 is the 16th stable release of 5.1, it follows skiboot-5.1.14 (which was released March 9th, 2016). This release contains one bug fix, a fix for a memory leak in an error path for AMI BMC based systems when logging non-severe errors. As such, it is a minor bug fix update.

5.1.15 skiboot-5.1.16

skiboot-5.1.16 was released on Friday April 29th, 2016.

skiboot-5.1.16 is the 17th stable release of 5.1, it follows skiboot-5.1.15 (which was released March 16th, 2016).

This release contains a few bug fixes and is a recommended upgrade.

Changes

PHB3 (all POWER8 platforms)

- hw/phb3: Ensure PQ bits are cleared in the IVC when masking IRQ When we mask an interrupt, we may race with another interrupt coming in from the hardware. If this occurs, the P and/or Q bit may end up being set but we never EOI/clear them. This could result in a lost interrupt or the next interrupt that comes in after re-enabling never being presented.

This fixes a bug seen with some CAPI workloads which have lots of interrupt masking at the same time as high interrupt load. The fix is not specific to CAPI though.

- **hw/phb3: Fix potential race in EOI** When we EOI we need to clear the present (P) bit in the Interrupt Vector Cache (IVC). We must clear P ensuring that any additional interrupts that come in aren't lost while also maintaining coherency with the Interrupt Vector Table (IVT).

  To do this, the hardware provides a conditional update bit in the IVC. This bit ensures that generation counts between the IVT and the IVC updates are synchronised.

  Unfortunately we never set this the bit to conditionally update the P bit in the IVC based on the generation count. Also, we didn't set what we wanted the new generation count to be if the update was successful.

FSP platforms

- OPAL:Handle mbox response with bad status:0x24 during FSP termination OPAL committed a predictive log with SRC BB822411 in some situations.

Generic

- hmi: Fix a bug where partial hmi event was reported to host. This bug fix ensures the CPU PIR is reported correctly:

```
  [  305.628283] Fatal Hypervisor Maintenance interrupt [Not recovered]
  [  305.628341]  Error detail: Malfunction Alert
  [  305.628388]     HMER: 8040000000000000
- [  305.628423]      CPU PIR: 00000000
+ [  200.123021]    CPU PIR: 000008e8
  [  305.628458]  [Unit: VSU] Logic core check stop
```

5.1.16 skiboot-5.1.17

skiboot-5.1.17 was released on Thursday 21st July 2016.

skiboot-5.1.17 is the 18th stable release of 5.1, it follows skiboot-5.1.16 (which was released April 29th, 2016).

This release contains a few minor bug fixes.

Changes

All platforms:

- Fix a few typos in user visible (OPAL log) strings
- pci: Do a dummy config write to devices to establish bus number
- Make the XSCOM engine code more resilient to errors: - hw/xscom: Reset XSCOM engine after querying sleeping core FIR - hw/xscom: Reset XSCOM engine after finite number of retries when busy - xscom: Return OPAL_WRONG_STATE on XSCOM ops if CPU is asleep

5.1.17 skiboot-5.1.18

skiboot-5.1.18 was released on Friday 26th August 2016.

skiboot-5.1.18 is the 19th stable release of 5.1, it follows skiboot-5.1.17 (which was released July 21st, 2016).

This release contains a few minor bug fixes.

Changes are:

All platforms:

- opal/hmi: Fix a TOD HMI failure during a race condition. Rare race condition which meant we wouldn't recover from TOD error
- hw/phb3: Update capi initialization sequence The capi initialization sequence was revised in a circumvention document when a 'link down' error was converted from fatal to Endpoint Recoverable. Other, non-capi, register setup was corrected even before the initial open-source release of skiboot, but a few capi-related registers were not updated then, so this patch fixes it. The point is that a link-down error detected by the UTL logic will lead to an AIB fence, so that the CAPP unit can detect the error.

FSP platforms:

- FSP/ELOG: Fix OPAL generated elog resend logic
- FSP/ELOG: Fix possible event notifier hangs
- FSP/ELOG: Disable event notification if list is not consistent
- FSP/ELOG: Fix OPAL generated elog event notification
- FSP/ELOG: Disable event notification during kexec

5.1.18 skiboot-5.1.19

skiboot-5.1.19 was released on Monday 16th January 2017.

skiboot-5.1.19 is the 20th stable release of 5.1, it follows skiboot-5.1.18 (which was released 26th August 2016).

This release contains a few minor bug fixes.

Changes are:

Generic:

- Makefile: Disable stack protector due to gcc problems
- stack: Don't recurse into __stack_chk_fail
- Makefile: Use -ffixed-r13 We did not find evidence of this ever being a problem, but this fix is good and preventative.
- Limit number of "Poller recursion detected" errors to display In some error conditions, we could spiral out of control on this and spend all of our time printing the exact same backtrace. Limit it to 16 times, because 16 is a nice number.

FSP based Systems:

- fsp: Don't recurse pollers in ibm_fsp_terminate If we were to terminate in a poller, we'd call op_display() which called pollers which hit the recursive poller warning, which ended in not much fun at all.

PCI:

- hw/phb3: set PHB retry state correctly when fresetting during a creset

- **phb3: Lock the PHB on set_xive callbacks** Those are called by the interrupts core and thus skip the locking implicit in the PCI opal calls.
- hw/{phb3, p7ioc}: Return success for freset on empty PHB OPAL_CLOSED is returned when fundamental reset is issued on the PHB who doesn't have subordinate devices (root port excluded). The kernel raises an error message, which is unnecessary. This returns OPAL_SUCCESS for this case to avoid the error message.
- hw/phb3: fix error handling in complete reset During a complete reset, when we get a timeout waiting for pending transaction in state PHB3_STATE_CRESET_WAIT_CQ, we mark the PHB as broken and return OPAL_PARAMETER. Change the return code to OPAL_HARDWARE which is way more sensible, and set the state to PHB3_STATE_FENCED so that the kernel can retry the complete reset.

5.1.19 skiboot-5.1.2

skiboot-5.1.2 was released on September 9th, 2015.

skiboot-5.1.2 is the third stable release of 5.1, it follows skiboot-5.1.1 (which was released August 18th, 2015).

Skiboot 5.1.2 contains all fixes from skiboot-5.1.1 and is a minor bugfix release.

Changes

Over skiboot-5.1.1, we have the following changes:

- phb3: Handle fence in phb3_pci_msi_check_q to fix hang

  If the PHB is fenced during phb3_pci_msi_check_q, it can get stuck in an infinite loop waiting to lock the FFI. Further, as the phb lock is held during this function it will prevent any other CPUs from dealing with the fence, leading to the entire system hanging.

  If the PHB_FFI_LOCK returns all Fs, return immediately to allow the fence to be dealt with.
- phb3: Continue CAPP setup even if PHB is already in CAPP mode This fixes a critical bug in CAPI support.
- Platform hook for terminate call
  - on assert() or other firmware failure, we will make a SEL callout on ASTBMC platforms
  - (slight) refactor of code for IBM-FSP platforms
- refactor slot naming code
- Slot names for Habanero platform
- misc improvements in userspace utilities (incl pflash, gard)
- build improvements
  - fixes for two compiler warnings were squashed in 5.1.1 commit, re-introduce the fixes.
  - misc compiler/static analysis warning fixes
- gard utility:
  - If gard tool detects the GUARD PNOR partition is corrupted, it will pro-actively re-initialize it. Modern Hostboot is more sensitive to the content of the GUARD partition in order to boot.
  - Update record clearing to match Hostboots expectations We now write ECC bytes throughout the whole partition. Without this fix, hostboot may not bring up the machine.
  - In the event of a corrupted GUARD partition so that even the first entry cannot be read, the gard utility now provides the user with the option to wipe the entirety of the GUARD partition to attempt recovery.

- opal_prd utility:
  - Add run command to pass through commands to HostBoot RunTime (HBRT)
    - this is for OpenPower firmware developers only.
  - Add htmght-passthru command.
    - this is for OpenPower firmware developers only.
  - Add override interface to pass attribute-override information to HBRT.
  - Server sends response in error path, so that client doesn't block forever
- external/mambo tcl scripts
  - Running little-endian kernels in mambo requires HILE to be set properly, which requires a bump in the machine's pvr value to a DD2.x chip.

Stats

For skiboot-5.1.0 to 5.1.2: Processed 67 csets from 11 developers 1 employers found A total of 2258 lines added, 784 removed (delta 1474)

Developers with the most changesets

Stewart Smith	24 (35.8%)
Cyril Bur	18 (26.9%)
Vasant Hegde	8 (11.9%)
Neelesh Gupta	5 (7.5%)
Benjamin Herrenschmidt	5 (7.5%)
Daniel Axtens	2 (3.0%)
Samuel Mendoza-Jonas	1 (1.5%)
Vaidyanathan Srinivasan	1 (1.5%)
Vipin K Parashar	1 (1.5%)
Ian Munsie	1 (1.5%)
Michael Neuling	1 (1.5%)

Developers with the most changed lines

Cyril Bur	969 (42.5%)
Neelesh Gupta	433 (19.0%)
Benjamin Herrenschmidt	304 (13.3%)
Vasant Hegde	236 (10.3%)
Stewart Smith	163 (7.1%)
Vaidyanathan Srinivasan	135 (5.9%)
Vipin K Parashar	8 (0.4%)
Ian Munsie	8 (0.4%)
Daniel Axtens	2 (0.1%)
Michael Neuling	2 (0.1%)
Samuel Mendoza-Jonas	1 (0.0%)

Developers with the most lines removed

Daniel Axtens	2 (0.3%)
Michael Neuling	1 (0.1%)

Developers with the most signoffs (total 44)

Stewart Smith	43 (97.7%)
Neelesh Gupta	1 (2.3%)

Developers with the most reviews (total 8)

Patrick Williams	5 (62.5%)
Samuel Mendoza-Jonas	3 (37.5%)

Developers with the most test credits (total 0)

Developers who gave the most tested-by credits (total 0)

Developers with the most report credits (total 1)

Benjamin Herrenschmidt	1 (100.0%)

Developers who gave the most report credits (total 1)

Samuel Mendoza-Jonas	1 (100.0%)

Top changeset contributors by employer

IBM	67 (100.0%)

Top lines changed by employer

IBM	2281 (100.0%)

Employers with the most signoffs (total 44)

IBM	44 (100.0%)

Employers with the most hackers (total 11)

IBM	11 (100.0%)

5.1.20 skiboot-5.1.20

skiboot-5.1.20 was released on Friday 18th August 2017.

skiboot-5.1.20 is the 21st stable release of 5.1, it follows skiboot-5.1.19 (which was released 16th January 2017).

This release contains a few minor bug fixes backported to the 5.1.x series. All of the fixes have previously appeared in the 5.4.x stable series.

Changes are:

- FSP/CONSOLE: Workaround for unresponsive ipmi daemon

  In some corner cases, where FSP is active but not responding to console MBOX message (due to buggy IPMI) and we have heavy console write happening from kernel, then eventually our console buffer becomes full. At this point OPAL starts sending OPAL_BUSY_EVENT to kernel. Kernel will keep on retrying. This is creating kernel soft lockups. In some extreme case when every CPU is trying to write to console, user will not be able to ssh and thinks system is hang.

  If we reset FSP or restart IPMI daemon on FSP, system recovers and everything becomes normal.

  This patch adds workaround to above issue by returning OPAL_HARDWARE when cosole is full. Side effect of this patch is, we may endup dropping latest console data. But better to drop console data than system hang.

  Alternative approach is to drop old data from console buffer, make space for new data. But in normal condition only FSP can update 'next_out' pointer and if we touch that pointer, it may introduce some other race conditions. Hence we decided to just new console write request.

- FSP: Set status field in response message for timed out message

  For timed out FSP messages, we set message status as "fsp_msg_timeout". But most FSP driver users (like surviellance) are ignoring this field. They always look for FSP returned status value in callback function (second byte in word1). So we endup treating timed out message as success response from FSP.

  Sample output:

  ```
  [69902.432509048,7] SURV: Sending the heartbeat command to FSP
  [70023.226860117,4] FSP: Response from FSP timed out, word0 = d66a00d7, word1 = 0␣
  ↪state: 3
  ....
  [70023.226901445,7] SURV: Received heartbeat acknowledge from FSP
  [70023.226903251,3] FSP: fsp_trigger_reset() entry
  ```

  Here SURV code thought it got valid response from FSP. But actually we didn't receive response from FSP.

- FSP: Improve timeout message

  Presently we print word0 and word1 in error log. word0 contains sequence number and command class. One has to understand word0 format to identify command class.

  Lets explicitly print command class, sub command etc.

- FSP/RTC: Remove local fsp_in_reset variable

  Now that we are using fsp_in_rr() to detect FSP reset/reload, fsp_in_reset become redundant. Lets remove this local variable.

- FSP/RTC: Fix possible FSP R/R issue in rtc write path

  fsp_opal_rtc_write() checks FSP status before queueing message to FSP. But if FSP R/R starts before getting response to queued message then we will continue to return OPAL_BUSY_EVENT to host. In some extreme condition host may experience hang. Once FSP is back we will repost message, get response from FSP and return OPAL_SUCCESS to host.

This patch caches new values and returns OPAL_SUCCESS if FSP R/R is happening. And once FSP is back we will send cached value to FSP.

- hw/fsp/rtc: read/write cached rtc tod on fsp hir.

  Currently fsp-rtc reads/writes the cached RTC TOD on an fsp reset. Use latest fsp_in_rr() function to properly read the cached rtc value when fsp reset initiated by the hir.

  Below is the kernel trace when we set hw clock, when hir process starts.

```
[ 1727.775824] NMI watchdog: BUG: soft lockup - CPU#57 stuck for 23s!␣
↪[hwclock:7688]
[ 1727.775856] Modules linked in: vmx_crypto ibmpowernv ipmi_powernv uio_pdrv_
↪genirq ipmi_devintf powernv_op_panel uio ipmi_msghandler powernv_rng leds_
↪powernv ip_tables x_tables autofs4 ses enclosure scsi_transport_sas crc32c_
↪vpmsum lpfc ipr tg3 scsi_transport_fc
[ 1727.775883] CPU: 57 PID: 7688 Comm: hwclock Not tainted 4.10.0-14-generic #16-
↪Ubuntu
[ 1727.775883] task: c000000fdfdc8400 task.stack: c000000fdfef4000
[ 1727.775884] NIP: c00000000090540c LR: c0000000000846f4 CTR: 000000003006dd70
[ 1727.775885] REGS: c000000fdfef79a0 TRAP: 0901   Not tainted  (4.10.0-14-
↪generic)
[ 1727.775886] MSR: 9000000000009033 <SF,HV,EE,ME,IR,DR,RI,LE>
[ 1727.775889]   CR: 28024442  XER: 20000000
[ 1727.775890] CFAR: c00000000008472c SOFTE: 1
               GPR00: 0000000030005128 c000000fdfef7c20 c00000000144c900␣
↪fffffffffffffff4
               GPR04: 0000000028024442 c00000000090540c 9000000000009033␣
↪0000000000000000
               GPR08: 0000000000000000 0000000031fc4000 c000000000084710␣
↪9000000000001003
               GPR12: c0000000000846e8 c00000000fba0100
[ 1727.775897] NIP [c00000000090540c] opal_set_rtc_time+0x4c/0xb0
[ 1727.775899] LR [c0000000000846f4] opal_return+0xc/0x48
[ 1727.775899] Call Trace:
[ 1727.775900] [c000000fdfef7c20] [c00000000090540c] opal_set_rtc_time+0x4c/0xb0␣
↪(unreliable)
[ 1727.775901] [c000000fdfef7c60] [c000000000900828] rtc_set_time+0xb8/0x1b0
[ 1727.775903] [c000000fdfef7ca0] [c000000000902364] rtc_dev_ioctl+0x454/0x630
[ 1727.775904] [c000000fdfef7d40] [c00000000035b1f4] do_vfs_ioctl+0xd4/0x8c0
[ 1727.775906] [c000000fdfef7de0] [c00000000035bab4] SyS_ioctl+0xd4/0xf0
[ 1727.775907] [c000000fdfef7e30] [c00000000000b184] system_call+0x38/0xe0
[ 1727.775908] Instruction dump:
[ 1727.775909] f821ffc1 39200000 7c832378 91210028 38a10020 39200000 38810028␣
↪f9210020
[ 1727.775911] 4bfffe6d e8810020 80610028 4b77f61d <60000000> 7c7f1b78 3860000a␣
↪2fbffff4
```

  This is found when executing the op-test-framework fspresetReload testcase

  With this fix ran fsp hir torture testcase in the above test which is working fine.

- FSP/CHIPTOD: Return false in error path
- On FSP platforms: notify FSP of Platform Log ID after Host Initiated Reset Reload Trigging a Host Initiated Reset (when the host detects the FSP has gone out to lunch and should be rebooted), would cause “Unknown Command” messages to appear in the OPAL log.

  This patch implements those messages.

  Log showing unknown command:

```
/ # cat /sys/firmware/opal/msglog | grep -i ,3
[  110.232114723,3] FSP: fsp_trigger_reset() entry
[  188.431793837,3] FSP #0: Link down, starting R&R
[  464.109239162,3] FSP #0: Got XUP with no pending message !
[  466.340598554,3] FSP-DPO: Unknown command 0xce0900
[  466.340600126,3] FSP: Unhandled message ce0900
```

- hw/i2c: Fix early lock drop

  When interacting with an I2C master the p8-i2c driver (common to p9) aquires a per-master lock which it holds for the duration of it's interaction with the master. Unfortunately, when p8_i2c_check_initial_status() detects that the master is busy with another transaction it drops the lock and returns OPAL_BUSY. This is contrary to the driver's locking strategy which requires that the caller aquire and drop the lock. This leads to a crash due to the double unlock(), which skiboot treats as fatal.

- head.S: store all of LR and CTR

  When saving the CTR and LR registers the skiboot exception handlers use the 'stw' instruction which only saves the lower 32 bits of the register. Given these are both 64 bit registers this leads to some strange register dumps, for example:

```
***********************************************
Unexpected exception 200 !
SRR0 : 0000000030016968 SRR1 : 9000000000201000
HSRR0: 0000000000000180 HSRR1: 9000000000001000
LR   : 3003438830823f50 CTR  : 3003438800000018
CFAR : 00000000300168fc
CR   : 40004208  XER: 00000000
```

  In this dump the upper 32 bits of LR and CTR are actually stack gunk which obscures the underlying issue.

- hw/fsp: Do not queue SP and SPCN class messages during reset/reload In certain cases of communicating with the FSP (e.g. sensors), the OPAL FSP driver returns a default code (async completion) even though there is no known bound from the time of this error return to the actual data being available. The kernel driver keeps waiting leading to soft-lockup on the host side.

  Mitigate both these (known) cases by returning OPAL_BUSY so the host driver knows to retry later.

5.1.21 skiboot-5.1.21

skiboot-5.1.21 was released on Tuesday 19th September 2017.

skiboot-5.1.21 is the 22nd stable release of 5.1, it follows skiboot-5.1.20 (which was released 18th August 2017).

This release contains one backported bug fix to the 5.1.x series.

Changes are:

- FSP: Add check to detect FSP Reset/Reload inside fsp_sync_msg()

  During FSP Reset/Reload we move outstanding MBOX messages from msgq to rr_queue including inflight message (fsp_reset_cmdclass()). But we are not resetting inflight message state.

  In extreme corner case where we sent message to FSP via fsp_sync_msg() path and FSP Reset/Reload happens before getting respose from FSP, then we will endup waiting in fsp_sync_msg() until everything becomes normal.

  This patch adds fsp_in_rr() check to fsp_sync_msg() and return error to caller if FSP is in R/R.

5.1.22 skiboot-5.1.3

skiboot-5.1.3 was released on September 15th, 2015.

skiboot-5.1.3 is the 4th stable release of 5.1, it follows skiboot-5.1.2 (which was released September 9th, 2015).

Skiboot 5.1.3 contains all fixes from skiboot-5.1.2 and is a minor bugfix release.

Changes

Over skiboot-5.1.2, we have the following changes:

- slot names for firestone platform
- fix display of LPC errors
- SBE based timer support
  - on supported platforms limits reliance on Linux heartbeat
- fix use-after-free in fsp/ipmi
- fix hang on TOD/TB errors (time-of-day/timebase) on OpenPower systems
  - On getting a Hypervizor Maintenance Interrupt to get the timebase back into a running state, we would call prlog which would use the LPC UART console driver on OpenPower systems, which depends on a working timebase, leading to a hang. We now don't depend on a working timebase in this recovery codepath.
- enable prd for garrison platform
- PCI: Clear error bits after changing MPS Chaning MPS on PCI upstream bridge might cause error bits set on downstream endpoints when system boots into Linux as below case shows:

```
host# lspci -vvs 0001:06:00.0
0001:06:00.0 Ethernet controller: Broadcom Corporation \
             NetXtreme II BCM57810 10 Gigabit Ethernet (rev 10)
DevSta:      CorrErr+ UncorrErr- FatalErr- UnsuppReq+ AuxPwr- TransPend-
CESta:       RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr+
```

  This clears those error bits in AER and PCIe capability after MPS is changed. With the patch applied, no more error bits are seen.

Contributors

Processed 14 csets from 6 developers 1 employers found A total of 462 lines added, 163 removed (delta 299)

Developers with the most changesets

Benjamin Herrenschmidt	5 (35.7%)
Stewart Smith	4 (28.6%)
Mahesh Salgaonkar	2 (14.3%)
Gavin Shan	1 (7.1%)
Jeremy Kerr	1 (7.1%)
Neelesh Gupta	1 (7.1%)

Developers with the most changed lines

Benjamin Herrenschmidt	407 (80.8%)
Mahesh Salgaonkar	23 (4.6%)
Gavin Shan	19 (3.8%)
Stewart Smith	18 (3.6%)
Jeremy Kerr	5 (1.0%)
Neelesh Gupta	2 (0.4%)

Developers with the most lines removed

Stewart Smith	8 (4.9%)
Jeremy Kerr	3 (1.8%)
Neelesh Gupta	1 (0.6%)

Developers with the most signoffs (total 10)

Stewart Smith	10 (100.0%)

Developers with the most reviews (total 1)

Joel Stanley	1 (100.0%)

Developers with the most test credits (total 0)

Developers who gave the most tested-by credits (total 0)

Developers with the most report credits (total 1)

John Walthour	1 (100.0%)

Developers who gave the most report credits (total 1)

Gavin Shan	1 (100.0%)

Top changeset contributors by employer

IBM	14 (100.0%)

Top lines changed by employer

IBM	504 (100.0%)

Employers with the most signoffs (total 10)

IBM	10 (100.0%)

Employers with the most hackers (total 6)

IBM	6 (100.0%)

5.1.23 skiboot-5.1.4

skiboot-5.1.4 was released on September 26th, 2015.

skiboot-5.1.4 is the 5th stable release of 5.1, it follows skiboot-5.1.3 (which was released September 15th, 2015).

Skiboot 5.1.4 contains all fixes from skiboot-5.1.3 and is an important bug fix release and a strongly recommended update from any prior skiboot-5.1.x release.

Changes

Over skiboot-5.1.3, we have the following changes:

- Rate limit OPAL_MSG_OCC to only one outstanding message to host

  In the event of a lot of OCC events (or many CPU cores), we could send many OCC messages to the host, which if it wasn't calling opal_get_msg really often, would cause skiboot to malloc() additional messages until we ran out of skiboot heap and things didn't end up being much fun.

  When running certain hardware exercisers, they seem to steal all time from Linux being able to call opal_get_msg, causing these to queue up and get "opalmsg: No available node in the free list, allocating" warnings followed by tonnes of backtraces of failing memory allocations.

- Ensure reserved memory ranges are exposed correctly to host (fix corrupted SLW image)

  We seem to have not hit this on ASTBMC based OpenPower machines, but was certainly hit on FSP based machines

5.1.24 skiboot-5.1.5

skiboot-5.1.5 was released on October 1st, 2015.

skiboot-5.1.5 is the 6th stable release of 5.1, it follows skiboot-5.1.4 (which was released September 26th, 2015).

Skiboot 5.1.5 contains all fixes from skiboot-5.1.4 and is a minor bug fix release.

Changes

Over skiboot-5.1.4, we have the following changes:

Generic

- centaur: Add indirect XSCOM support Fixes a bug where opal-prd would not be able to recover from a bunch of errors as the indirect XSCOMs to centaurs would fail.
- xscom: Fix logging of indirect XSCOM errors Better logging of error messages.

- PHB3: Fix wrong PE number in error injection
- Improvement in boot_test.sh utility to support copying a pflash binary to BMCs.

AST BMC machines

- **ipmi-sel: Run power action immediately if host not up** Our normal sequence for a soft power action (IPMI 'power soft' or 'power cycle') involve receiving a SEL from the BMC, sending a message to Linux's opal platform support which instructs the host OS to shut down, and finally the host will request OPAL to cut power.

  When the host is not yet up we will send the message to /dev/null, and no action will be taken. This patches changes that behaviour to perform the action immediately if we know how.

OpenPower machines:

- opal-prd: Increase IPMI timeout to a slightly better value Proactively bump the timeout to 5seconds to match current value in petitboot Observed in the wild that this fixes bugs for petitboot.

5.1.25 skiboot-5.1.6

skiboot-5.1.6 was released on October 8th, 2015.

skiboot-5.1.6 is the 7th stable release of 5.1, it follows skiboot-5.1.5 (which was released October 1st, 2015).

Skiboot 5.1.6 contains all fixes from skiboot-5.1.5 and is a minor bug fix release.

Changes

Over skiboot-5.1.5, we have the following changes:

Generic:

- Ensure we run pollers in cpu_wait_job()

  In root causing a bug on AST BMC Alistair found that pollers weren't being run for around 3800ms.

  This could show as not resetting the boot count sensor on successful boot.

AST BMC Machines

- hw/bt.c: Check for timeout after checking for message response

  When deciding if a BT message has timed out we should first check for a message response. This will ensure that messages will not time out if there was a delay calling the pollers.

  This could show as not resetting the boot count sensor on successful boot.

5.1.26 skiboot-5.1.7

skiboot-5.1.7 was released on October 13th, 2015.

skiboot-5.1.7 is the 8th stable release of 5.1, it follows skiboot-5.1.6 (which was released October 8th, 2015).

Skiboot 5.1.7 contains all fixes from skiboot-5.1.6 and is a minor bug fix release with one important bug fix for FSP systems.

Over skiboot-5.1.6, we have the following changes:

Generic:

- **PHB3: Retry fundamental reset** This introduces another PHB3 state (PHB3_STATE_FRESET_START) allowing to redo fundamental reset if the link doesn't come up in time at the first attempt, to improve the robustness of PHB's fundamental reset. If the link comes up after the first reset, the 2nd reset won't be issued at all.

FSP based systems:

- hw/fsp/fsp-leds.c: use allocated buffer for FSP_CMD_GET_LED_LIST response

  This fixes a bug where we would overwrite roughly 4kb of memory belonging to Linux when the FSP would ask firmware for a list of LEDs in the system. This wouldn't happen often (once before Linux was running and possibly only once during runtime, and *early* runtime at that) but it was possible for this corruption to show up and be detected.

5.1.27 skiboot-5.1.8

skiboot-5.1.8 was released on October 19th, 2015.

skiboot-5.1.8 is the 9th stable release of 5.1, it follows skiboot-5.1.7 (which was released October 13th, 2015).

Skiboot 5.1.8 contains all fixes from skiboot-5.1.7 and is a minor bug fix release, with a single fix for recovery from a (rare) error.

Over skiboot-5.1.7, we have the following change:

- opal/hmi: Fix a soft lockup issue on Hypervisor Maintenance Interrupt for certain timebase errors.

  We also introduce a timeout to handle the worst situation where all other threads are badly stuck without setting a cleanup done bit. Under such situation timeout will help to avoid soft lockups and report failure to kernel.

5.1.28 skiboot-5.1.9

skiboot-5.1.9 was released on October 30th, 2015.

skiboot-5.1.9 is the 10th stable release of 5.1, it follows skiboot-5.1.8 (which was released October 19th, 2015).

Skiboot 5.1.9 contains all fixes from skiboot-5.1.8 and is a minor bug fix release, with a single fix to help diagnosis after a rare error condition.

Over skiboot-5.1.8, we have the following change:

- opal/hmi: Signal PRD about NX unit checkstop. We now signal Processor Recovery & Diagnostics (PRD) correctly following an NX unit checkstop
- minor fix to the boot_test.sh test script

5.1.29 skiboot-5.10

skiboot v5.10 was released on Friday February 23rd 2018. It is the first release of skiboot 5.10, and becomes the new stable release of skiboot following the 5.9 release, first released October 31st 2017.

skiboot v5.10 contains all bug fixes as of *skiboot-5.9.8* and *skiboot-5.4.9*. We do not forsee any further 5.9.x releases.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

Over skiboot-5.9, we have the following changes:

New Features

Since skiboot-5.10-rc3:

- sensor-groups: occ: Add support to disable/enable sensor group

  This patch adds a new opal call to enable/disable a sensor group. This call is used to select the sensor groups that needs to be copied to main memory by OCC at runtime.

- sensors: occ: Add energy counters

  Export the accumulated power values as energy sensors. The accumulator field of power sensors are used for representing energy counters which can be exported as energy counters in Linux hwmon interface.

- sensors: Support reading u64 sensor values

  This patch adds support to read u64 sensor values. This also adds changes to the core and the backend implementation code to make this API as the base call. Host can use this new API to read sensors upto 64bits.

  This adds a list to store the pointer to the kernel u32 buffer, for older kernels making async sensor u32 reads.

- dt: add /cpus/ibm,powerpc-cpu-features device tree bindings

  This is a new CPU feature advertising interface that is fine-grained, extensible, aware of privilege levels, and gives control of features to all levels of the stack (firmware, hypervisor, and OS).

  The design and binding specification is described in detail in doc/.

Since skiboot-5.10-rc2:

- DT: Add "version" property under ibm, firmware-versions node

  First line of VERSION section in PNOR contains firmware version. Use that to add "version" property under firmware versions dt node.

  Sample output:

  ```
  root@xxx2:/proc/device-tree/ibm,firmware-versions# lsprop
  version          "witherspoon-ibm-OP9_v1.19_1.94"
  ```

Since skiboot-5.10-rc1:

- hw/npu2: Implement logging HMI actions

Since skiboot-5.9:

- hdata: Parse IPL FW feature settings

  Add parsing for the firmware feature flags in the HDAT. This indicates the settings of various parameters which are set at IPL time by firmware.

- opal/xstop: Use nvram option to enable/disable sw checkstop.

  Add a mechanism to enable/disable sw checkstop by looking at nvram option opal-sw-xstop=<enable/disable>.

  For now this patch disables the sw checkstop trigger unless explicitly enabled through nvram option 'opal-sw-xstop=enable'i for p9. This will allow an opportunity to get host kernel in panic path or xmon for unrecoverable HMIs or MCE, to be able to debug the issue effectively.

  To enable sw checkstop in opal issue following command:

  ```
  nvram -p ibm,skiboot --update-config opal-sw-xstop=enable
  ```

  NOTE: This is a workaround patch to disable sw checkstop by default to gain control in host kernel for better checkstop debugging. Once we have most of the checkstop issues stabilized/resolved, revisit this patch to enable sw checkstop by default.

  For p8 platform it will remain enabled by default unless explicitly disabled.

  To disable sw checkstop on p8 issue following command:

  ```
  nvram -p ibm,skiboot --update-config opal-sw-xstop=disable
  ```

- hdata: Parse SPD data

    Parse SPD data and populate device tree.

    list of properties parsing from SPD:

    ```
    [root@ltc-wspoon dimm@d00f]# lsprop .
    memory-id        0000000c (12)       # DIMM type
    product-version  00000032 (50)       # Module Revision Code
    device_type      "memory-dimm-ddr4"
    serial-number    15d9acb6 (366587062)
    status           "okay"
    size             00004000 (16384)
    phandle          000000bd (189)
    ibm,loc-code     "UOPWR.0000000-Node0-DIMM7"
    part-number      "36ASF2G72PZ-2G6B2    "
    reg              0000d007 (53255)
    name             "dimm"
    manufacturer-id  0000802c (32812)   # Vendor ID, we can get vendor name␣
    ↪from this ID
    ```

    Also update documentation.

- hdata: Add memory hierarchy under xscom node

  We have memory to chip mapping but doesn't have complete memory hierarchy. This patch adds memory hierarchy under xscom node. This is specific to P9 system as these hierarchy may change between processor generation.

  It uses memory controller ID details and populates nodes like: xscom@<addr>/mcbist@<mcbist_id>/mcs@<mcs_id>/mca@

  Also this patch adds few properties under dimm node. Finally make sure xscom nodes created before calling memory_parse().

Fast Reboot and Quiesce

We have a preliminary fast reboot implementation for POWER9 systems, which we look to enabling by default in the next release.

The OPAL Quiesce calls are designed to improve reliability and debuggability around reboot and error conditions. See the full API documentation for details: *OPAL_QUIESCE*.

- fast-reboot: bare bones fast reboot implementation for POWER9

  This is an initial fast reboot implementation for p9 which has only been tested on the Witherspoon platform, and without the use of NPUs, NX/VAS, etc.

  This has worked reasonably well so far, with no failures in about 100 reboots. It is hidden behind the traditional fast-reboot experimental nvram option, until more platforms and configurations are tested.

- fast-reboot: move boot CPU clean-up logically together with secondaries

  Move the boot CPU clean-up and state transition to active, logically together with secondaries. Don't release secondaries from fast reboot hold until everyone has cleaned up and transitioned to active.

  This is cosmetic, but it is helpful to run the fast reboot state machine the same way on all CPUs.

- fast-reboot: improve failure error messages

  Change existing failure error messages to PR_NOTICE so they get printed to the console, and add some new ones. It's not a more severe class because it falls back to IPL on failure.

- fast-reboot: quiesce opal before initiating a fast reboot

  Switch fast reboot to use quiescing rather than "wait for a while".

  If firmware can not be quiesced, then fast reboot is skipped. This significantly improves the robustness of fast reboot in the face of bugs or unexpected latencies.

  Complexity of synchronization in fast-reboot is reduced, because we are guaranteed to be single-threaded when quiesce succeeds, so locks can be removed.

  In the case that firmware can be quiesced, then it will generally reduce fast reboot times by nearly 200ms, because quiescing usually takes very little time.

- core: Add support for quiescing OPAL

  Quiescing is ensuring all host controlled CPUs (except the current one) are out of OPAL and prevented from entering. This can be use in debug and shutdown paths, particularly with system reset sequences.

  This patch adds per-CPU entry and exit tracking for OPAL calls, and adds logic to "hold" or "reject" at entry time, if OPAL is quiesced.

  An OPAL call is added, to expose the functionality to Linux, where it can be used for shutdown, kexec, and before generating sreset IPIs for debugging (so the debug code does not recurse into OPAL).

- dctl: p9 increase thread quiesce timeout

  We require all instructions to be completed before a thread is considered stopped, by the dctl interface. Long running instructions like cache misses and CI loads may take a significant amount of time to complete, and timeouts have been observed in stress testing.

  Increase the timeout significantly, to cover this. The workbook just says to poll, but we like to have timeouts to avoid getting stuck in firmware.

POWER9 power saving

There is much improved support for deeper sleep/idle (stop) states on POWER9.

- OCC: Increase max pstate check on P9 to 255

  This has changed from P8, we can now have > 127 pstates.

This was observed on Boston during WoF bring up.

- SLW: Add idle state stop5 for DD2.0 and above

  Adding stop5 idle state with rough residency and latency numbers.

- SLW: Add p9_stop_api calls for IMC

  Add p9_stop_api for EVENT_MASK and PDBAR scoms. These scoms are lost on wakeup from stop11.

- SCOM restore for DARN and XIVE

  While waking up from stop11, we want NCU_DARN_BAR to have enable bit set. Without this stop_api call, the value restored is without enable bit set. We loose NCU_SPEC_BAR when the quad goes into stop11, stop_api will restore while waking up from stop11.

- SLW: Call p9_stop_api only if deep_states are enabled

  All init time p9_stop_api calls have been isolated to slw_late_init. If p9_stop_api fails, then the deep states can be excluded from device tree.

  For p9_stop_api called after device-tree for cpuidle is created , has_deep_states will be used to check if this call is even required.

- Better handle errors in setting up sleep states (p9_stop_api)

  We won't put affected stop states in the device tree if the wakeup engine is not present or has failed.

- SCOM Restore: Increased the EQ SCOM restore limit.

  Commit increases the SCOM restore limit from 16 to 31.

- hw/dts: retry special wakeup operation if core still gated

  It has been observed that in some cases the special wakeup operation can "succeed" but the core is still in a gated/offline state.

  Check for this state after attempting to wakeup a core and retry the wakeup if necessary.

- core/direct-controls: add function to read core gated state
- core/direct-controls: wait for core special wkup bit cleared

  When clearing special wakeup bit on a core, wait until the bit is actually cleared by the hardware in the status register until returning success.

  This may help avoid issues with back-to-back reads where the special wakeup request is cleared but the firmware is still processing the request and the next attempt to set the bit reads an immediate success from the previous operation.

- p9_stop_api: PM: Added support for version control in SCOM restore entries.
  - adds version info in SCOM restore entry header
  - adds version specific details in SCOM restore entry header
  - retains old behaviour of SGPE Hcode's base version
- p9_stop_api: EQ SCOM Restore: Introduced version control in SCOM restore entry.
  - introduces version control in header of SCOM restore entry
  - ensures backward compatibility
  - introduces flexibility to handle any number of SCOM restore entry.

Secure and Trusted Boot for POWER9

We introduce support for Secure and Trusted Boot for POWER9 systems, with equal functionality that we have on POWER8 systems, that is, we have the mechanisms in place to boot to petitboot (i.e. to BOOTKERNEL).

See the *Secure and Trusted Boot Library (LibSTB) Documentation* for full documentation of OPAL secure and trusted boot.

Since skiboot-5.10-rc2:

- stb: Put correct label (for skiboot) into container

  Hostboot will expect the label field of the stb header to contain "PAYLOAD" for skiboot or it will fail to load and run skiboot.

  The failure looks something like this:

```
53.40896|ISTEP 20. 1 - host_load_payload
53.65840|secure|Secureboot Failure plid = 0x90000755, rc = 0x1E07

53.65881|System shutting down with error status 0x1E07
53.67547|================================================
53.67954|Error reported by secure (0x1E00) PLID 0x90000755
53.67560|  Container's component ID does not match expected component ID
53.67561|  ModuleId   0x09 SECUREBOOT::MOD_SECURE_VERIFY_COMPONENT
53.67845|  ReasonCode 0x1e07 SECUREBOOT::RC_ROM_VERIFY
53.67998|  UserData1   : 0x0000000000000000
53.67999|  UserData2   : 0x0000000000000000
53.67999|------------------------------------------------
53.68000|  Callout type             : Procedure Callout
53.68000|  Procedure                : EPUB_PRC_HB_CODE
53.68001|  Priority                 : SRCI_PRIORITY_HIGH
53.68001|------------------------------------------------
53.68002|  Callout type             : Procedure Callout
53.68003|  Procedure                : EPUB_PRC_FW_VERIFICATION_ERR
53.68003|  Priority                 : SRCI_PRIORITY_HIGH
53.68004|------------------------------------------------
```

Since skiboot-5.10-rc1:

- stb: Enforce secure boot if called before libstb initialized
- stb: Correctly error out when no PCR for resource
- core/init: move imc catalog preload init after the STB init.

  As a safer side move the imc catalog preload after the STB init to make sure the imc catalog resource get's verified and measured properly during loading when both secure and trusted boot modes are on.

- libstb: fix failure of calling trusted measure without STB initialization.

  When we load a flash resource during OPAL init, STB calls trusted measure to measure the given resource. There is a situation when a flash gets loaded before STB initialization then trusted measure cannot measure properly.

  So this patch fixes this issue by calling trusted measure only if the corresponding trusted init was done.

  The ideal fix is to make sure STB init done at the first place during init and then do the loading of flash resources, by that way STB can properly verify and measure the all resources.

- libstb: fix failure of calling cvc verify without STB initialization.

Currently in OPAL init time at various stages we are loading various PNOR partition containers from the flash device. When we load a flash resource STB calls the CVC verify and trusted measure(sha512) functions. So when we have a flash resource gets loaded before STB initialization, then cvc verify function fails to start the verify and enforce the boot.

Below is one of the example failure where our VERSION partition gets loading early in the boot stage without STB initialization done.

This is with secure mode off. STB: VERSION NOT VERIFIED, invalid param. buf=0x305ed930, len=4096 key-hash=0x0 hash-size=0

In the same code path when secure mode is on, the boot process will abort.

So this patch fixes this issue by calling cvc verify only if we have STB init was done.

And also we need a permanent fix in init path to ensure STB init gets done at first place and then start loading all other flash resources.

- libstb/tpm_chip: Add missing new line to print messages.
- libstb: increase the log level of verify/measure messages to PR_NOTICE.

  Currently libstb logs the verify and hash caluculation messages in PR_INFO level. So when there is a secure boot enforcement happens in loading last flash resource(Ex: BOOTKERNEL), the previous verify and measure messages are not logged to console, which is not clear to the end user which resource is verified and measured. So this patch fixes this by increasing the log level to PR_NOTICE.

Since skiboot-5.9:

- allow secure boot if not enforcing it

  We check the secure boot containers no matter what, only *enforcing* secure boot if we're booting in secure mode. This gives us an extra layer of checking firmware is legit even when secure mode isn't enabled, as well as being really useful for testing.

- libstb/(create|print)-container: Sync with sb-signing-utils

  The sb-signing-utils project has improved upon the skeleton create-container tool that existed in skiboot, including being able to (quite easily) create *signed* images.

  This commit brings in that code (and makes it build in the skiboot build environment) and updates our skiboot.*.stb generating code to use the development keys. This means that by default, skiboot build process will let you build firmware that can do a secure boot with *development* keys.

  See *Signing Firmware Code* for details on firmware signing.

  We also update print-container as well, syncing it with the upstream project.

  Derived from github.com:open-power/sb-signing-utils.git at v0.3-5-gcb111c03ad7f (Some discussion ongoing on the changes, another sync will come shortly)

- doc: update libstb documentation with POWER9 changes. See: *Secure and Trusted Boot Library (LibSTB) Documentation*.

  POWER9 changes reflected in the libstb:

  – bumped ibm,secureboot node to v2
  – added ibm,cvc node
  – hash-algo superseded by hw-key-hash-size

- libstb/cvc: update memory-region to point to /reserved-memory

  The linux documentation, reserved-memory.txt, says that memory-region is a phandle that pairs to a children of /reserved-memory.

This updates /ibm,secureboot/ibm,cvc/memory-region to point to /reserved-memory/secure-crypt-algo-code instead of /ibm,hostboot/reserved-memory/secure-crypt-algo-code.

- libstb: add support for ibm,secureboot-v2

  ibm,secureboot-v2 changes:

  - The Container Verification Code is represented by the ibm,cvc node.
  - Each ibm,cvc child describes a CVC service.
  - hash-algo is superseded by hw-key-hash-size.

- hdata/tpmrel.c: add ibm, cvc device tree node

  In P9, the Container Verification Code is stored in a hostboot reserved memory and the list of provided CVC services is stored in the TPMREL_IDATA_HASH_VERIF_OFFSETS idata array. Each CVC service has an offset and version.

  This adds the ibm,cvc device tree node and its documentation.

- hdata/tpmrel.c: add firmware event log info to the tpm node

  This parses the firmware event log information from the secureboot_tpm_info HDAT structure and add it to the tpm device tree node.

  There can be multiple secureboot_tpm_info entries with each entry corresponding to a master processor that has a tpm device, however, multiple tpm is not supported.

- hdata/spira: add ibm,secureboot node in P9

  In P9, skiboot builds the device tree from the HDAT. These are the "ibm,secureboot" node changes compared to P8:

  - The Container-Verification-Code (CVC), a.k.a. ROM code, is no longer stored in a secure ROM with static address. In P9, it is stored in a hostboot reserved memory and each service provided also has a version, not only an offset.
  - The hash-algo property is not provided via HDAT, instead it provides the hw-key-hash-size, which is indeed the information required by the CVC to verify containers.

  This parses the iplparams_sysparams HDAT structure and creates the "ibm,secureboot", which is bumped to "ibm,secureboot-v2".

  In "ibm,secureboot-v2":

  - hash-algo property is superseded by hw-key-hash-size.
  - container verification code is explicitly described by a child node. Added in a subsequent patch.

  See *ibm,secureboot* for documentation.

- libstb/tpm_chip.c: define pr_fmt and fix messages logged

  This defines pr_fmt and also fix messages logged:

  - EV_SEPARATOR instead of 0xFFFFFFFF
  - when an event is measured it also prints the tpm id, event type and event log length

  Now we can filter the messages logged by libstb and its sub-modules by running:

  ```
  grep STB /sys/firmware/opal/msglog
  ```

- libstb/tss: update the list of event types supported

Skiboot, precisely the tpmLogMgr, initializes the firmware event log by calculating its length so that a new event can be recorded without exceeding the log size. In order to calculate the size, it walks through the log until it finds a specific event type. However, if the log has an unknown event type, the tpmLogMgr will not be able to reach the end of the log.

This updates the list of event types with all of those supported by hostboot. Thus, skiboot can properly calculate the event log length.

- tpm_i2c_nuvoton: add nuvoton, npct601 to the compatible property

  The linux kernel doesn't have a driver compatible with "nuvoton,npct650", but it does have for "nuvoton,npct601", which should also be compatible with npct650.

  This adds "nuvoton,npct601" to the compatible devtree property.

- libstb/trustedboot.c: import stb_final() from stb.c

  The stb_final() primary goal is to measure the event EV_SEPARATOR into PCR[0-7] when trusted boot is about to exit the boot services.

  This imports the stb_final() from stb.c into trustedboot.c, but making the following changes:

  - Rename it to trustedboot_exit_boot_services().
  - As specified in the TCG PC Client specification, EV_SEPARATOR events must be logged with the name 0xFFFFFF.
  - Remove the ROM driver clean-up call.
  - Don't allow code to be measured in skiboot after trustedboot_exit_boot_services() is called.

- libstb/cvc.c: import softrom behaviour from drivers/sw_driver.c

  Softrom is used only for testing with mambo. By setting compatible="ibm,secureboot-v1-softrom" in the "ibm,secureboot" node, firmware images can be properly measured even if the Container-Verification-Code (CVC) is not available. In this case, the mbedtls_sha512() function is used to calculate the sha512 hash of the firmware images.

  This imports the softrom behaviour from libstb/drivers/sw_driver.c code into cvc.c, but now softrom is implemented as a flag. When the flag is set, the wrappers for the CVC services work the same way as in sw_driver.c.

- libstb/trustedboot.c: import tb_measure() from stb.c

  This imports tb_measure() from stb.c, but now it calls the CVC sha512 wrapper to calculate the sha512 hash of the firmware image provided.

  In trustedboot.c, the tb_measure() is renamed to trustedboot_measure().

  The new function, trustedboot_measure(), no longer checks if the container payload hash calculated at boot time matches with the hash found in the container header. A few reasons:

  - If the system admin wants the container header to be checked/validated, the secure boot jumper must be set. Otherwise, the container header information may not be reliable.
  - The container layout is expected to change over time. Skiboot would need to maintain a parser for each container layout change.
  - Skiboot could be checking the hash against a container version that is not supported by the Container-Verification-Code (CVC).

    The tb_measure() calls are updated to trustedboot_measure() in a subsequent patch.

- libstb/secureboot.c: import sb_verify() from stb.c

This imports the sb_verify() function from stb.c, but now it calls the CVC verify wrapper in order to verify signed firmware images. The hw-key-hash and hw-key-hash-size initialized in secureboot.c are passed to the CVC verify function wrapper.

In secureboot.c, the sb_verify() is renamed to secureboot_verify(). The sb_verify() calls are updated in a subsequent patch.

XIVE

- xive: Don't bother cleaning up disabled EQs in reset

  Additionally, warn if we find an enabled one that isn't one of the firmware built-in queues.

- xive: Warn on valid VPs found in abnormal cases

  If an allocated VP is left valid at xive_reset() or Linux tries to free a valid (enabled) VP block, print errors. The former happens occasionally if kdump'ing while KVM is running so keep it as a debug message. The latter is a programming error in Linux so use a an error log level.

- xive: Properly reserve built-in VPs in non-group mode

  This is not normally used but if the #define is changed to disable block group mode we would incorrectly clear the buddy completely without marking the built-in VPs reserved.

- xive: Quieten debug messages in standard builds

  This makes a bunch of messages, especially the per-CPU ones, only enabled in debug builds. This avoids clogging up the OPAL logs with XIVE related messages that have proven not being particularly useful for field defects.

- xive: Implement "single escalation" feature

  This adds a new VP flag to control the new DD2.0 "single escalation" feature.

  This feature allows us to have a single escalation interrupt per VP instead of one per queue.

  It works by hijacking queue 7 (which is this no longer usable when that is enabled) and exploiting two new hardware bits that will:

  - Make the normal queues (0..6) escalate unconditionally thus ignoring the ESe bits.
  - Route the above escalations to queue 7
  - Have queue 7 silently escalate without notification

  Thus the escalation of queue 7 becomes the one escalation interrupt for all the other queues.

- xive: When disabling a VP, wipe all of its settings
- xive: Improve cleaning up of EQs

  Factors out the function that sets an EQ back to a clean state and add a cleaning pass for queue left enabled when freeing a block of VPs.

- xive: When disabling an EQ, wipe all of its settings

  This avoids having configuration bits left over

- xive: Define API for single-escalation VP mode

  This mode allows all queues of a VP to use the same escalation interrupt, at the cost of losing priority 7.

  This adds the definition and documentation of the API, the implementation will come next.

- xive: Fix ability to clear some EQ flags

  We could never clear "unconditional notify" and "escalate"

- xive: Update inits for DD2.0

  This updates some inits based on information from the HW designers. This includes enabling some new DD2.0 features that we don't yet exploit.

- xive: Ensure VC informational FIRs are masked

  Some HostBoot versions leave those as checkstop, they are harmless and can sometimes occur during normal operations.

- xive: Fix occasional VC checkstops in xive_reset

  The current workaround for the scrub bug described in __xive_cache_scrub() has an issue in that it can leave dirty invalid entries in the cache.

  When cleaning up EQs or VPs during reset, if we then remove the underlying indirect page for these entries, the XIVE will checkstop when trying to flush them out of the cache.

  This replaces the existing workaround with a new pair of workarounds for VPs and EQs:

  – The VP one does the dummy watch on another entry than the one we scrubbed (which does the job of pushing old stores out) using an entry that is known to be backed by a permanent indirect page.
  – The EQ one switches to a more efficient workaround which consists of doing a non-side-effect ESB load from the EQ's ESe control bits.

- xive: Do not return a trigger page for an escalation interrupt

  This is bogus, we don't support them. (Thankfully the callers didn't actually try to use this on escalation interrupts).

- xive: Mark a freed IRQs IVE as valid and masked

  Removing the valid bit means a FIR will trip if it's accessed inadvertently. Under some circumstances, the XIVE will speculatively access an IVE for a masked interrupt and trip it. So make sure that freed entries are still marked valid (but masked).

PCI

Since skiboot-5.10-rc3:

- phb3/phb4/p7ioc: Document supported TCE sizes in DT

  Add a new property, "ibm,supported-tce-sizes", to advertise to Linux how big the available TCE sizes are. Each value is a bit shift, from smallest to largest.

- phb4: Fix TCE page size

  The page sizes for TCEs on P9 were inaccurate and just copied from PHB3, so correct them.

- Revert "pci: Shared slot state synchronisation for hot reset"

  An issue was found in shared slot reset where the system can be stuck in an infinite loop, pull the code out until there's a proper fix.

  This reverts commit 1172a6c57ff3c66f6361e572a1790cbcc0e5ff37.

- hdata/iohub: Use only wildcard slots for pluggables

  We don't want to cause a VID:DID check against pluggable devices, as they may use multiple devids.

Narrow the condition under which VID:DID is listed in the dt, so that we'll end up creating a wildcard slot for these instead.

Since skiboot-5.9:

- pci: Shared slot state synchronisation for hot reset

  When a device is shared between two PHBs, it doesn't get reset properly unless both PHBs issue a hot reset at "the same time". Practically this means a hot reset needs to be issued on both sides, and neither should bring the link up until the reset on both has completed.

- pci: Track peers of slots

  Witherspoon introduced a new concept where one physical slot is shared between two PHBs. Making a slot aware of its peer enables syncing between them where necessary.

PHB4

Since skiboot-5.10-rc4:

- phb4: Disable lane eq when retrying some nvidia GEN3 devices

  This fixes these nvidia cards training at only GEN2 spends rather than GEN3 by disabling PCIe lane equalisation.

  Firstly we check if the card is in a whitelist. If it is and the link has not trained optimally, retry with lane equalisation off. We do this on all POWER9 chip revisions since this is a device issue, not a POWER9 chip issue.

Since skiboot-5.10-rc2:

- phb4: Only escalate freezes on MMIO load where necessary

  In order to work around a hardware issue, MMIO load freezes were escalated to fences on every chip. Now that hardware no longer requires this, restrict escalation to the chips that actually need it.

Since skiboot-5.9:

- phb4: Change PCI MMIO timers

  Currently we have a mismatch between the NCU and PCI timers for MMIO accesses. The PCI timers must be lower than the NCU timers otherwise it may cause checkstops.

  This changes PCI timeouts controlled by skiboot to 33-50ms. It should be forwards and backwards compatible with expected hostboot changes to the NCU timer.

- phb4: Change default GEN3 lane equalisation setting to 0x54

  Currently our GEN3 lane equalisation settings are set to 0x77. Change this to 0x54. This change will allow us to train at GEN3 in a shorter time and more consistently.

  This setting gives us a TX preset 0x4 and RX hint 0x5. This gives a boost in gain for high frequency signalling. It allows the most optimal continuous time linear equalizers (CTLE) for the remote receiver port and de-emphasis and pre-shoot for the remote transmitter port.

  Machine Readable Workbooks (MRW) are moving to this new value also.

- phb4: Init changes

  These init changes for phb4 from the HW team.

  Link down are now endpoint recoverable (ERC) rather than PHB fatal errors.

  BLIF Completion Timeout Error now generate an interrupt rather than causing freeze events.

- phb4: Fix lane equalisation setting

  Fix cut and paste from phb3. The sizes have changes now we have GEN4, so the check here needs to change also

  Without this we end up with the default settings (all '7') rather than what's in HDAT.

- hdata: Fix copying GEN4 lane equalisation settings

  These aren't copied currently but should be.

- phb4: Fix PE mapping of M32 BAR

  The M32 BAR is the PHB4 region used to map all the non-prefetchable or 32-bit device BARs. It's supposed to have its segments remapped via the MDT and Linux relies on that to assign them individual PE#.

  However, we weren't configuring that properly and instead used the mode where PE# == segment#, thus causing EEH to freeze the wrong device or PE#.

- phb4: Fix lost bit in PE number on config accesses

  A PE number can be up to 9 bits, using a uint8_t won't fly..

  That was causing error on config accesses to freeze the wrong PE.

- phb4: Update inits

  New init value from HW folks for the fence enable register.

  This clears bit 17 (CFG Write Error CA or UR response) and bit 22 (MMIO Write DAT_ERR Indication) and sets bit 21 (MMIO CFG Pending Error)

CAPI

Since skiboot-5.10-rc2:

- capi: Enable channel tag streaming for PHB in CAPP mode

  We re-enable channel tag streaming for PHB in CAPP mode as without it PEC was waiting for cresp for each DMA write command before sending a new DMA write command on the Powerbus. This resulted in much lower DMA write performance than expected.

  The patch updates enable_capi_mode() to remove the masking of channel_streaming_en bit in PBCQ Hardware Configuration Register. Also does some re-factoring of the code that updates this register to use xscom_write_mask instead of xscom_read followed by a xscom_write.

Since skiboot-5.10-rc1:

- capi: Fix the max tlbi divider and the directory size.

  Switch to 512KB mode (directory size) as we don't use bit 48 of the tag in addressing the array. This mode is controlled by the Snoop CAPI Configuration Register. Set the maximum of the number of data polls received before signaling TLBI hang detect timer expired. The value of '0000' is equal to 16.

Since skiboot-5.9:

- capi: Disable CAPP virtual machines

  When exercising more than one CAPI accelerators simultaneously in cache coherency mode, the verification team is seeing a deadlock. To fix this a workaround of disabling CAPP virtual machines is suggested. These 'virtual machines' let PSL queue multiple CAPP commands for servicing by CAPP there by increasing throughput. Below is the error scenario described by the h/w team:

  " With virtual machines enabled we had a deadlock scenario where with 2 or more CAPI's in a system you could get in a deadlock scenario due to cast-outs that are required break the deadlock (evict lines that another CAPI is requesting) get stuck in the virtual machine queue by a command ahead of it that is being retried by the same scenario in the other CAPI. "

- capi: Perform capp recovery sequence only when PBCQ is idle

  Presently during a CRESET the CAPP recovery sequence can be executed multiple times in case PBCQ on the PEC is still busy processing in/out bound in-flight transactions.

- xive: Mask MMIO load/store to bad location FIR

  For opencapi, the trigger page of an interrupt is mapped to user space. The intent is to write the page to raise an interrupt but there's nothing to prevent a user process from reading it, which has the unfortunate consequence of checkstopping the system.

  Mask the FIR bit raised when an MMIO operation targets an invalid location. It's the recommendation from recent documentation and hostboot is expected to mask it at some point. In the meantime, let's play it safe.

- phb4: Dump CAPP error registers when it asserts link down

  This patch introduces a new function phb4_dump_app_err_regs() that dumps CAPP error registers in case the PEC nestfir register indicates that the fence was due to a CAPP error (BIT-24).

  Contents of these registers are helpful in diagnosing CAPP issues. Registers that are dumped in phb4_dump_app_err_regs() are:

  - CAPP FIR Register
  - CAPP APC Master Error Report Register
  - CAPP Snoop Error Report Register
  - CAPP Transport Error Report Register
  - CAPP TLBI Error Report Register
  - CAPP Error Status and Control Register

- capi: move the acknowledge of the HMI interrupt

  We need to acknowledge an eventual HMI initiated by the previous forced fence on the PHB to work around a non-existent PE in the phb4_creset() function. For this reason do_capp_recovery_scoms() is called now at the beginning of the step: PHB4_SLOT_CRESET_WAIT_CQ

- capi: update ci store buffers and dma engines

  The number of read (APC type traffic) and mmio store (MSG type traffic) resources assigned to the CAPP is controlled by the CAPP control register.

  According to the type of CAPI cards present on the server, we have to configure differently the CAPP messages and the DMA read engines given to the CAPP for use.

HMI

- core/hmi: Display chip location code while displaying core FIR.
- core/hmi: Do not display FIR details if none of the bits are set.

  So that we don't flood OPAL console logs with information that is not useful.

- opal/hmi: HMI logging with location code info.

  Add few HMI debug prints with location code info few additional info.

  No functionality change.

  With this patch the log messages will look like:

```
[210612.175196744,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
[210612.175200449,7] HMI: [Loc: UOPWR.1302LFA-Node0-Proc1]: P:8 C:16 T:1:␣
↪TFMR(2d12000870e04020) Timer Facility Error

[210660.259689526,7] HMI: Received HMI interrupt: HMER = 0x2040000000000000
[210660.259695649,7] HMI: [Loc: UOPWR.1302LFA-Node0-Proc0]: P:0 C:16 T:1:␣
↪Processor recovery Done.
```

- core/hmi: Use pr_fmt macro for tagging log messages

  No functionality changes.

- opal: Get chip location code

  and store it under proc_chip for quick reference during HMI handling code.

Sensors

- occ-sensors: Fix up quad/gpu location mix-up

  The GPU and QUAD sensor location types are swapped compared to what exists in the OCC code base which is authoritative. Fix them up.

- sensors: occ: Skip counter type of sensors

  Don't add counter type of sensors to device-tree as they don't fit into hwmon sensor interface.

- sensors: dts: Assert special wakeup on idle cores while reading temperature

  In P9, when a core enters a stop state, its clocks will be stopped to save power and hence we will not be able to perform a SCOM operation to read the DTS temperature sensor. Hence, assert a special wakeup on cores that have entered a stop state in order to successfully complete the SCOM operation.

- sensors: occ: Skip power sensors with zero sample value

  APSS is not available on platforms like Zaius, Romulus where OCC can only measure Vdd (core) and Vdn (nest) power from the AVSbus reading. So all the sensors for APSS channels will be populated with 0. Different component power sensors like system, memory which point to the APSS channels will also be 0.

  As per OCC team (Martha Broyles) zeroed power sensor means that the system doesn't have it. So this patch filters out these sensors.

- sensors: occ: Skip GPU sensors for non-gpu systems

- sensors: Fix dtc warning for new occ in-band sensors.

  dtc complains about missing reg property when a DT node is having a unit name or address but no reg property.

```
/ibm,opal/sensors/vrm-in@c00004 has a unit name, but no reg property
/ibm,opal/sensors/gpu-in@c0001f has a unit name, but no reg property
/ibm,opal/sensor-groups/occ-js@1c00040 has a unit name, but no reg property
```

  This patch fixes these warnings for new occ in-band sensors and also for sensor-groups by adding necessary properties.

- sensors: Fix dtc warning for dts sensors.

  dtc complains about missing reg property when a DT node is having a unit name or address but no reg property.

  Example warning for core dts sensor:

```
/ibm,opal/sensors/core-temp@5c has a unit name, but no reg property
/ibm,opal/sensors/core-temp@804 has a unit name, but no reg property
```

This patch fixes this by adding necessary properties.

- hw/occ: Fix psr cpu-to-gpu sensors node dtc warning.

  dtc complains about missing reg property when a DT node is having a unit name or address but no reg property.

  ```
  /ibm,opal/power-mgt/psr/cpu-to-gpu@0 has a unit name, but no reg property
  /ibm,opal/power-mgt/psr/cpu-to-gpu@100 has a unit name, but no reg property
  ```

  This patch fixes this by adding necessary properties.

General fixes

Since skiboot-5.10-rc3:

- core: Fix mismatched names between reserved memory nodes & properties

  OPAL exposes reserved memory regions through the device tree in both new (nodes) and old (properties) formats.

  However, the names used for these don't match - we use a generated cell address for the nodes, but the plain region name for the properties.

  This fixes a warning from FWTS

Since skiboot-5.10-rc2:

- vas: Disable VAS/NX-842 on some P9 revisions

  VAS/NX-842 are not functional on some P9 revisions, so disable them in hardware and skip creating their device tree nodes.

  Since the intent is to prevent OS from configuring VAS/NX, we remove only the platform device nodes but leave the VAS/NX DT nodes under xscom (i.e we don't skip add_vas_node() in hdata/spira.c)

- core/device.c: Fix dt_find_compatible_node

  dt_find_compatible_node() and dt_find_compatible_node_on_chip() are used to find device nodes under a parent/root node with a given compatible property.

  dt_next(root, prev) is used to walk the child nodes of the given parent and takes two arguments - root contains the parent node to walk whilst prev contains the previous child to search from so that it can be used as an iterator over all children nodes.

  The first iteration of dt_find_compatible_node(root, prev) calls dt_next(root, root) which is not a well defined operation as prev is assumed to be child of the root node. The result is that when a node contains no children it will start returning the parent nodes siblings until it hits the top of the tree at which point a NULL derefence is attempted when looking for the root nodes parent.

  Dereferencing NULL can result in undesirable data exceptions during system boot and untimely non-hilarious system crashes. dt_next() should not be called with prev == root. Instead we add a check to dt_next() such that passing prev = NULL will cause it to start iterating from the first child node (if any).

  This manifested itself in a crash on boot on ZZ systems.

- hw/occ: Fix fast-reboot crash in P8 platforms.

commit 85a1de35cbe4 ("fast-boot: occ: Re-parse the pstate table during fast-boot") breaks the fast-reboot on P8 platforms while reiniting the OCC pstates. On P8 platforms OPAL adds additional two properties #address-cells and #size-cells under ibm,opal/power-mgmt/ DT node. While in fast-reboot same properties adding back to the same node results in Duplicate properties and hence fast-reboot fails with below traces.

```
[  541.410373292,5] OCC: All Chip Rdy after 0 ms
[  541.410488745,3] Duplicate property "#address-cells" in node /ibm,opal/power-
↪mgt
[  541.410694290,0] Aborting!
CPU 0058 Backtrace:
 S: 0000000031d639d0 R: 000000003001367c   .backtrace+0x48
 S: 0000000031d63a60 R: 000000003001a03c   ._abort+0x4c
 S: 0000000031d63ae0 R: 00000000300267d8   .new_property+0xd8
 S: 0000000031d63b70 R: 0000000030026a28   .__dt_add_property_cells+0x30
 S: 0000000031d63c10 R: 000000003003ea3c   .occ_pstates_init+0x984
 S: 0000000031d63d90 R: 00000000300142d8   .load_and_boot_kernel+0x86c
 S: 0000000031d63e70 R: 000000003002586c   .fast_reboot_entry+0x358
 S: 0000000031d63f00 R: 00000000300029f4   fast_reset_entry+0x2c
```

This patch fixes this issue by removing these two properties on P8 while doing OCC pstates re-init in fast-reboot code path.

Since skiboot-5.10-rc1:

- fast-reboot: occ: Re-parse the pstate table during fast-reboot

  OCC shares the frequency list to host by copying the pstate table to main memory in HOMER. This table is parsed during boot to create device-tree properties for frequency and pstate IDs. OCC can update the pstate table to present a new set of frequencies to the host. But host will remain oblivious to these changes unless it is re-inited with the updated device-tree CPU frequency properties. So this patch allows to re-parse the pstate table and update the device-tree properties during fast-reboot.

  OCC updates the pstate table when asked to do so using pstate-table bias command. And this is mainly used by WOF team for characterization purposes.

- fast-reboot: move pci_reset error handling into fast-reboot code

  pci_reset() currently does a platform reboot if it fails. It should not know about fast-reboot at this level, so instead have it return an error, and the fast reboot caller will do the platform reboot.

  The code essentially does the same thing, but flexibility is improved. Ideally the fast reboot code should perform pci_reset and all such fail-able operations before the CPU resets itself and destroys its own stack. That's not the case now, but that should be the goal.

Since skiboot-5.9:

- lpc: Clear pending IRQs at boot

  When we come in from hostboot the LPC master has the bus reset indicator set. This error isn't handled until the host kernel unmasks interrupts, at which point we get the following spurious error:

  ```
  [   20.053560375,3] LPC: Got LPC reset on chip 0x0 !
  [   20.053564560,3] LPC[000]: Unknown LPC error Error address reg: 0x00000000
  ```

  Fix this by clearing the various error bits in the LPC status register before we initialise the skiboot LPC bus driver.

- hw/imc: Check ucode state before exposing units to Linux

  disable_unavailable_units() checks whether the ucode is in the running state before enabling the nest units in the device tree. From a recent debug, it is found that on some system boot, ucode is not loaded and running in all the chips in the system. And this caused a fail in OPAL_IMC_COUNTERS_STOP call where we check for ucode state on each chip. Bug here is that disable_unavailable_units() checks the state of the ucode only in boot cpu chip. Patch adds a condition in disable_unavailable_units() to check for the ucode state in all the chip before enabling the nest units in the device tree node.

- hdata/vpd: Add vendor property

  ibm,vpd blob contains VN field. Use that to populate vendor property for various FRU's.

- hdata/vpd: Fix DTC warnings

  All the nodes under the vpd hierarchy have a unit address (their SLCA index) but no reg properties. Add them and their size/address cells to squash the warnings.

- HDAT/i2c: Fix SPD EEPROM compatible string

  Hostboot doesn't give us accurate information about the DIMM SPD devices. Hack around by assuming any EEPROM we find on the SPD I2C master is an SPD EEPROM.

- hdata/i2c: Fix 512Kb EEPROM size

  There's no such thing as a 412Kb EEPROM.

- libflash/mbox-flash: fall back to requesting lower MBOX versions from BMC

  Some BMC mbox implementations seem to sometimes mysteriously fail when trying to negotiate v3 when they only support v2. To work around this, we can fall back to requesting lower mbox protocol versions until we find one that works.

  In theory, this should already "just work", but we have a counter example, which this patch fixes.

- IPMI: Fix platform.cec_reboot() null ptr checks

  Kudos to Hugo Landau who reported this in: https://github.com/open-power/skiboot/issues/142

- hdata: Add location code property to xscom node

  This patch adds chip location code property to xscom node.

- p8-i2c: Limit number of retry attempts

  Current we will attempt to start an I2C transaction until it succeeds. In the event that the OCC does not release the lock on an I2C bus this results in an async token being held forever and the kernel thread that started the transaction will block forever while waiting for an async completion message. Fix this by limiting the number of attempts to start the transaction.

- p8-i2c: Don't write the watermark register at init

  On P9 the I2C master is shared with the OCC. Currently the watermark values are set once at init time which is bad for two reasons:

  a) We don't take the OCC master lock before setting it. Which may cause issues if the OCC is currently using the master.

  b) The OCC might change the watermark levels and we need to reset them.

  Change this so that we set the watermark value when a new transaction is started rather than at init time.

- hdata: Rename 'fsp-ipl-side' as 'sp-ipl-side'

  as OPAL is building device tree for both FSP and BMC system. Also I don't see anyone using this property today. Hence renaming should be fine.

- hdata/vpd: add support for parsing CPU VRML records

  Allows skiboot to parse out the processor part/serial numbers on OpenPOWER P9 machines.

- core/lock: Introduce atomic cmpxchg and implement try_lock with it

  cmpxchg will be used in a subsequent change, and this reduces the amount of asm code.

- direct-controls: add xscom error handling for p8

  Add xscom checks which will print something useful and return error back to callers (which already have error handling plumbed in).

- direct-controls: p8 implementation of generic direct controls

  This reworks the sreset functionality that was brought over from fast-reboot, and fits it under the generic direct controls APIs.

  The fast reboot APIs are implemented using generic direct controls, which also makes them available on p9.

- fast-reboot: allow mambo fast reboot independent of CPU type

  Don't tie mambo fast reboot to POWER8 CPU type.

- fast-reboot: remove delay after sreset

  There is a 100ms delay when targets reach sreset which does not appear to have a good purpose. Remove it and therefore reduce the sreset timeout by the same amount.

- fast-reboot: add more barriers around cpu state changes

  This is a bit of paranoia, but when a CPU changes state to signal it has reached a particular point, all previous stores should be visible.

- fast-reboot: add sreset timeout detection and handling

  Have the initiator wait for all its sreset targets to call in, and time out after 200ms if they did not. Fail and revert to IPL reboot.

  Testing indicates that after successful sreset_all_others(), it takes less than 102ms (in hundreds of fast reboots) for secondaries to call in. 100 of that is due to an initial delay, but core un-splitting was not measured.

- fast-reboot: make spin loops consistent and SMT friendly
- fast-reboot: add sreset_all_others error handling

  Pass back failures from sreset_all_others, also change return codes to OPAL form in sreset_all_prepare to match.

  Errors will revert to the IPL path, so it's not critical to completely clean up everything if that would complicate things. Detecting the error and failing is the important thing.

- fast-reboot: restore SMT priority on spin loop exit
- Add documentation for ibm, firmware-versions device tree node
- NX: Print read xscom config failures.

  Currently in NX, only write xscom config failures are tracing. Add trace statements for read xscom config failures too. No functional changes.

- hw/nx: Fix NX BAR assignments

  The NX rng BAR is used by each core to source random numbers for the DARN instruction. Currently we configure each core to use the NX rng of the chip that it exists on. Unfortunately, the NX can be de-configured by hostboot and in this case we need to use the NX of a different chip.

  This patch moves the BAR assignments for the NX into the normal nx-rng init path. This lets us check if the normal (chip local) NX is active when configuring which NX a core should use so that we can fall back gracefully.

- FSP-elog: Reduce verbosity of elog messages

  These messages just fill up the opal console log with useless messages resulting in us losing useful information.

  They have been like this since the first commit in skiboot. Make them trace.

- core/bitmap: fix bitmap iteration limit corruption

  The bitmap iterators did not reduce the number of bits to scan when searching for the next bit, which would result in them overrunning their bitmap.

  These are only used in one place, in xive reset, and the effect is that the xive reset code will keep zeroing memory until it reaches a block of memory of MAX_EQ_COUNT >> 3 bits in length, all zeroes.

- hw/imc: always enable "imc_nest_chip" exports property

  imc_dt_update_nest_node() adds a "imc_nest_chip" property to the "exports" node (under opal_node) to view nest counter region. This comes handy when debugging ucode runtime errors (like counter data update or control block update so on…). And current code enables the property only if the microcode is in running state at system boot. To aid the debug of ucode not running/starting issues at boot, enable the addition of "imc_nest_chip" property always.

NVLINK2

Since skiboot-5.10-rc2:

- npu2: Disable TVT range check when in bypass mode

  On POWER9 the GPUs need to be able to access the MMIO memory space. Therefore the TVT range check needs to include the MMIO address space. As any possible range check would cover all of memory anyway this patch just disables the TVT range check all together when bypassing the TCE tables.

- hw/npu2: support creset of npu2 devices

  creset calls in the hw procedure that resets the PHY, we don't take them out of reset, just put them in reset.

  this fixes a kexec issue.

Since skiboot-5.10-rc1:

- npu2/tce: Fix page size checking

  The page size is encoded in the TVT data [59:63] as @shift+11 but the tce_kill handler does not do the math right; this fixes it.

Since skiboot-5.9:

- npu2-hw-procedures.c: Correct phy lane mapping

  Each NVLINK2 device is associated with a particular group of OBUS lanes via a lane mask which is read from HDAT via the device-tree. However Skiboot's interpretation of lane mask was different to what is exported from the HDAT.

  Specifically the lane mask bits in the HDAT are encoded in IBM bit ordering for a 24-bit wide value. So for example in normal bit ordering lane-0 is represented by having lane-mask bit 23 set and lane-23 is represented by lane-mask bit 0. This patch alters the Skiboot interpretation to match what is passed from HDAT.

- npu2-hw-procedures.c: Power up lanes during ntl reset

  Newer versions of Hostboot will not power up the NVLINK2 PHY lanes by default. The phy_reset procedure already powers up the lanes but they also need to be powered up in order to access the DL.

  The reset_ntl procedure is called by the device driver to bring the DL out of reset and get it into a working state. Therefore we also need to add lane and clock power up to the reset_ntl procedure.

- npu2.c: Add PE error detection

  Invalid accesses from the GPU can cause a specific PE to be frozen by the NPU. Add an interrupt handler which reports the frozen PE to the operating system via as an EEH event.

- npu2.c: Fix XIVE IRQ alignment
- npu2: hw-procedures: Refactor reset_ntl procedure

  Change the implementation of reset_ntl to match the latest programming guide documentation.

- npu2: hw-procedures: Add phy_rx_clock_sel()

  Change the RX clk mux control to be done by software instead of HW. This avoids glitches caused by changing the mux setting.

- npu2: hw-procedures: Change phy_rx_clock_sel values

  The clock selection bits we set here are inputs to a state machine.

  DL clock select (bits 30-31)

  0b00 lane 0 clock

  0b01 lane 7 clock

  0b10 grid clock

  0b11 invalid/no-op

  To recover from a potential glitch, we need to ensure that the value we set forces a state change. Our current sequence is to set 0x3 followed by 0x1. With the above now known, that is actually a no-op followed by selection of lane 7. Depending on lane reversal, that selection is not a state change for some bricks.

  The way to force a state change in all cases is to switch to the grid clock, and then back to a lane.

- npu2: hw-procedures: Manipulate IOVALID during training

  Ensure that the IOVALID bit for this brick is raised at the start of link training, in the reset_ntl procedure.

  Then, to protect us from a glitch when the PHY clock turns off or gets chopped, lower IOVALID for the duration of the phy_reset and phy_rx_dccal procedures.

- npu2: hw-procedures: Add check_credits procedure

  As an immediate mitigation for a current hardware glitch, add a procedure that can be used to validate NTL credit values. This will be called as a safeguard to check that link training succeeded.

  Assert that things are exactly as we expect, because if they aren't, the system will experience a catastrophic failure shortly after the start of link traffic.

- npu2: Print bdfn in NPU2DEV* logging macros

  Revise the NPU2DEV{DBG,INF,ERR} logging macros to include the device's bdfn. It's useful to know exactly which link we're referring to.

  For instance, instead of

  ```
  [  234.044921238,6] NPU6: Starting procedure reset_ntl
  [  234.048578101,6] NPU6: Starting procedure reset_ntl
  [  234.051049676,6] NPU6: Starting procedure reset_ntl
  [  234.053503542,6] NPU6: Starting procedure reset_ntl
  [  234.057182864,6] NPU6: Starting procedure reset_ntl
  [  234.059666137,6] NPU6: Starting procedure reset_ntl
  ```

  we'll get

```
[  234.044921238,6] NPU6:0:0.0 Starting procedure reset_ntl
[  234.048578101,6] NPU6:0:0.1 Starting procedure reset_ntl
[  234.051049676,6] NPU6:0:0.2 Starting procedure reset_ntl
[  234.053503542,6] NPU6:0:1.0 Starting procedure reset_ntl
[  234.057182864,6] NPU6:0:1.1 Starting procedure reset_ntl
[  234.059666137,6] NPU6:0:1.2 Starting procedure reset_ntl
```

- npu2: Move to new GPU memory map

  There are three different ways we configure the MCD and memory map.

  1) **Old way (current way)** Skiboot configures the MCD and puts GPUs at 4TB and below
  2) **New way with MCD** Hostboot configures the MCD and skiboot puts GPU at 4TB and above
  3) **New way without MCD** No one configures the MCD and skiboot puts GPU at 4TB and below

  The patch keeps option 1 and adds options 2 and 3.

  The different configurations are detected using certain scoms (see patch).

  Option 1 will go away eventually as it's a configuration that can cause xstops or data integrity problems. We are keeping it around to support existing hostboot.

  Option 2 supports only 4 GPUs and 512GB of memory per socket.

  Option 3 supports 6 GPUs and 4TB of memory but may have some performance impact.

- phys-map: Rename GPU_MEM to GPU_MEM_4T_DOWN

  This map is soon to be replaced, but we are going to keep it around for a little while so that we support older hostboot firmware.

Platform Specific Fixes

Witherspoon

- Witherspoon: Remove old Witherspoon platform definition

  An old Witherspoon platform definition was added to aid the transition from versions of Hostboot which didn't have the correct NVLINK2 HDAT information available and/or planar VPD. These system should now be updated so remove the possibly incorrect default assumption.

  This may disable NVLINK2 on old out-dated systems but it can easily be restored with the appropriate FW and/or VPD updates. In any case there is a a 50% chance the existing default behaviour was incorrect as it only supports 6 GPU systems. Using an incorrect platform definition leads to undefined behaviour which is more difficult to detect/debug than not creating the NVLINK2 devices so remove the possibly incorrect default behaviour.

- Witherspoon: Fix VPD EEPROM type

  There are user-space tools that update the planar VPD via the sysfs interface. Currently we do not get correct information from hostboot about the exact type of the EEPROM so we need to manually fix it up here. This needs to be done as a platform specific fix since there is not standardised VPD EEPROM type.

IBM FSP Systems

- nvram: Fix 'missing' nvram on FSP systems.

commit ba4d46fdd9eb ("console: Set log level from nvram") wants to read from NVRAM rather early. This works fine on BMC based systems as nvram_init() is actually synchronous. This is not true for FSP systems and it turns out that the query for the console log level simply queries blank nvram.

The simple fix is to wait for the NVRAM read to complete before performing any query. Unfortunately it turns out that the fsp-nvram code does not inform the generic NVRAM layer when the read is complete, rather, it must be prompted to do so.

This patch addresses both these problems. This patch adds a check before the first read of the NVRAM (for the console log level) that the read has completed. The fsp-nvram code has been updated to inform the generic layer as soon as the read completes.

The old prompt to the fsp-nvram code has been removed but a check to ensure that the NVRAM has been loaded remains. It is conservative but if the NVRAM is not done loading before the host is booted it will not have an nvram device-tree node which means it won't be able to access the NVRAM at all, ever, even after the NVRAM has loaded.

Utilities

Since skiboot-5.10-rc1:

- opal-prd: Fix FTBFS with -Werror=format-overflow

  i2c.c fails to compile with gcc7 and -Werror=format-overflow used in Debian Unstable and Ubuntu 18.04 :

  ```
  i2c.c: In function 'i2c_init':
  i2c.c:211:15: error: '%s' directive writing up to 255 bytes into a
  region of size 236 [-Werror=format-overflow=]
  ```

Since skiboot-5.9:

- Fix xscom-utils distclean target

  In Debian/Ubuntu, the packaging system likes to have a full clean-up that restores the tree back to original one, so add some files to the distclean target.

- Add man pages for xscom-utils and pflash

  For the need of Debian/Ubuntu packaging, I inferred some initial man pages from their help output.

gard

- gard: Add tests

  I hear Stewart likes these for some reason. Dunno why.

- gard: Add OpenBMC vPNOR support

  A big-ol-hack to add some checking for OpenBMC's vPNOR GUARD files under /media/pnor-prsv. This isn't ideal since it doesn't handle the create case well, but it's better than nothing.

- gard: Always use MTD to access flash

  Direct mode is generally either unsafe or unsupported. We should always access the PNOR via an MTD device so make that the default. If someone really needs direct mode, then they can use pflash.

- gard: Fix up do_create return values

  The return value of a subcommand is interpreted as a libflash error code when it's positive or some subcommand specific error when negative. Currently the create subcommand always returns zero when exiting (even for errors) so fix that.

- gard: Add usage message for -p

  The -p argument only really makes sense when -f is specified. Print an actual error message rather than just the usage blob.

- gard: Fix max instance count

  There's an entire byte for the instance count rather than a nibble. Only barf if the instance number is beyond 255 rather than 16.

- gard: Fix up path parsing

  Currently we assume that the Unit ID can be used as an array index into the chip_units[] structure. There are holes in the ID space though, so this doesn't actually work. Fix it up by walking the array looking for the ID.

- gard: Set chip generation based on PVR

  Currently we assume that this tool is being used on a P8 system by default and allow the user to override this behaviour using the -8 and -9 command line arguments. When running on the host we can use the PVR to guess what chip generation so do that.

  This also changes the default behaviour to assume that the host is a P9 when running on an ARM system. This tool didn't even work when compiled for ARM until recently and the OpenBMC vPNOR hack that we have currently is broken for P9 systems that don't use vPNOR (Zaius and Romulus).

- gard: Allow records with an ID of 0xffffffff

  We currently assume that a record with an ID of 0xffffffff is invalid. Apparently this is incorrect and we should display these records, so expand the check to compare the entire record with 0xff rather than just the ID.

- gard: create: Allow creating arbitrary GARD records

  Add a new sub-command that allows us to create GARD records for arbitrary chip units. There isn't a whole lot of constraints on this and that limits how useful it can be, but it does allow a user to GARD out individual DIMMs, chips or cores from the BMC (or host) if needed.

  There are a few caveats though:

  1) Not everything can, or should, have a GARD record applied it to.
  2) There is no validation that the unit actually exists. Doing that sort of validation requires something that understands the FAPI targeting information (I think) and adding support for it here would require some knowledge from the system XML file.
  3) There's no way to get a list of paths in the system.
  4) Although we can create a GARD record at runtime it won't be applied until the next IPL.

- gard: Add path parsing support

  In order to support manual GARD records we need to be able to parse the hardware unit path strings. This patch implements that.

- gard: list: Improve output

  Display the full path to the GARDed hardware unit in each record rather than relying on the output of *gard show* and convert do_list() to use the iterator while we're here.

- gard: {list, show}: Fix the Type field in the output

  The output of *gard list* has a field named "Type", however this doesn't actually indicate the type of the record. Rather, it shows the type of the path used to identify the hardware being GARDed. This is of pretty dubious value considering the Physical path seems to always be used when referring to GARDed hardware.

- gard: Add P9 support

- gard: Update chip unit data

  Source the list of units from the hostboot source rather than the previous hard coded list. The list of path element types changes between generations so we need to add a level of indirection to accommodate P9. This also changes the names used to match those printed by Hostboot at IPL time and paves the way to adding support for manual GARD record creation.

- gard: show: Remove "Res Recovery" field

  This field has never been populated by hostboot on OpenPower systems so there's no real point in reporting it's contents.

libflash / pflash

Anybody shipping libflash or pflash to interact with POWER9 systems must upgrade to this version.

Since skiboot-5.10-rc2:

- pflash: Fix makefile dependency issue

Since skiboot-5.9:

- pflash: Support for volatile flag

  The volatile flag was added to the PNOR image to indicate partitions that are cleared during a host power off. Display this flag from the pflash command.

- pflash: Support for clean_on_ecc_error flag

  Add the misc flag clear_on_ecc_error to libflash/pflash. This was the only missing flag. The generator of the virtual PNOR image relies on libflash/pflash to provide the partition information, so all flags are needed to build an accurate virtual PNOR partition table.

- pflash: Respect write(2) return values

  The write(2) system call returns the number of bytes written, this is important since it is entitled to write less than what we requested. Currently we ignore the return value and assume it wrote everything we requested. While in practice this is likely to always be the case, it isn't actually correct.

- external/pflash: Fix erasing within a single erase block

  It is possible to erase within a single erase block. Currently the pflash code assumes that if the erase starts part way into an erase block it is because it needs to be aligned up to the boundary with the next erase block.

  Doing an erase smaller than a single erase block will cause underflows and looping forever on erase.

- external/pflash: Fix non-zero return code for successful read when size%256 != 0

  When performing a read the return value from pflash is non-zero, even for a successful read, when the size being read is not a multiple of 256. This is because do_read_file returns the value from the write system call which is then returned by pflash. When the size is a multiple of 256 we get lucky in that this wraps around back to zero. However for any other value the return code is size % 256. This means even when the operation is successful the return code will seem to reflect an error.

  Fix this by returning zero if the entire size was read correctly, otherwise return the corresponding error code.

- libflash: Fix parity calculation on ARM

  To calculate the ECC syndrome we need to calculate the parity of a 64bit number. On non-powerpc platforms we use the GCC builtin function __builtin_parityl() to do this calculation. This is broken on 32bit ARM where sizeof(unsigned long) is four bytes. Using __builtin_parityll() instead cures this.

- libflash/mbox-flash: Add the ability to lock flash

- libflash/mbox-flash: Understand v3
- libflash/mbox-flash: Use BMC suggested timeout value
- libflash/mbox-flash: Simplify message sending

  hw/lpc-mbox no longer requires that the memory associated with messages exist for the lifetime of the message. Once it has been sent to the BMC, that is bmc_mbox_enqueue() returns, lpc-mbox does not need the message to continue to exist. On the receiving side, lpc-mbox will ensure that a message exists for the receiving callback function.

  Remove all code to deal with allocating messages.

- hw/lpc-mbox: Simplify message bookkeeping and timeouts

  Currently the hw/lpc-mbox layer keeps a pointer for the currently in-flight message for the duration of the mbox call. This creates problems when messages timeout, is that pointer still valid, what can we do with it. The memory is owned by the caller but if the caller has declared a timeout, it may have freed that memory.

  Another problem is locking. This patch also locks around sending and receiving to avoid races with timeouts and possible resends. There was some locking previously which was likely insufficient - definitely too hard to be sure is correct

  All this is made much easier with the previous rework which moves sequence number allocation and verification into lpc-mbox rather than the caller.

- libflash/mbox-flash: Allow mbox-flash to tell the driver msg timeouts

  Currently when mbox-flash decides that a message times out the driver has no way of knowing to drop the message and will continue waiting for a response indefinitely preventing more messages from ever being sent.

  This is a problem if the BMC crashes or has some other issue where it won't ever respond to our outstanding message.

  This patch provides a method for mbox-flash to tell the driver how long it should wait before it no longer needs to care about the response.

- libflash/mbox-flash: Move sequence handling to driver level
- libflash/mbox-flash: Always close windows before opening a new window

  The MBOX protocol states that if an open window command fails then all open windows are closed. Currently, if an open window command fails mbox-flash will erroneously assume that the previously open window is still open.

  The solution to this is to mark all windows as closed before issuing an open window command and then on success we'll mark the new window as open.

- libflash/mbox-flash: Add v2 error codes

opal-prd

Anybody shipping *opal-prd* for POWER9 systems must upgrade *opal-prd* to this new version.

- prd: Log unsupported message type

  Useful for debugging.

  Sample output:

  ```
  [29155.157050283,7] PRD: Unsupported prd message type : 0xc
  ```

- opal-prd: occ: Add support for runtime OCC load/start in ZZ

  This patch adds support to handle OCC load/start event from FSP/PRD. During IPL we send a success directly to FSP without invoking any HBRT load routines on receiving OCC load mbox message from FSP. At runtime we forward this event to host opal-prd.

  This patch provides support for invoking OCC load/start HBRT routines like load_pm_complex() and start_pm_complex() from opal-prd.

- opal-prd: Add support for runtime OCC reset in ZZ

  This patch handles OCC_RESET runtime events in host opal-prd and also provides support for calling 'hostinterface->wakeup()' which is required for doing the reset operation.

- prd: Enable error logging via firmware_request interface

  In P9 HBRT sends error logs to FSP via firmware_request interface. This patch adds support to parse error log and send it to FSP.

- prd: Add generic response structure inside prd_fw_msg

  This patch adds generic response structure. Also sync prd_fw_msg type macros with hostboot.

- opal-prd: flush after logging to stdio in debug mode

  When in debug mode, flush after each log output. This makes it more likely that we'll catch failure reasons on severe errors.

Debugging and reliability improvements

Since skiboot-5.10-rc3:

- increase log verbosity in debug builds
- Add -debug to version on DEBUG builds
- cpu_wait_job: Correctly report time spent waiting for job

Since skiboot-5.10-rc2:

- ATTN: Enable flush instruction cache bit in HID register

  In P9, we have to enable "flush the instruction cache" bit along with "attn instruction support" bit to trigger attention.

Since skiboot-5.10-rc1:

- core/init: manage MSR[ME] explicitly, always enable

  The current boot sequence inherits MSR[ME] from the IPL firmware, and never changes it. Some environments disable MSR[ME] (e.g., mambo), and others can enable it (hostboot).

  This has two problems. First, MSR[ME] must be disabled while in the process of taking over the interrupt vector from the previous environment. Second, after installing our machine check handler, MSR[ME] should be enabled to get some useful output rather than a checkstop.

- core/exception: beautify exception handler, add MCE-involved registers

  Print DSISR and DAR, to help with deciphering machine check exceptions, and improve the output a bit, decode NIP symbol, improve alignment, etc. Also print a specific header for machine check, because we do expect to see these if there is a hardware failure.

  Before:

```
[    0.005968779,3] ***********************************************
[    0.005974102,3] Unexpected exception 200 !
[    0.005978696,3] SRR0 : 000000003002ad80 SRR1 : 9000000000001000
[    0.005985239,3] HSRR0: 00000000300027b4 HSRR1: 9000000030001000
[    0.005991782,3] LR   : 000000003002ad80 CTR  : 0000000000000000
[    0.005998130,3] CFAR : 00000000300b58bc
[    0.006002769,3] CR   : 40000004  XER: 20000000
[    0.006008069,3] GPR00: 000000003002ad80 GPR16: 0000000000000000
[    0.006015170,3] GPR01: 0000000031c03bd0 GPR17: 0000000000000000
[...]
```

After:

```
[    0.003287941,3] ***********************************************
[    0.003561769,3] Fatal MCE at 000000003002ad80   .nvram_init+0x24
[    0.003579628,3] CFAR : 00000000300b5964
[    0.003584268,3] SRR0 : 000000003002ad80 SRR1 : 9000000000001000
[    0.003590812,3] HSRR0: 00000000300027b4 HSRR1: 9000000030001000
[    0.003597355,3] DSISR: 00000000         DAR  : 0000000000000000
[    0.003603480,3] LR   : 000000003002ad68 CTR  : 0000000030093d80
[    0.003609930,3] CR   : 40000004         XER  : 20000000
[    0.003615698,3] GPR00: 00000000300149e8 GPR16: 0000000000000000
[    0.003622799,3] GPR01: 0000000031c03bc0 GPR17: 0000000000000000
[...]
```

Since skiboot-5.9:

- lock: Add additional lock auditing code

  Keep track of lock owner name and replace lock_depth counter with a per-cpu list of locks held by the cpu.

  This allows us to print the actual locks held in case we hit the (in)famous message about opal_pollers being run with a lock held.

  It also allows us to warn (and drop them) if locks are still held when returning to the OS or completing a scheduled job.

- Add support for new GCC 7 parametrized stack protector

  This gives us per-cpu guard values as well. For now I just XOR a magic constant with the CPU PIR value.

- Mambo: run hello_world and sreset_world tests with Secure and Trusted Boot

  We *disable* the secure boot part, but we keep the verified boot part as we don't currently have container verification code for Mambo.

  We can run a small part of the code currently though.

- core/flash.c: extern function to get the name of a PNOR partition

  This adds the flash_map_resource_name() to allow skiboot subsystems to lookup the name of a PNOR partition. Thus, we don't need to duplicate the same information in other places (e.g. libstb).

- libflash/mbox-flash: only wait for MBOX_DEFAULT_POLL_MS if busy

  This makes the mbox unit test run 300x quicker and seems to shave about 6 seconds from boot time on Witherspoon.

- make check: Make valgrind optional

  To (slightly) lower the barrier for contributions, we can make valgrind optional with just a small amount of plumbing.

This allows make check to run successfully without valgrind.

- libflash/test: Add tests for mbox-flash

  A first basic set of tests for mbox-flash. These tests do their testing by stubbing out or otherwise replacing functions not in libflash/mbox-flash.c. The stubbed out version of the function can then be used to emulate a BMC mbox daemon talking to back to the code in mbox-flash and it can ensure that there is some adherence to the protocol and that from a block-level api point of view the world appears sane.

  This makes these tests simple to run and they have been integrated into *make check*. The down side is that these tests rely on duplicated feature incomplete BMC daemon behaviour. Therefore these tests are a strong indicator of broken behaviour but a very unreliable indicator of correctness.

  Full integration tests with a 'real' BMC daemon are probably beyond the scope of this repository.

- external/test/test.sh: fix VERSION substitution when no tags

  i.e. we get a hash rather than a version number

  This seems to be occurring in Travis if it doesn't pull a tag.

- external/test: make stripping out version number more robust

  For some bizarre reason, Travis started failing on this substitution when there'd been zero code changes in this area… This at least papers over whatever the problem is for the time being.

- io: Add load_wait() helper

  This uses the standard form twi/isync pair to ensure a load is consumed by the core before continuing. This can be necessary under some circumstances for example when having the following sequence:

  - Store reg A
  - Load reg A (ensure above store pushed out)
  - delay loop
  - Store reg A

  I.E., a mandatory delay between 2 stores. In theory the first store is only guaranteed to reach the device after the load from the same location has completed. However the processor will start executing the delay loop without waiting for the return value from the load.

  This construct enforces that the delay loop isn't executed until the load value has been returned.

- chiptod: Keep boot timestamps contiguous

  Currently we reset the timebase value to (almost) zero when synchronising the timebase of each chip to the Chip TOD network which results in this:

```
[   42.374813167,5] CPU: All 80 processors called in...
[    2.222791151,5] FLASH: Found system flash: Macronix MXxxL51235F id:0
[    2.222977933,5] BT: Interface initialized, IO 0x00e4
```

  This patch modifies the chiptod_init() process to use the current timebase value rather than resetting it to zero. This results in the timestamps remaining contiguous from the start of hostboot until the petikernel starts. e.g.

```
[   70.188811484,5] CPU: All 144 processors called in...
[   72.458004252,5] FLASH: Found system flash:  id:0
[   72.458147358,5] BT: Interface initialized, IO 0x00e4
```

- hdata/spira: Add missing newline to prlog() call

  We're missing a n here.

- opal/xscom: Add recovery for lost core wakeup SCOM failures.

  Due to a hardware issue where core responding to SCOM was delayed due to thread reconfiguration, leaves the SCOM logic in a state where the subsequent SCOM to that core can get errors. This is affected for Core PC SCOM registers in the range of 20010A80-20010ABF

  The solution is if a xscom timeout occurs to one of Core PC SCOM registers in the range of 20010A80-20010ABF, a clearing SCOM write is done to 0x20010800 with data of '0x00000000' which will also get a timeout but clears the SCOM logic errors. After the clearing write is done the original SCOM operation can be retried.

  The SCOM timeout is reported as status 0x4 (Invalid address) in HMER[21-23].

- opal/xscom: Move the delay inside xscom_reset() function.

  So caller of xscom_reset() does not have to bother about adding a delay separately. Instead caller can control whether to add a delay or not using second argument to xscom_reset().

- timer: Stop calling list_top() racily

  This will trip the debug checks in debug builds under some circumstances and is actually a rather bad idea as we might look at a timer that is concurrently being removed and modified, and thus incorrectly assume there is no work to do.

- fsp: Bail out of HIR if FSP is resetting voluntarily

  a. Surveillance response times out and OPAL triggers a HIR
  b. Before the HIR process kicks in, OPAL gets a PSI interrupt indicating link down
  c. HIR process continues and OPAL tries to write to DRCR; PSI link inactive => xstop

  OPAL should confirm that the FSP is not already in reset in the HIR path.

- sreset_kernel: only run SMT tests due to not supporting re-entry
- Use systemsim-p9 v1.1
- direct-controls: enable fast reboot direct controls for mambo

  Add mambo direct controls to stop threads, which is required for reliable fast-reboot. Enable direct controls by default on mambo.

- core/opal: always verify cpu->pir on entry
- asm/head: add entry/exit calls

  Add entry and exit C functions that can do some more complex checks before the opal proper call. This requires saving off volatile registers that have arguments in them.

- core/lock: improve bust_locks

  Prevent try_lock from modifying the lock state when bust_locks is set. unlock will not unlock it in that case, so locks will get taken and never released while bust_locks is set.

- hw/occ: Log proper SCOM register names

  This patch fixes the logging of incorrect SCOM register names.

- mambo: Add support for NUMA

  Currently the mambo scripts can do multiple chips, but only the first ever has memory.

  This patch adds support for having memory on each chip, with each appearing as a separate NUMA node. Each node gets MEM_SIZE worth of memory.

  It's opt-in, via `export MAMBO_NUMA=1`.

- external/mambo: Switch qtrace command to use plug-ins

  The plug-in seems to be the preferred way to do this now, it works better, and the qtracer emitter seems to generate invalid traces in new mambo versions.

- asm/head: Loop after attn

  We use the attn instruction to raise an error in early boot if OPAL don't recognise the PVR. It's possible for hostboot to disable the attn instruction before entering OPAL so add an extra busy loop after the attn to prevent attempting to boot on an unknown processor.

Contributors

- 302 csets from 32 developers
- 3 employers found
- A total of 15919 lines added, 4786 removed (delta 11133)

Extending the analysis done for some previous releases, we can see our trends in code review across versions:

Release	csets	Ack %	Reviews %	Tested %	Reported %
5.0	329	15 (5%)	20 (6%)	1 (0%)	0 (0%)
5.1	372	13 (3%)	38 (10%)	1 (0%)	4 (1%)
5.2-rc1	334	20 (6%)	34 (10%)	6 (2%)	11 (3%)
5.3-rc1	302	36 (12%)	53 (18%)	4 (1%)	5 (2%)
5.4	361	16 (4%)	28 (8%)	1 (0%)	9 (2%)
5.5	408	11 (3%)	48 (12%)	14 (3%)	10 (2%)
5.6	87	12 (14%)	6 (7%)	5 (6%)	2 (2%)
5.7	232	30 (13%)	32 (14%)	5 (2%)	2 (1%)
5.8	157	13 (8%)	36 (23%)	2 (1%)	6 (4%)
5.9	209	15 (7%)	78 (37%)	3 (1%)	10 (5%)
5.10	302	20 (6%)	62 (21%)	24 (8%)	11 (4%)

The review count for v5.9 is largely bogus, there was a series of 25 whitespace patches that got "Reviewed-by" and if we exclude them, we're back to 14%, which is more like what I'd expect.

For 5.10, We've seen an increase in Reviewed-by from 5.9, back to closer to 5.8 levels. I'm hoping we can keep the ~20% up.

Initially I was really pleased with the increase in Tested-by, but with closer examination, 17 of those are actually from various automated testing on commits to code we bring in from hostboot/other firmware components. When you exclude them, we're back down to 2% getting Tested-by, which isn't great.

Developers with the most changesets

Developer	#	%
Stewart Smith	40	(13.2%)
Nicholas Piggin	37	(12.3%)
Oliver O'Halloran	36	(11.9%)
Benjamin Herrenschmidt	23	(7.6%)
Claudio Carvalho	20	(6.6%)
Cyril Bur	19	(6.3%)

Continued on next page

Table 1 – continued from previous page

Developer	#	%
Michael Neuling	13	(4.3%)
Shilpasri G Bhat	12	(4.0%)
Reza Arbab	12	(4.0%)
Pridhiviraj Paidipeddi	11	(3.6%)
Vasant Hegde	10	(3.3%)
Akshay Adiga	10	(3.3%)
Mahesh Salgaonkar	8	(2.6%)
Russell Currey	7	(2.3%)
Alistair Popple	7	(2.3%)
Vaibhav Jain	5	(1.7%)
Prem Shanker Jha	4	(1.3%)
Robert Lippert	4	(1.3%)
Frédéric Bonnard	3	(1.0%)
Christophe Lombard	3	(1.0%)
Jeremy Kerr	2	(0.7%)
Michael Ellerman	2	(0.7%)
Balbir Singh	2	(0.7%)
Andrew Donnellan	2	(0.7%)
Madhavan Srinivasan	2	(0.7%)
Adriana Kobylak	2	(0.7%)
Sukadev Bhattiprolu	1	(0.3%)
Alexey Kardashevskiy	1	(0.3%)
Frederic Barrat	1	(0.3%)
Ananth N Mavinakayanahalli	1	(0.3%)
Suraj Jitindar Singh	1	(0.3%)
Guilherme G. Piccoli	1	(0.3%)

Developers with the most changed lines

Developer	#	%
Stewart Smith	4284	(24.5%)
Nicholas Piggin	2924	(16.7%)
Claudio Carvalho	2476	(14.2%)
Shilpasri G Bhat	1490	(8.5%)
Cyril Bur	1475	(8.4%)
Oliver O'Halloran	1242	(7.1%)
Benjamin Herrenschmidt	736	(4.2%)
Alistair Popple	498	(2.8%)
Vasant Hegde	299	(1.7%)
Akshay Adiga	273	(1.6%)
Reza Arbab	231	(1.3%)
Mahesh Salgaonkar	225	(1.3%)
Balbir Singh	213	(1.2%)
Frédéric Bonnard	169	(1.0%)
Michael Neuling	142	(0.8%)
Robert Lippert	97	(0.6%)
Pridhiviraj Paidipeddi	93	(0.5%)
Prem Shanker Jha	92	(0.5%)

Continued on next page

Table 2 – continued from previous page

Developer	#	%
Christophe Lombard	80	(0.5%)
Russell Currey	78	(0.4%)
Michael Ellerman	72	(0.4%)
Adriana Kobylak	71	(0.4%)
Madhavan Srinivasan	61	(0.3%)
Sukadev Bhattiprolu	58	(0.3%)
Vaibhav Jain	52	(0.3%)
Jeremy Kerr	27	(0.2%)
Ananth N Mavinakayanahalli	16	(0.1%)
Frederic Barrat	9	(0.1%)
Andrew Donnellan	5	(0.0%)
Alexey Kardashevskiy	3	(0.0%)
Suraj Jitindar Singh	1	(0.0%)
Guilherme G. Piccoli	1	(0.0%)

Developers with the most lines removed

Developer	#	%
Alistair Popple	304	(6.4%)
Andrew Donnellan	1	(0.0%)

Developers with the most signoffs

Developer	#	%
Stewart Smith	262	(99.2%)
Reza Arbab	1	(0.4%)
Mahesh Salgaonkar	1	(0.4%)

Developers with the most reviews

Developer	#	%
Andrew Donnellan	8	(13.6%)
Balbir Singh	5	(8.5%)
Vasant Hegde	5	(8.5%)
Gregory S. Still	4	(6.8%)
Nicholas Piggin	4	(6.8%)
Reza Arbab	3	(5.1%)
Alistair Popple	3	(5.1%)
RANGANATHPRASAD G. BRAHMASAMUDRA	3	(5.1%)
Jennifer A. Stofer	3	(5.1%)
Oliver O'Halloran	3	(5.1%)
Vaidyanathan Srinivasan	2	(3.4%)
Hostboot Team	2	(3.4%)
Christian R. Geddes	2	(3.4%)
Frederic Barrat	2	(3.4%)
Cyril Bur	2	(3.4%)
Stewart Smith	1	(1.7%)
Cédric Le Goater	1	(1.7%)
Samuel Mendoza-Jonas	1	(1.7%)
Daniel M. Crowell	1	(1.7%)
Vaibhav Jain	1	(1.7%)
Madhavan Srinivasan	1	(1.7%)
Michael Ellerman	1	(1.7%)
Shilpasri G Bhat	1	(1.7%)
Total	59	(100%)

Developers with the most test credits

Developer	#	%
FSP CI Jenkins	4	(16.7%)
Jenkins Server	4	(16.7%)
Hostboot CI	4	(16.7%)
Oliver O'Halloran	3	(12.5%)
Jenkins OP Build CI	3	(12.5%)
Jenkins OP HW	2	(8.3%)
Pridhiviraj Paidipeddi	2	(8.3%)
Andrew Donnellan	1	(4.2%)
Vaidyanathan Srinivasan	1	(4.2%)
Total	24	(100%)

Developers who gave the most tested-by credits

Developer	#	%
Prem Shanker Jha	17	(70.8%)
Benjamin Herrenschmidt	3	(12.5%)
Stewart Smith	2	(8.3%)
Shilpasri G Bhat	1	(4.2%)
Ananth N Mavinakayanahalli	1	(4.2%)
Total	24	(100%)

Developers with the most report credits

Developer	#	%
Pridhiviraj Paidipeddi	2	(18.2%)
Benjamin Herrenschmidt	1	(9.1%)
Andrew Donnellan	1	(9.1%)
Michael Ellerman	1	(9.1%)
Deb McLemore	1	(9.1%)
Brad Bishop	1	(9.1%)
Michel Normand	1	(9.1%)
Hugo Landau	1	(9.1%)
Minda Wei	1	(9.1%)
Francesco A Campisano	1	(9.1%)
Total	11	(100%)

Developers who gave the most report credits

Developer	#	%
Stewart Smith	7	(63.6%)
Suraj Jitindar Singh	1	(9.1%)
Jeremy Kerr	1	(9.1%)
Michael Neuling	1	(9.1%)
Frédéric Bonnard	1	(9.1%)
Total	11	(100%)

Changesets and Employers

Top changeset contributors by employer:

Employer	#	%
IBM	298	(98.7%)
Google	3	(1.0%)
(Unknown)	1	(0.3%)

Top lines changed by employer:

Employer	#	%
IBM	17396	(99.4%)
Google	73	(0.4%)
(Unknown)	24	(0.1%)

Employers with the most signoffs (total 264):

Employer	#	%
IBM	264	(100.0%)

Employers with the most hackers (total 33)

Employer	#	%
IBM	31	(93.9%)
Google	1	(3.0%)
(Unknown)	1	(3.0%)

5.1.30 skiboot-5.10-rc1

skiboot v5.10-rc1 was released on Tuesday February 6th 2018. It is the first release candidate of skiboot 5.10, which will become the new stable release of skiboot following the 5.9 release, first released October 31st 2017.

skiboot v5.10-rc1 contains all bug fixes as of *skiboot-5.9.8* and *skiboot-5.4.9* (the currently maintained stable releases). There may be more 5.9.x stable releases, it will depend on demand.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.10 in February, with skiboot 5.10 being for all POWER8 and POWER9 platforms in op-build v1.21. This release will be targeted to early POWER9 systems.

Over skiboot-5.9, we have the following changes:

New Features

- hdata: Parse IPL FW feature settings

  Add parsing for the firmware feature flags in the HDAT. This indicates the settings of various parameters which are set at IPL time by firmware.

- opal/xstop: Use nvram option to enable/disable sw checkstop.

  Add a mechanism to enable/disable sw checkstop by looking at nvram option opal-sw-xstop=<enable/disable>.

  For now this patch disables the sw checkstop trigger unless explicitly enabled through nvram option 'opal-sw-xstop=enable'i for p9. This will allow an opportunity to get host kernel in panic path or xmon for unrecoverable HMIs or MCE, to be able to debug the issue effectively.

  To enable sw checkstop in opal issue following command:

  ```
  nvram -p ibm,skiboot --update-config opal-sw-xstop=enable
  ```

  NOTE: This is a workaround patch to disable sw checkstop by default to gain control in host kernel for better checkstop debugging. Once we have most of the checkstop issues stabilized/resolved, revisit this patch to enable sw checkstop by default.

For p8 platform it will remain enabled by default unless explicitly disabled.

To disable sw checkstop on p8 issue following command:

```
nvram -p ibm,skiboot --update-config opal-sw-xstop=disable
```

- hdata: Parse SPD data

  Parse SPD data and populate device tree.

  list of properties parsing from SPD:

  ```
  [root@ltc-wspoon dimm@d00f]# lsprop .
  memory-id        0000000c (12)      # DIMM type
  product-version  00000032 (50)      # Module Revision Code
  device_type      "memory-dimm-ddr4"
  serial-number    15d9acb6 (366587062)
  status           "okay"
  size             00004000 (16384)
  phandle          000000bd (189)
  ibm,loc-code     "UOPWR.0000000-Node0-DIMM7"
  part-number      "36ASF2G72PZ-2G6B2   "
  reg              0000d007 (53255)
  name             "dimm"
  manufacturer-id  0000802c (32812)  # Vendor ID, we can get vendor name␣
  ↪from this ID
  ```

  Also update documentation.

- hdata: Add memory hierarchy under xscom node

  We have memory to chip mapping but doesn't have complete memory hierarchy. This patch adds memory hierarchy under xscom node. This is specific to P9 system as these hierarchy may change between processor generation.

  It uses memory controller ID details and populates nodes like: xscom@<addr>/mcbist@<mcbist_id>/mcs@<mcs_id>/mca@

  Also this patch adds few properties under dimm node. Finally make sure xscom nodes created before calling memory_parse().

Fast Reboot and Quiesce

We have a preliminary fast reboot implementation for POWER9 systems, which we look to enabling by default in the next release.

The OPAL Quiesce calls are designed to improve reliability and debuggability around reboot and error conditions. See the full API documentation for details: *OPAL_QUIESCE*.

- fast-reboot: bare bones fast reboot implementation for POWER9

  This is an initial fast reboot implementation for p9 which has only been tested on the Witherspoon platform, and without the use of NPUs, NX/VAS, etc.

  This has worked reasonably well so far, with no failures in about 100 reboots. It is hidden behind the traditional fast-reboot experimental nvram option, until more platforms and configurations are tested.

- fast-reboot: move boot CPU clean-up logically together with secondaries

  Move the boot CPU clean-up and state transition to active, logically together with secondaries. Don't release secondaries from fast reboot hold until everyone has cleaned up and transitioned to active.

  This is cosmetic, but it is helpful to run the fast reboot state machine the same way on all CPUs.

- fast-reboot: improve failure error messages

  Change existing failure error messages to PR_NOTICE so they get printed to the console, and add some new ones. It's not a more severe class because it falls back to IPL on failure.

- fast-reboot: quiesce opal before initiating a fast reboot

  Switch fast reboot to use quiescing rather than "wait for a while".

  If firmware can not be quiesced, then fast reboot is skipped. This significantly improves the robustness of fast reboot in the face of bugs or unexpected latencies.

  Complexity of synchronization in fast-reboot is reduced, because we are guaranteed to be single-threaded when quiesce succeeds, so locks can be removed.

  In the case that firmware can be quiesced, then it will generally reduce fast reboot times by nearly 200ms, because quiescing usually takes very little time.

- core: Add support for quiescing OPAL

  Quiescing is ensuring all host controlled CPUs (except the current one) are out of OPAL and prevented from entering. This can be use in debug and shutdown paths, particularly with system reset sequences.

  This patch adds per-CPU entry and exit tracking for OPAL calls, and adds logic to "hold" or "reject" at entry time, if OPAL is quiesced.

  An OPAL call is added, to expose the functionality to Linux, where it can be used for shutdown, kexec, and before generating sreset IPIs for debugging (so the debug code does not recurse into OPAL).

- dctl: p9 increase thread quiesce timeout

  We require all instructions to be completed before a thread is considered stopped, by the dctl interface. Long running instructions like cache misses and CI loads may take a significant amount of time to complete, and timeouts have been observed in stress testing.

  Increase the timeout significantly, to cover this. The workbook just says to poll, but we like to have timeouts to avoid getting stuck in firmware.

POWER9 power saving

There is much improved support for deeper sleep/idle (stop) states on POWER9.

- OCC: Increase max pstate check on P9 to 255

  This has changed from P8, we can now have > 127 pstates.

  This was observed on Boston during WoF bring up.

- SLW: Add idle state stop5 for DD2.0 and above

  Adding stop5 idle state with rough residency and latency numbers.

- SLW: Add p9_stop_api calls for IMC

  Add p9_stop_api for EVENT_MASK and PDBAR scoms. These scoms are lost on wakeup from stop11.

- SCOM restore for DARN and XIVE

  While waking up from stop11, we want NCU_DARN_BAR to have enable bit set. Without this stop_api call, the value restored is without enable bit set. We loose NCU_SPEC_BAR when the quad goes into stop11, stop_api will restore while waking up from stop11.

- SLW: Call p9_stop_api only if deep_states are enabled

  All init time p9_stop_api calls have been isolated to slw_late_init. If p9_stop_api fails, then the deep states can be excluded from device tree.

  For p9_stop_api called after device-tree for cpuidle is created , has_deep_states will be used to check if this call is even required.

- Better handle errors in setting up sleep states (p9_stop_api)

  We won't put affected stop states in the device tree if the wakeup engine is not present or has failed.

- SCOM Restore: Increased the EQ SCOM restore limit.

  Commit increases the SCOM restore limit from 16 to 31.

- hw/dts: retry special wakeup operation if core still gated

  It has been observed that in some cases the special wakeup operation can "succeed" but the core is still in a gated/offline state.

  Check for this state after attempting to wakeup a core and retry the wakeup if necessary.

- core/direct-controls: add function to read core gated state
- core/direct-controls: wait for core special wkup bit cleared

  When clearing special wakeup bit on a core, wait until the bit is actually cleared by the hardware in the status register until returning success.

  This may help avoid issues with back-to-back reads where the special wakeup request is cleared but the firmware is still processing the request and the next attempt to set the bit reads an immediate success from the previous operation.

- p9_stop_api: PM: Added support for version control in SCOM restore entries.
  - adds version info in SCOM restore entry header
  - adds version specific details in SCOM restore entry header
  - retains old behaviour of SGPE Hcode's base version
- p9_stop_api: EQ SCOM Restore: Introduced version control in SCOM restore entry.
  - introduces version control in header of SCOM restore entry
  - ensures backward compatibility
  - introduces flexibility to handle any number of SCOM restore entry.

Secure and Trusted Boot for POWER9

We introduce support for Secure and Trusted Boot for POWER9 systems, with equal functionality that we have on POWER8 systems, that is, we have the mechanisms in place to boot to petitboot (i.e. to BOOTKERNEL).

See the *Secure and Trusted Boot Library (LibSTB) Documentation* for full documentation of OPAL secure and trusted boot.

- allow secure boot if not enforcing it

  We check the secure boot containers no matter what, only *enforcing* secure boot if we're booting in secure mode. This gives us an extra layer of checking firmware is legit even when secure mode isn't enabled, as well as being really useful for testing.

- libstb/(create|print)-container: Sync with sb-signing-utils

  The sb-signing-utils project has improved upon the skeleton create-container tool that existed in skiboot, including being able to (quite easily) create *signed* images.

  This commit brings in that code (and makes it build in the skiboot build environment) and updates our skiboot.*.stb generating code to use the development keys. This means that by default, skiboot build process will let you build firmware that can do a secure boot with *development* keys.

  See *Signing Firmware Code* for details on firmware signing.

  We also update print-container as well, syncing it with the upstream project.

  Derived from github.com:open-power/sb-signing-utils.git at v0.3-5-gcb111c03ad7f (Some discussion ongoing on the changes, another sync will come shortly)

- doc: update libstb documentation with POWER9 changes. See: *Secure and Trusted Boot Library (LibSTB) Documentation*.

  POWER9 changes reflected in the libstb:

  – bumped ibm,secureboot node to v2

  – added ibm,cvc node

  – hash-algo superseded by hw-key-hash-size

- libstb/cvc: update memory-region to point to /reserved-memory

  The linux documentation, reserved-memory.txt, says that memory-region is a phandle that pairs to a children of /reserved-memory.

  This updates /ibm,secureboot/ibm,cvc/memory-region to point to /reserved-memory/secure-crypt-algo-code instead of /ibm,hostboot/reserved-memory/secure-crypt-algo-code.

- libstb: add support for ibm,secureboot-v2

  ibm,secureboot-v2 changes:

  – The Container Verification Code is represented by the ibm,cvc node.

  – Each ibm,cvc child describes a CVC service.

  – hash-algo is superseded by hw-key-hash-size.

- hdata/tpmrel.c: add ibm, cvc device tree node

  In P9, the Container Verification Code is stored in a hostboot reserved memory and the list of provided CVC services is stored in the TPMREL_IDATA_HASH_VERIF_OFFSETS idata array. Each CVC service has an offset and version.

  This adds the ibm,cvc device tree node and its documentation.

- hdata/tpmrel.c: add firmware event log info to the tpm node

  This parses the firmware event log information from the secureboot_tpm_info HDAT structure and add it to the tpm device tree node.

  There can be multiple secureboot_tpm_info entries with each entry corresponding to a master processor that has a tpm device, however, multiple tpm is not supported.

- hdata/spira: add ibm,secureboot node in P9

  In P9, skiboot builds the device tree from the HDAT. These are the “ibm,secureboot” node changes compared to P8:

  - The Container-Verification-Code (CVC), a.k.a. ROM code, is no longer stored in a secure ROM with static address. In P9, it is stored in a hostboot reserved memory and each service provided also has a version, not only an offset.
  - The hash-algo property is not provided via HDAT, instead it provides the hw-key-hash-size, which is indeed the information required by the CVC to verify containers.

  This parses the iplparams_sysparams HDAT structure and creates the "ibm,secureboot", which is bumped to "ibm,secureboot-v2".

  In "ibm,secureboot-v2":

  - hash-algo property is superseded by hw-key-hash-size.
  - container verification code is explicitly described by a child node. Added in a subsequent patch.

  See *ibm,secureboot* for documentation.

- libstb/tpm_chip.c: define pr_fmt and fix messages logged

  This defines pr_fmt and also fix messages logged:

  - EV_SEPARATOR instead of 0xFFFFFFFF
  - when an event is measured it also prints the tpm id, event type and event log length

  Now we can filter the messages logged by libstb and its sub-modules by running:

  ```
  grep STB /sys/firmware/opal/msglog
  ```

- libstb/tss: update the list of event types supported

  Skiboot, precisely the tpmLogMgr, initializes the firmware event log by calculating its length so that a new event can be recorded without exceeding the log size. In order to calculate the size, it walks through the log until it finds a specific event type. However, if the log has an unknown event type, the tpmLogMgr will not be able to reach the end of the log.

  This updates the list of event types with all of those supported by hostboot. Thus, skiboot can properly calculate the event log length.

- tpm_i2c_nuvoton: add nuvoton, npct601 to the compatible property

  The linux kernel doesn't have a driver compatible with "nuvoton,npct650", but it does have for "nuvoton,npct601", which should also be compatible with npct650.

  This adds "nuvoton,npct601" to the compatible devtree property.

- libstb/trustedboot.c: import stb_final() from stb.c

  The stb_final() primary goal is to measure the event EV_SEPARATOR into PCR[0-7] when trusted boot is about to exit the boot services.

  This imports the stb_final() from stb.c into trustedboot.c, but making the following changes:

  - Rename it to trustedboot_exit_boot_services().
  - As specified in the TCG PC Client specification, EV_SEPARATOR events must be logged with the name 0xFFFFFF.
  - Remove the ROM driver clean-up call.
  - Don't allow code to be measured in skiboot after trustedboot_exit_boot_services() is called.

- libstb/cvc.c: import softrom behaviour from drivers/sw_driver.c

  Softrom is used only for testing with mambo. By setting compatible="ibm,secureboot-v1-softrom" in the "ibm,secureboot" node, firmware images can be properly measured even if the Container-Verification-Code

(CVC) is not available. In this case, the mbedtls_sha512() function is used to calculate the sha512 hash of the firmware images.

This imports the softrom behaviour from libstb/drivers/sw_driver.c code into cvc.c, but now softrom is implemented as a flag. When the flag is set, the wrappers for the CVC services work the same way as in sw_driver.c.

- libstb/trustedboot.c: import tb_measure() from stb.c

  This imports tb_measure() from stb.c, but now it calls the CVC sha512 wrapper to calculate the sha512 hash of the firmware image provided.

  In trustedboot.c, the tb_measure() is renamed to trustedboot_measure().

  The new function, trustedboot_measure(), no longer checks if the container payload hash calculated at boot time matches with the hash found in the container header. A few reasons:

  – If the system admin wants the container header to be checked/validated, the secure boot jumper must be set. Otherwise, the container header information may not be reliable.

  – The container layout is expected to change over time. Skiboot would need to maintain a parser for each container layout change.

  – Skiboot could be checking the hash against a container version that is not supported by the Container-Verification-Code (CVC).

    The tb_measure() calls are updated to trustedboot_measure() in a subsequent patch.

- libstb/secureboot.c: import sb_verify() from stb.c

  This imports the sb_verify() function from stb.c, but now it calls the CVC verify wrapper in order to verify signed firmware images. The hw-key-hash and hw-key-hash-size initialized in secureboot.c are passed to the CVC verify function wrapper.

  In secureboot.c, the sb_verify() is renamed to secureboot_verify(). The sb_verify() calls are updated in a subsequent patch.

XIVE

- xive: Don't bother cleaning up disabled EQs in reset

  Additionally, warn if we find an enabled one that isn't one of the firmware built-in queues.

- xive: Warn on valid VPs found in abnormal cases

  If an allocated VP is left valid at xive_reset() or Linux tries to free a valid (enabled) VP block, print errors. The former happens occasionally if kdump'ing while KVM is running so keep it as a debug message. The latter is a programming error in Linux so use a an error log level.

- xive: Properly reserve built-in VPs in non-group mode

  This is not normally used but if the #define is changed to disable block group mode we would incorrectly clear the buddy completely without marking the built-in VPs reserved.

- xive: Quieten debug messages in standard builds

  This makes a bunch of messages, especially the per-CPU ones, only enabled in debug builds. This avoids clogging up the OPAL logs with XIVE related messages that have proven not being particularly useful for field defects.

- xive: Implement "single escalation" feature

  This adds a new VP flag to control the new DD2.0 "single escalation" feature.

  This feature allows us to have a single escalation interrupt per VP instead of one per queue.

It works by hijacking queue 7 (which is this no longer usable when that is enabled) and exploiting two new hardware bits that will:

- Make the normal queues (0..6) escalate unconditionally thus ignoring the ESe bits.
- Route the above escalations to queue 7
- Have queue 7 silently escalate without notification

Thus the escalation of queue 7 becomes the one escalation interrupt for all the other queues.

- xive: When disabling a VP, wipe all of its settings
- xive: Improve cleaning up of EQs

  Factors out the function that sets an EQ back to a clean state and add a cleaning pass for queue left enabled when freeing a block of VPs.
- xive: When disabling an EQ, wipe all of its settings

  This avoids having configuration bits left over
- xive: Define API for single-escalation VP mode

  This mode allows all queues of a VP to use the same escalation interrupt, at the cost of losing priority 7.

  This adds the definition and documentation of the API, the implementation will come next.
- xive: Fix ability to clear some EQ flags

  We could never clear "unconditional notify" and "escalate"
- xive: Update inits for DD2.0

  This updates some inits based on information from the HW designers. This includes enabling some new DD2.0 features that we don't yet exploit.
- xive: Ensure VC informational FIRs are masked

  Some HostBoot versions leave those as checkstop, they are harmless and can sometimes occur during normal operations.
- xive: Fix occasional VC checkstops in xive_reset

  The current workaround for the scrub bug described in __xive_cache_scrub() has an issue in that it can leave dirty invalid entries in the cache.

  When cleaning up EQs or VPs during reset, if we then remove the underlying indirect page for these entries, the XIVE will checkstop when trying to flush them out of the cache.

  This replaces the existing workaround with a new pair of workarounds for VPs and EQs:

  - The VP one does the dummy watch on another entry than the one we scrubbed (which does the job of pushing old stores out) using an entry that is known to be backed by a permanent indirect page.
  - The EQ one switches to a more efficient workaround which consists of doing a non-side-effect ESB load from the EQ's ESe control bits.
- xive: Do not return a trigger page for an escalation interrupt

  This is bogus, we don't support them. (Thankfully the callers didn't actually try to use this on escalation interrupts).
- xive: Mark a freed IRQs IVE as valid and masked

  Removing the valid bit means a FIR will trip if it's accessed inadvertently. Under some circumstances, the XIVE will speculatively access an IVE for a masked interrupt and trip it. So make sure that freed entries are still marked valid (but masked).

PCI

- pci: Shared slot state synchronisation for hot reset

  When a device is shared between two PHBs, it doesn't get reset properly unless both PHBs issue a hot reset at "the same time". Practically this means a hot reset needs to be issued on both sides, and neither should bring the link up until the reset on both has completed.

- pci: Track peers of slots

  Witherspoon introduced a new concept where one physical slot is shared between two PHBs. Making a slot aware of its peer enables syncing between them where necessary.

PHB4

- phb4: Change PCI MMIO timers

  Currently we have a mismatch between the NCU and PCI timers for MMIO accesses. The PCI timers must be lower than the NCU timers otherwise it may cause checkstops.

  This changes PCI timeouts controlled by skiboot to 33-50ms. It should be forwards and backwards compatible with expected hostboot changes to the NCU timer.

- phb4: Change default GEN3 lane equalisation setting to 0x54

  Currently our GEN3 lane equalisation settings are set to 0x77. Change this to 0x54. This change will allow us to train at GEN3 in a shorter time and more consistently.

  This setting gives us a TX preset 0x4 and RX hint 0x5. This gives a boost in gain for high frequency signalling. It allows the most optimal continuous time linear equalizers (CTLE) for the remote receiver port and de-emphasis and pre-shoot for the remote transmitter port.

  Machine Readable Workbooks (MRW) are moving to this new value also.

- phb4: Init changes

  These init changes for phb4 from the HW team.

  Link down are now endpoint recoverable (ERC) rather than PHB fatal errors.

  BLIF Completion Timeout Error now generate an interrupt rather than causing freeze events.

- phb4: Fix lane equalisation setting

  Fix cut and paste from phb3. The sizes have changes now we have GEN4, so the check here needs to change also

  Without this we end up with the default settings (all '7') rather than what's in HDAT.

- hdata: Fix copying GEN4 lane equalisation settings

  These aren't copied currently but should be.

- phb4: Fix PE mapping of M32 BAR

  The M32 BAR is the PHB4 region used to map all the non-prefetchable or 32-bit device BARs. It's supposed to have its segments remapped via the MDT and Linux relies on that to assign them individual PE#.

  However, we weren't configuring that properly and instead used the mode where PE# == segment#, thus causing EEH to freeze the wrong device or PE#.

- phb4: Fix lost bit in PE number on config accesses

  A PE number can be up to 9 bits, using a uint8_t won't fly..

That was causing error on config accesses to freeze the wrong PE.

- phb4: Update inits

  New init value from HW folks for the fence enable register.

  This clears bit 17 (CFG Write Error CA or UR response) and bit 22 (MMIO Write DAT_ERR Indication) and sets bit 21 (MMIO CFG Pending Error)

CAPI

- capi: Disable CAPP virtual machines

  When exercising more than one CAPI accelerators simultaneously in cache coherency mode, the verification team is seeing a deadlock. To fix this a workaround of disabling CAPP virtual machines is suggested. These 'virtual machines' let PSL queue multiple CAPP commands for servicing by CAPP there by increasing throughput. Below is the error scenario described by the h/w team:

  " With virtual machines enabled we had a deadlock scenario where with 2 or more CAPI's in a system you could get in a deadlock scenario due to cast-outs that are required break the deadlock (evict lines that another CAPI is requesting) get stuck in the virtual machine queue by a command ahead of it that is being retried by the same scenario in the other CAPI. "

- capi: Perform capp recovery sequence only when PBCQ is idle

  Presently during a CRESET the CAPP recovery sequence can be executed multiple times in case PBCQ on the PEC is still busy processing in/out bound in-flight transactions.

- xive: Mask MMIO load/store to bad location FIR

  For opencapi, the trigger page of an interrupt is mapped to user space. The intent is to write the page to raise an interrupt but there's nothing to prevent a user process from reading it, which has the unfortunate consequence of checkstopping the system.

  Mask the FIR bit raised when an MMIO operation targets an invalid location. It's the recommendation from recent documentation and hostboot is expected to mask it at some point. In the meantime, let's play it safe.

- phb4: Dump CAPP error registers when it asserts link down

  This patch introduces a new function phb4_dump_app_err_regs() that dumps CAPP error registers in case the PEC nestfir register indicates that the fence was due to a CAPP error (BIT-24).

  Contents of these registers are helpful in diagnosing CAPP issues. Registers that are dumped in phb4_dump_app_err_regs() are:

  - CAPP FIR Register
  - CAPP APC Master Error Report Register
  - CAPP Snoop Error Report Register
  - CAPP Transport Error Report Register
  - CAPP TLBI Error Report Register
  - CAPP Error Status and Control Register

- capi: move the acknowledge of the HMI interrupt

  We need to acknowledge an eventual HMI initiated by the previous forced fence on the PHB to work around a non-existent PE in the phb4_creset() function. For this reason do_capp_recovery_scoms() is called now at the beginning of the step: PHB4_SLOT_CRESET_WAIT_CQ

- capi: update ci store buffers and dma engines

  The number of read (APC type traffic) and mmio store (MSG type traffic) resources assigned to the CAPP is controlled by the CAPP control register.

  According to the type of CAPI cards present on the server, we have to configure differently the CAPP messages and the DMA read engines given to the CAPP for use.

HMI

- core/hmi: Display chip location code while displaying core FIR.
- core/hmi: Do not display FIR details if none of the bits are set.

  So that we don't flood OPAL console logs with information that is not useful.
- opal/hmi: HMI logging with location code info.

  Add few HMI debug prints with location code info few additional info.

  No functionality change.

  With this patch the log messages will look like:

  ```
  [210612.175196744,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
  [210612.175200449,7] HMI: [Loc: UOPWR.1302LFA-Node0-Proc1]: P:8 C:16 T:1:␣
  ↪TFMR(2d12000870e04020) Timer Facility Error

  [210660.259689526,7] HMI: Received HMI interrupt: HMER = 0x2040000000000000
  [210660.259695649,7] HMI: [Loc: UOPWR.1302LFA-Node0-Proc0]: P:0 C:16 T:1:␣
  ↪Processor recovery Done.
  ```

- core/hmi: Use pr_fmt macro for tagging log messages

  No functionality changes.
- opal: Get chip location code

  and store it under proc_chip for quick reference during HMI handling code.

Sensors

- occ-sensors: Fix up quad/gpu location mix-up

  The GPU and QUAD sensor location types are swapped compared to what exists in the OCC code base which is authoritative. Fix them up.
- sensors: occ: Skip counter type of sensors

  Don't add counter type of sensors to device-tree as they don't fit into hwmon sensor interface.
- sensors: dts: Assert special wakeup on idle cores while reading temperature

  In P9, when a core enters a stop state, its clocks will be stopped to save power and hence we will not be able to perform a SCOM operation to read the DTS temperature sensor. Hence, assert a special wakeup on cores that have entered a stop state in order to successfully complete the SCOM operation.
- sensors: occ: Skip power sensors with zero sample value

  APSS is not available on platforms like Zaius, Romulus where OCC can only measure Vdd (core) and Vdn (nest) power from the AVSbus reading. So all the sensors for APSS channels will be populated with 0. Different component power sensors like system, memory which point to the APSS channels will also be 0.

As per OCC team (Martha Broyles) zeroed power sensor means that the system doesn't have it. So this patch filters out these sensors.

- sensors: occ: Skip GPU sensors for non-gpu systems
- sensors: Fix dtc warning for new occ in-band sensors.

  dtc complains about missing reg property when a DT node is having a unit name or address but no reg property.

  ```
  /ibm,opal/sensors/vrm-in@c00004 has a unit name, but no reg property
  /ibm,opal/sensors/gpu-in@c0001f has a unit name, but no reg property
  /ibm,opal/sensor-groups/occ-js@1c00040 has a unit name, but no reg property
  ```

  This patch fixes these warnings for new occ in-band sensors and also for sensor-groups by adding necessary properties.

- sensors: Fix dtc warning for dts sensors.

  dtc complains about missing reg property when a DT node is having a unit name or address but no reg property.

  Example warning for core dts sensor:

  ```
  /ibm,opal/sensors/core-temp@5c has a unit name, but no reg property
  /ibm,opal/sensors/core-temp@804 has a unit name, but no reg property
  ```

  This patch fixes this by adding necessary properties.

- hw/occ: Fix psr cpu-to-gpu sensors node dtc warning.

  dtc complains about missing reg property when a DT node is having a unit name or address but no reg property.

  ```
  /ibm,opal/power-mgt/psr/cpu-to-gpu@0 has a unit name, but no reg property
  /ibm,opal/power-mgt/psr/cpu-to-gpu@100 has a unit name, but no reg property
  ```

  This patch fixes this by adding necessary properties.

General fixes

- lpc: Clear pending IRQs at boot

  When we come in from hostboot the LPC master has the bus reset indicator set. This error isn't handled until the host kernel unmasks interrupts, at which point we get the following spurious error:

  ```
  [   20.053560375,3] LPC: Got LPC reset on chip 0x0 !
  [   20.053564560,3] LPC[000]: Unknown LPC error Error address reg: 0x00000000
  ```

  Fix this by clearing the various error bits in the LPC status register before we initialise the skiboot LPC bus driver.

- hw/imc: Check ucode state before exposing units to Linux

  disable_unavailable_units() checks whether the ucode is in the running state before enabling the nest units in the device tree. From a recent debug, it is found that on some system boot, ucode is not loaded and running in all the chips in the system. And this caused a fail in OPAL_IMC_COUNTERS_STOP call where we check for ucode state on each chip. Bug here is that disable_unavailable_units() checks the state of the ucode only in boot cpu chip. Patch adds a condition in disable_unavailable_units() to check for the ucode state in all the chip before enabling the nest units in the device tree node.

- hdata/vpd: Add vendor property

  ibm,vpd blob contains VN field. Use that to populate vendor property for various FRU's.

- hdata/vpd: Fix DTC warnings

  All the nodes under the vpd hierarchy have a unit address (their SLCA index) but no reg properties. Add them and their size/address cells to squash the warnings.

- HDAT/i2c: Fix SPD EEPROM compatible string

  Hostboot doesn't give us accurate information about the DIMM SPD devices. Hack around by assuming any EEPROM we find on the SPD I2C master is an SPD EEPROM.

- hdata/i2c: Fix 512Kb EEPROM size

  There's no such thing as a 412Kb EEPROM.

- libflash/mbox-flash: fall back to requesting lower MBOX versions from BMC

  Some BMC mbox implementations seem to sometimes mysteriously fail when trying to negotiate v3 when they only support v2. To work around this, we can fall back to requesting lower mbox protocol versions until we find one that works.

  In theory, this should already "just work", but we have a counter example, which this patch fixes.

- IPMI: Fix platform.cec_reboot() null ptr checks

  Kudos to Hugo Landau who reported this in: https://github.com/open-power/skiboot/issues/142

- hdata: Add location code property to xscom node

  This patch adds chip location code property to xscom node.

- p8-i2c: Limit number of retry attempts

  Current we will attempt to start an I2C transaction until it succeeds. In the event that the OCC does not release the lock on an I2C bus this results in an async token being held forever and the kernel thread that started the transaction will block forever while waiting for an async completion message. Fix this by limiting the number of attempts to start the transaction.

- p8-i2c: Don't write the watermark register at init

  On P9 the I2C master is shared with the OCC. Currently the watermark values are set once at init time which is bad for two reasons:

  a) We don't take the OCC master lock before setting it. Which may cause issues if the OCC is currently using the master.

  b) The OCC might change the watermark levels and we need to reset them.

  Change this so that we set the watermark value when a new transaction is started rather than at init time.

- hdata: Rename 'fsp-ipl-side' as 'sp-ipl-side'

  as OPAL is building device tree for both FSP and BMC system. Also I don't see anyone using this property today. Hence renaming should be fine.

- hdata/vpd: add support for parsing CPU VRML records

  Allows skiboot to parse out the processor part/serial numbers on OpenPOWER P9 machines.

- core/lock: Introduce atomic cmpxchg and implement try_lock with it

  cmpxchg will be used in a subsequent change, and this reduces the amount of asm code.

- direct-controls: add xscom error handling for p8

  Add xscom checks which will print something useful and return error back to callers (which already have error handling plumbed in).

- direct-controls: p8 implementation of generic direct controls

  This reworks the sreset functionality that was brought over from fast-reboot, and fits it under the generic direct controls APIs.

  The fast reboot APIs are implemented using generic direct controls, which also makes them available on p9.

- fast-reboot: allow mambo fast reboot independent of CPU type

  Don't tie mambo fast reboot to POWER8 CPU type.

- fast-reboot: remove delay after sreset

  There is a 100ms delay when targets reach sreset which does not appear to have a good purpose. Remove it and therefore reduce the sreset timeout by the same amount.

- fast-reboot: add more barriers around cpu state changes

  This is a bit of paranoia, but when a CPU changes state to signal it has reached a particular point, all previous stores should be visible.

- fast-reboot: add sreset timeout detection and handling

  Have the initiator wait for all its sreset targets to call in, and time out after 200ms if they did not. Fail and revert to IPL reboot.

  Testing indicates that after successful sreset_all_others(), it takes less than 102ms (in hundreds of fast reboots) for secondaries to call in. 100 of that is due to an initial delay, but core un-splitting was not measured.

- fast-reboot: make spin loops consistent and SMT friendly
- fast-reboot: add sreset_all_others error handling

  Pass back failures from sreset_all_others, also change return codes to OPAL form in sreset_all_prepare to match.

  Errors will revert to the IPL path, so it's not critical to completely clean up everything if that would complicate things. Detecting the error and failing is the important thing.

- fast-reboot: restore SMT priority on spin loop exit
- Add documentation for ibm, firmware-versions device tree node
- NX: Print read xscom config failures.

  Currently in NX, only write xscom config failures are tracing. Add trace statements for read xscom config failures too. No functional changes.

- hw/nx: Fix NX BAR assignments

  The NX rng BAR is used by each core to source random numbers for the DARN instruction. Currently we configure each core to use the NX rng of the chip that it exists on. Unfortunately, the NX can be de-configured by hostboot and in this case we need to use the NX of a different chip.

  This patch moves the BAR assignments for the NX into the normal nx-rng init path. This lets us check if the normal (chip local) NX is active when configuring which NX a core should use so that we can fall back gracefully.

- FSP-elog: Reduce verbosity of elog messages

  These messages just fill up the opal console log with useless messages resulting in us losing useful information.

  They have been like this since the first commit in skiboot. Make them trace.

- core/bitmap: fix bitmap iteration limit corruption

  The bitmap iterators did not reduce the number of bits to scan when searching for the next bit, which would result in them overrunning their bitmap.

These are only used in one place, in xive reset, and the effect is that the xive reset code will keep zeroing memory until it reaches a block of memory of MAX_EQ_COUNT >> 3 bits in length, all zeroes.

- hw/imc: always enable "imc_nest_chip" exports property

  imc_dt_update_nest_node() adds a "imc_nest_chip" property to the "exports" node (under opal_node) to view nest counter region. This comes handy when debugging ucode runtime errors (like counter data update or control block update so on…). And current code enables the property only if the microcode is in running state at system boot. To aid the debug of ucode not running/starting issues at boot, enable the addition of "imc_nest_chip" property always.

NVLINK2

- npu2-hw-procedures.c: Correct phy lane mapping

  Each NVLINK2 device is associated with a particular group of OBUS lanes via a lane mask which is read from HDAT via the device-tree. However Skiboot's interpretation of lane mask was different to what is exported from the HDAT.

  Specifically the lane mask bits in the HDAT are encoded in IBM bit ordering for a 24-bit wide value. So for example in normal bit ordering lane-0 is represented by having lane-mask bit 23 set and lane-23 is represented by lane-mask bit 0. This patch alters the Skiboot interpretation to match what is passed from HDAT.

- npu2-hw-procedures.c: Power up lanes during ntl reset

  Newer versions of Hostboot will not power up the NVLINK2 PHY lanes by default. The phy_reset procedure already powers up the lanes but they also need to be powered up in order to access the DL.

  The reset_ntl procedure is called by the device driver to bring the DL out of reset and get it into a working state. Therefore we also need to add lane and clock power up to the reset_ntl procedure.

- npu2.c: Add PE error detection

  Invalid accesses from the GPU can cause a specific PE to be frozen by the NPU. Add an interrupt handler which reports the frozen PE to the operating system via as an EEH event.

- npu2.c: Fix XIVE IRQ alignment
- npu2: hw-procedures: Refactor reset_ntl procedure

  Change the implementation of reset_ntl to match the latest programming guide documentation.

- npu2: hw-procedures: Add phy_rx_clock_sel()

  Change the RX clk mux control to be done by software instead of HW. This avoids glitches caused by changing the mux setting.

- npu2: hw-procedures: Change phy_rx_clock_sel values

  The clock selection bits we set here are inputs to a state machine.

  DL clock select (bits 30-31)

  0b00 lane 0 clock

  0b01 lane 7 clock

  0b10 grid clock

  0b11 invalid/no-op

  To recover from a potential glitch, we need to ensure that the value we set forces a state change. Our current sequence is to set 0x3 followed by 0x1. With the above now known, that is actually a no-op followed by selection of lane 7. Depending on lane reversal, that selection is not a state change for some bricks.

The way to force a state change in all cases is to switch to the grid clock, and then back to a lane.

- npu2: hw-procedures: Manipulate IOVALID during training

  Ensure that the IOVALID bit for this brick is raised at the start of link training, in the reset_ntl procedure.

  Then, to protect us from a glitch when the PHY clock turns off or gets chopped, lower IOVALID for the duration of the phy_reset and phy_rx_dccal procedures.

- npu2: hw-procedures: Add check_credits procedure

  As an immediate mitigation for a current hardware glitch, add a procedure that can be used to validate NTL credit values. This will be called as a safeguard to check that link training succeeded.

  Assert that things are exactly as we expect, because if they aren't, the system will experience a catastrophic failure shortly after the start of link traffic.

- npu2: Print bdfn in NPU2DEV* logging macros

  Revise the NPU2DEV{DBG,INF,ERR} logging macros to include the device's bdfn. It's useful to know exactly which link we're referring to.

  For instance, instead of

  ```
  [  234.044921238,6] NPU6: Starting procedure reset_ntl
  [  234.048578101,6] NPU6: Starting procedure reset_ntl
  [  234.051049676,6] NPU6: Starting procedure reset_ntl
  [  234.053503542,6] NPU6: Starting procedure reset_ntl
  [  234.057182864,6] NPU6: Starting procedure reset_ntl
  [  234.059666137,6] NPU6: Starting procedure reset_ntl
  ```

  we'll get

  ```
  [  234.044921238,6] NPU6:0:0.0 Starting procedure reset_ntl
  [  234.048578101,6] NPU6:0:0.1 Starting procedure reset_ntl
  [  234.051049676,6] NPU6:0:0.2 Starting procedure reset_ntl
  [  234.053503542,6] NPU6:0:1.0 Starting procedure reset_ntl
  [  234.057182864,6] NPU6:0:1.1 Starting procedure reset_ntl
  [  234.059666137,6] NPU6:0:1.2 Starting procedure reset_ntl
  ```

- npu2: Move to new GPU memory map

  There are three different ways we configure the MCD and memory map.

  1) **Old way (current way)** Skiboot configures the MCD and puts GPUs at 4TB and below
  2) **New way with MCD** Hostboot configures the MCD and skiboot puts GPU at 4TB and above
  3) **New way without MCD** No one configures the MCD and skiboot puts GPU at 4TB and below

  The patch keeps option 1 and adds options 2 and 3.

  The different configurations are detected using certain scoms (see patch).

  Option 1 will go away eventually as it's a configuration that can cause xstops or data integrity problems. We are keeping it around to support existing hostboot.

  Option 2 supports only 4 GPUs and 512GB of memory per socket.

  Option 3 supports 6 GPUs and 4TB of memory but may have some performance impact.

- phys-map: Rename GPU_MEM to GPU_MEM_4T_DOWN

  This map is soon to be replaced, but we are going to keep it around for a little while so that we support older hostboot firmware.

Platform Specific Fixes

Witherspoon

- Witherspoon: Remove old Witherspoon platform definition

  An old Witherspoon platform definition was added to aid the transition from versions of Hostboot which didn't have the correct NVLINK2 HDAT information available and/or planar VPD. These system should now be updated so remove the possibly incorrect default assumption.

  This may disable NVLINK2 on old out-dated systems but it can easily be restored with the appropriate FW and/or VPD updates. In any case there is a a 50% chance the existing default behaviour was incorrect as it only supports 6 GPU systems. Using an incorrect platform definition leads to undefined behaviour which is more difficult to detect/debug than not creating the NVLINK2 devices so remove the possibly incorrect default behaviour.

- Witherspoon: Fix VPD EEPROM type

  There are user-space tools that update the planar VPD via the sysfs interface. Currently we do not get correct information from hostboot about the exact type of the EEPROM so we need to manually fix it up here. This needs to be done as a platform specific fix since there is not standardised VPD EEPROM type.

IBM FSP Systems

- nvram: Fix 'missing' nvram on FSP systems.

  commit ba4d46fdd9eb ("console: Set log level from nvram") wants to read from NVRAM rather early. This works fine on BMC based systems as nvram_init() is actually synchronous. This is not true for FSP systems and it turns out that the query for the console log level simply queries blank nvram.

  The simple fix is to wait for the NVRAM read to complete before performing any query. Unfortunately it turns out that the fsp-nvram code does not inform the generic NVRAM layer when the read is complete, rather, it must be prompted to do so.

  This patch addresses both these problems. This patch adds a check before the first read of the NVRAM (for the console log level) that the read has completed. The fsp-nvram code has been updated to inform the generic layer as soon as the read completes.

  The old prompt to the fsp-nvram code has been removed but a check to ensure that the NVRAM has been loaded remains. It is conservative but if the NVRAM is not done loading before the host is booted it will not have an nvram device-tree node which means it won't be able to access the NVRAM at all, ever, even after the NVRAM has loaded.

Utilities

- Fix xscom-utils distclean target

  In Debian/Ubuntu, the packaging system likes to have a full clean-up that restores the tree back to original one, so add some files to the distclean target.

- Add man pages for xscom-utils and pflash

  For the need of Debian/Ubuntu packaging, I inferred some initial man pages from their help output.

gard

- gard: Add tests

  I hear Stewart likes these for some reason. Dunno why.

- gard: Add OpenBMC vPNOR support

  A big-ol-hack to add some checking for OpenBMC's vPNOR GUARD files under /media/pnor-prsv. This isn't ideal since it doesn't handle the create case well, but it's better than nothing.

- gard: Always use MTD to access flash

  Direct mode is generally either unsafe or unsupported. We should always access the PNOR via an MTD device so make that the default. If someone really needs direct mode, then they can use pflash.

- gard: Fix up do_create return values

  The return value of a subcommand is interpreted as a libflash error code when it's positive or some subcommand specific error when negative. Currently the create subcommand always returns zero when exiting (even for errors) so fix that.

- gard: Add usage message for -p

  The -p argument only really makes sense when -f is specified. Print an actual error message rather than just the usage blob.

- gard: Fix max instance count

  There's an entire byte for the instance count rather than a nibble. Only barf if the instance number is beyond 255 rather than 16.

- gard: Fix up path parsing

  Currently we assume that the Unit ID can be used as an array index into the chip_units[] structure. There are holes in the ID space though, so this doesn't actually work. Fix it up by walking the array looking for the ID.

- gard: Set chip generation based on PVR

  Currently we assume that this tool is being used on a P8 system by default and allow the user to override this behaviour using the -8 and -9 command line arguments. When running on the host we can use the PVR to guess what chip generation so do that.

  This also changes the default behaviour to assume that the host is a P9 when running on an ARM system. This tool didn't even work when compiled for ARM until recently and the OpenBMC vPNOR hack that we have currently is broken for P9 systems that don't use vPNOR (Zaius and Romulus).

- gard: Allow records with an ID of 0xffffffff

  We currently assume that a record with an ID of 0xffffffff is invalid. Apparently this is incorrect and we should display these records, so expand the check to compare the entire record with 0xff rather than just the ID.

- gard: create: Allow creating arbitrary GARD records

  Add a new sub-command that allows us to create GARD records for arbitrary chip units. There isn't a whole lot of constraints on this and that limits how useful it can be, but it does allow a user to GARD out individual DIMMs, chips or cores from the BMC (or host) if needed.

  There are a few caveats though:

  1) Not everything can, or should, have a GARD record applied it to.

  2) There is no validation that the unit actually exists. Doing that sort of validation requires something that understands the FAPI targeting information (I think) and adding support for it here would require some knowledge from the system XML file.

3) There's no way to get a list of paths in the system.
4) Although we can create a GARD record at runtime it won't be applied until the next IPL.

- gard: Add path parsing support

  In order to support manual GARD records we need to be able to parse the hardware unit path strings. This patch implements that.

- gard: list: Improve output

  Display the full path to the GARDed hardware unit in each record rather than relying on the output of *gard show* and convert do_list() to use the iterator while we're here.

- gard: {list, show}: Fix the Type field in the output

  The output of *gard list* has a field named "Type", however this doesn't actually indicate the type of the record. Rather, it shows the type of the path used to identify the hardware being GARDed. This is of pretty dubious value considering the Physical path seems to always be used when referring to GARDed hardware.

- gard: Add P9 support
- gard: Update chip unit data

  Source the list of units from the hostboot source rather than the previous hard coded list. The list of path element types changes between generations so we need to add a level of indirection to accommodate P9. This also changes the names used to match those printed by Hostboot at IPL time and paves the way to adding support for manual GARD record creation.

- gard: show: Remove "Res Recovery" field

  This field has never been populated by hostboot on OpenPower systems so there's no real point in reporting it's contents.

libflash / pflash

Anybody shipping libflash or pflash to interact with POWER9 systems must upgrade to this version.

- pflash: Support for volatile flag

  The volatile flag was added to the PNOR image to indicate partitions that are cleared during a host power off. Display this flag from the pflash command.

- pflash: Support for clean_on_ecc_error flag

  Add the misc flag clear_on_ecc_error to libflash/pflash. This was the only missing flag. The generator of the virtual PNOR image relies on libflash/pflash to provide the partition information, so all flags are needed to build an accurate virtual PNOR partition table.

- pflash: Respect write(2) return values

  The write(2) system call returns the number of bytes written, this is important since it is entitled to write less than what we requested. Currently we ignore the return value and assume it wrote everything we requested. While in practice this is likely to always be the case, it isn't actually correct.

- external/pflash: Fix erasing within a single erase block

  It is possible to erase within a single erase block. Currently the pflash code assumes that if the erase starts part way into an erase block it is because it needs to be aligned up to the boundary with the next erase block.

  Doing an erase smaller than a single erase block will cause underflows and looping forever on erase.

- external/pflash: Fix non-zero return code for successful read when size%256 != 0

  When performing a read the return value from pflash is non-zero, even for a successful read, when the size being read is not a multiple of 256. This is because do_read_file returns the value from the write system call which is then returned by pflash. When the size is a multiple of 256 we get lucky in that this wraps around back to zero. However for any other value the return code is size % 256. This means even when the operation is successful the return code will seem to reflect an error.

  Fix this by returning zero if the entire size was read correctly, otherwise return the corresponding error code.

- libflash: Fix parity calculation on ARM

  To calculate the ECC syndrome we need to calculate the parity of a 64bit number. On non-powerpc platforms we use the GCC builtin function __builtin_parityl() to do this calculation. This is broken on 32bit ARM where sizeof(unsigned long) is four bytes. Using __builtin_parityll() instead cures this.

- libflash/mbox-flash: Add the ability to lock flash
- libflash/mbox-flash: Understand v3
- libflash/mbox-flash: Use BMC suggested timeout value
- libflash/mbox-flash: Simplify message sending

  hw/lpc-mbox no longer requires that the memory associated with messages exist for the lifetime of the message. Once it has been sent to the BMC, that is bmc_mbox_enqueue() returns, lpc-mbox does not need the message to continue to exist. On the receiving side, lpc-mbox will ensure that a message exists for the receiving callback function.

  Remove all code to deal with allocating messages.

- hw/lpc-mbox: Simplify message bookkeeping and timeouts

  Currently the hw/lpc-mbox layer keeps a pointer for the currently in-flight message for the duration of the mbox call. This creates problems when messages timeout, is that pointer still valid, what can we do with it. The memory is owned by the caller but if the caller has declared a timeout, it may have freed that memory.

  Another problem is locking. This patch also locks around sending and receiving to avoid races with timeouts and possible resends. There was some locking previously which was likely insufficient - definitely too hard to be sure is correct

  All this is made much easier with the previous rework which moves sequence number allocation and verification into lpc-mbox rather than the caller.

- libflash/mbox-flash: Allow mbox-flash to tell the driver msg timeouts

  Currently when mbox-flash decides that a message times out the driver has no way of knowing to drop the message and will continue waiting for a response indefinitely preventing more messages from ever being sent.

  This is a problem if the BMC crashes or has some other issue where it won't ever respond to our outstanding message.

  This patch provides a method for mbox-flash to tell the driver how long it should wait before it no longer needs to care about the response.

- libflash/mbox-flash: Move sequence handling to driver level
- libflash/mbox-flash: Always close windows before opening a new window

  The MBOX protocol states that if an open window command fails then all open windows are closed. Currently, if an open window command fails mbox-flash will erroneously assume that the previously open window is still open.

  The solution to this is to mark all windows as closed before issuing an open window command and then on success we'll mark the new window as open.

- libflash/mbox-flash: Add v2 error codes

opal-prd

Anybody shipping *opal-prd* for POWER9 systems must upgrade *opal-prd* to this new version.

- prd: Log unsupported message type

  Useful for debugging.

  Sample output:

  ```
  [29155.157050283,7] PRD: Unsupported prd message type : 0xc
  ```

- opal-prd: occ: Add support for runtime OCC load/start in ZZ

  This patch adds support to handle OCC load/start event from FSP/PRD. During IPL we send a success directly to FSP without invoking any HBRT load routines on receiving OCC load mbox message from FSP. At runtime we forward this event to host opal-prd.

  This patch provides support for invoking OCC load/start HBRT routines like load_pm_complex() and start_pm_complex() from opal-prd.

- opal-prd: Add support for runtime OCC reset in ZZ

  This patch handles OCC_RESET runtime events in host opal-prd and also provides support for calling 'hostinterface->wakeup()' which is required for doing the reset operation.

- prd: Enable error logging via firmware_request interface

  In P9 HBRT sends error logs to FSP via firmware_request interface. This patch adds support to parse error log and send it to FSP.

- prd: Add generic response structure inside prd_fw_msg

  This patch adds generic response structure. Also sync prd_fw_msg type macros with hostboot.

- opal-prd: flush after logging to stdio in debug mode

  When in debug mode, flush after each log output. This makes it more likely that we'll catch failure reasons on severe errors.

Debugging and reliability improvements

- lock: Add additional lock auditing code

  Keep track of lock owner name and replace lock_depth counter with a per-cpu list of locks held by the cpu.

  This allows us to print the actual locks held in case we hit the (in)famous message about opal_pollers being run with a lock held.

  It also allows us to warn (and drop them) if locks are still held when returning to the OS or completing a scheduled job.

- Add support for new GCC 7 parametrized stack protector

  This gives us per-cpu guard values as well. For now I just XOR a magic constant with the CPU PIR value.

- Mambo: run hello_world and sreset_world tests with Secure and Trusted Boot

  We *disable* the secure boot part, but we keep the verified boot part as we don't currently have container verification code for Mambo.

We can run a small part of the code currently though.

- core/flash.c: extern function to get the name of a PNOR partition

  This adds the flash_map_resource_name() to allow skiboot subsystems to lookup the name of a PNOR partition. Thus, we don't need to duplicate the same information in other places (e.g. libstb).

- libflash/mbox-flash: only wait for MBOX_DEFAULT_POLL_MS if busy

  This makes the mbox unit test run 300x quicker and seems to shave about 6 seconds from boot time on Witherspoon.

- make check: Make valgrind optional

  To (slightly) lower the barrier for contributions, we can make valgrind optional with just a small amount of plumbing.

  This allows make check to run successfully without valgrind.

- libflash/test: Add tests for mbox-flash

  A first basic set of tests for mbox-flash. These tests do their testing by stubbing out or otherwise replacing functions not in libflash/mbox-flash.c. The stubbed out version of the function can then be used to emulate a BMC mbox daemon talking to back to the code in mbox-flash and it can ensure that there is some adherence to the protocol and that from a block-level api point of view the world appears sane.

  This makes these tests simple to run and they have been integrated into *make check*. The down side is that these tests rely on duplicated feature incomplete BMC daemon behaviour. Therefore these tests are a strong indicator of broken behaviour but a very unreliable indicator of correctness.

  Full integration tests with a 'real' BMC daemon are probably beyond the scope of this repository.

- external/test/test.sh: fix VERSION substitution when no tags

  i.e. we get a hash rather than a version number

  This seems to be occurring in Travis if it doesn't pull a tag.

- external/test: make stripping out version number more robust

  For some bizarre reason, Travis started failing on this substitution when there'd been zero code changes in this area… This at least papers over whatever the problem is for the time being.

- io: Add load_wait() helper

  This uses the standard form twi/isync pair to ensure a load is consumed by the core before continuing. This can be necessary under some circumstances for example when having the following sequence:

  – Store reg A

  – Load reg A (ensure above store pushed out)

  – delay loop

  – Store reg A

  I.E., a mandatory delay between 2 stores. In theory the first store is only guaranteed to reach the device after the load from the same location has completed. However the processor will start executing the delay loop without waiting for the return value from the load.

  This construct enforces that the delay loop isn't executed until the load value has been returned.

- chiptod: Keep boot timestamps contiguous

  Currently we reset the timebase value to (almost) zero when synchronising the timebase of each chip to the Chip TOD network which results in this:

```
[   42.374813167,5] CPU: All 80 processors called in...
[    2.222791151,5] FLASH: Found system flash: Macronix MXxxL51235F id:0
[    2.222977933,5] BT: Interface initialized, IO 0x00e4
```

This patch modifies the chiptod_init() process to use the current timebase value rather than resetting it to zero. This results in the timestamps remaining contiguous from the start of hostboot until the petikernel starts. e.g.

```
[   70.188811484,5] CPU: All 144 processors called in...
[   72.458004252,5] FLASH: Found system flash:  id:0
[   72.458147358,5] BT: Interface initialized, IO 0x00e4
```

- hdata/spira: Add missing newline to prlog() call

  We're missing a n here.

- opal/xscom: Add recovery for lost core wakeup SCOM failures.

  Due to a hardware issue where core responding to SCOM was delayed due to thread reconfiguration, leaves the SCOM logic in a state where the subsequent SCOM to that core can get errors. This is affected for Core PC SCOM registers in the range of 20010A80-20010ABF

  The solution is if a xscom timeout occurs to one of Core PC SCOM registers in the range of 20010A80-20010ABF, a clearing SCOM write is done to 0x20010800 with data of '0x00000000' which will also get a timeout but clears the SCOM logic errors. After the clearing write is done the original SCOM operation can be retried.

  The SCOM timeout is reported as status 0x4 (Invalid address) in HMER[21-23].

- opal/xscom: Move the delay inside xscom_reset() function.

  So caller of xscom_reset() does not have to bother about adding a delay separately. Instead caller can control whether to add a delay or not using second argument to xscom_reset().

- timer: Stop calling list_top() racily

  This will trip the debug checks in debug builds under some circumstances and is actually a rather bad idea as we might look at a timer that is concurrently being removed and modified, and thus incorrectly assume there is no work to do.

- fsp: Bail out of HIR if FSP is resetting voluntarily

  a. Surveillance response times out and OPAL triggers a HIR
  b. Before the HIR process kicks in, OPAL gets a PSI interrupt indicating link down
  c. HIR process continues and OPAL tries to write to DRCR; PSI link inactive => xstop

  OPAL should confirm that the FSP is not already in reset in the HIR path.

- sreset_kernel: only run SMT tests due to not supporting re-entry
- Use systemsim-p9 v1.1
- direct-controls: enable fast reboot direct controls for mambo

  Add mambo direct controls to stop threads, which is required for reliable fast-reboot. Enable direct controls by default on mambo.

- core/opal: always verify cpu->pir on entry
- asm/head: add entry/exit calls

  Add entry and exit C functions that can do some more complex checks before the opal proper call. This requires saving off volatile registers that have arguments in them.

- core/lock: improve bust_locks

  Prevent try_lock from modifying the lock state when bust_locks is set. unlock will not unlock it in that case, so locks will get taken and never released while bust_locks is set.

- hw/occ: Log proper SCOM register names

  This patch fixes the logging of incorrect SCOM register names.

- mambo: Add support for NUMA

  Currently the mambo scripts can do multiple chips, but only the first ever has memory.

  This patch adds support for having memory on each chip, with each appearing as a separate NUMA node. Each node gets MEM_SIZE worth of memory.

  It's opt-in, via `export MAMBO_NUMA=1`.

- external/mambo: Switch qtrace command to use plug-ins

  The plug-in seems to be the preferred way to do this now, it works better, and the qtracer emitter seems to generate invalid traces in new mambo versions.

- asm/head: Loop after attn

  We use the attn instruction to raise an error in early boot if OPAL don't recognise the PVR. It's possible for hostboot to disable the attn instruction before entering OPAL so add an extra busy loop after the attn to prevent attempting to boot on an unknown processor.

5.1.31 skiboot-5.10-rc2

skiboot v5.10-rc2 was released on Friday February 9th 2018. It is the second release candidate of skiboot 5.10, which will become the new stable release of skiboot following the 5.9 release, first released October 31st 2017.

skiboot v5.10-rc2 contains all bug fixes as of *skiboot-5.9.8* and *skiboot-5.4.9* (the currently maintained stable releases). There may be more 5.9.x stable releases, it will depend on demand.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.10 in February, with skiboot 5.10 being for all POWER8 and POWER9 platforms in op-build v1.21. This release will be targeted to early POWER9 systems.

Over skiboot-5.10-rc1, we have the following changes:

- hw/npu2: Implement logging HMI actions
- opal-prd: Fix FTBFS with -Werror=format-overflow

  i2c.c fails to compile with gcc7 and -Werror=format-overflow used in Debian Unstable and Ubuntu 18.04 :

```
i2c.c: In function 'i2c_init':
i2c.c:211:15: error: '%s' directive writing up to 255 bytes into a
region of size 236 [-Werror=format-overflow=]
```

- core/exception: beautify exception handler, add MCE-involved registers

  Print DSISR and DAR, to help with deciphering machine check exceptions, and improve the output a bit, decode NIP symbol, improve alignment, etc. Also print a specific header for machine check, because we do expect to see these if there is a hardware failure.

  Before:

```
[    0.005968779,3] ***********************************************
[    0.005974102,3] Unexpected exception 200 !
[    0.005978696,3] SRR0 : 000000003002ad80 SRR1 : 9000000000001000
[    0.005985239,3] HSRR0: 00000000300027b4 HSRR1: 9000000030001000
[    0.005991782,3] LR   : 000000003002ad80 CTR  : 0000000000000000
[    0.005998130,3] CFAR : 00000000300b58bc
[    0.006002769,3] CR   : 40000004  XER: 20000000
[    0.006008069,3] GPR00: 000000003002ad80 GPR16: 0000000000000000
[    0.006015170,3] GPR01: 0000000031c03bd0 GPR17: 0000000000000000
[...]
```

After:

```
[    0.003287941,3] ***********************************************
[    0.003561769,3] Fatal MCE at 000000003002ad80   .nvram_init+0x24
[    0.003579628,3] CFAR : 00000000300b5964
[    0.003584268,3] SRR0 : 000000003002ad80 SRR1 : 9000000000001000
[    0.003590812,3] HSRR0: 00000000300027b4 HSRR1: 9000000030001000
[    0.003597355,3] DSISR: 00000000         DAR  : 0000000000000000
[    0.003603480,3] LR   : 000000003002ad68 CTR  : 0000000030093d80
[    0.003609930,3] CR   : 40000004         XER  : 20000000
[    0.003615698,3] GPR00: 00000000300149e8 GPR16: 0000000000000000
[    0.003622799,3] GPR01: 0000000031c03bc0 GPR17: 0000000000000000
[...]
```

- core/init: manage MSR[ME] explicitly, always enable

  The current boot sequence inherits MSR[ME] from the IPL firmware, and never changes it. Some environments disable MSR[ME] (e.g., mambo), and others can enable it (hostboot).

  This has two problems. First, MSR[ME] must be disabled while in the process of taking over the interrupt vector from the previous environment. Second, after installing our machine check handler, MSR[ME] should be enabled to get some useful output rather than a checkstop.

- fast-reboot: occ: Re-parse the pstate table during fast-reboot

  OCC shares the frequency list to host by copying the pstate table to main memory in HOMER. This table is parsed during boot to create device-tree properties for frequency and pstate IDs. OCC can update the pstate table to present a new set of frequencies to the host. But host will remain oblivious to these changes unless it is re-inited with the updated device-tree CPU frequency properties. So this patch allows to re-parse the pstate table and update the device-tree properties during fast-reboot.

  OCC updates the pstate table when asked to do so using pstate-table bias command. And this is mainly used by WOF team for characterization purposes.

- fast-reboot: move pci_reset error handling into fast-reboot code

  pci_reset() currently does a platform reboot if it fails. It should not know about fast-reboot at this level, so instead have it return an error, and the fast reboot caller will do the platform reboot.

  The code essentially does the same thing, but flexibility is improved. Ideally the fast reboot code should perform pci_reset and all such fail-able operations before the CPU resets itself and destroys its own stack. That's not the case now, but that should be the goal.

- capi: Fix the max tlbi divider and the directory size.

  Switch to 512KB mode (directory size) as we don't use bit 48 of the tag in addressing the array. This mode is controlled by the Snoop CAPI Configuration Register. Set the maximum of the number of data polls received before signaling TLBI hang detect timer expired. The value of '0000' is equal to 16.

- npu2/tce: Fix page size checking

The page size is encoded in the TVT data [59:63] as @shift+11 but the tce_kill handler does not do the math right; this fixes it.

- stb: Enforce secure boot if called before libstb initialized
- stb: Correctly error out when no PCR for resource
- core/init: move imc catalog preload init after the STB init.

  As a safer side move the imc catalog preload after the STB init to make sure the imc catalog resource get's verified and measured properly during loading when both secure and trusted boot modes are on.
- libstb: fix failure of calling trusted measure without STB initialization.

  When we load a flash resource during OPAL init, STB calls trusted measure to measure the given resource. There is a situation when a flash gets loaded before STB initialization then trusted measure cannot measure properly.

  So this patch fixes this issue by calling trusted measure only if the corresponding trusted init was done.

  The ideal fix is to make sure STB init done at the first place during init and then do the loading of flash resources, by that way STB can properly verify and measure the all resources.
- libstb: fix failure of calling cvc verify without STB initialization.

  Currently in OPAL init time at various stages we are loading various PNOR partition containers from the flash device. When we load a flash resource STB calls the CVC verify and trusted measure(sha512) functions. So when we have a flash resource gets loaded before STB initialization, then cvc verify function fails to start the verify and enforce the boot.

  Below is one of the example failure where our VERSION partition gets loading early in the boot stage without STB initialization done.

  This is with secure mode off. STB: VERSION NOT VERIFIED, invalid param. buf=0x305ed930, len=4096 key-hash=0x0 hash-size=0

  In the same code path when secure mode is on, the boot process will abort.

  So this patch fixes this issue by calling cvc verify only if we have STB init was done.

  And also we need a permanent fix in init path to ensure STB init gets done at first place and then start loading all other flash resources.
- libstb/tpm_chip: Add missing new line to print messages.
- libstb: increase the log level of verify/measure messages to PR_NOTICE.

  Currently libstb logs the verify and hash caluculation messages in PR_INFO level. So when there is a secure boot enforcement happens in loading last flash resource(Ex: BOOTKERNEL), the previous verify and measure messages are not logged to console, which is not clear to the end user which resource is verified and measured. So this patch fixes this by increasing the log level to PR_NOTICE.

5.1.32 skiboot-5.10-rc3

skiboot v5.10-rc3 was released on Thursday February 15th 2018. It is the third release candidate of skiboot 5.10, which will become the new stable release of skiboot following the 5.9 release, first released October 31st 2017.

skiboot v5.10-rc3 contains all bug fixes as of *skiboot-5.9.8* and *skiboot-5.4.9* (the currently maintained stable releases). There may be more 5.9.x stable releases, it will depend on demand.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.10 in February, with skiboot 5.10 being for all POWER8 and POWER9 platforms in op-build v1.21. This release will be targeted to early POWER9 systems.

Over skiboot-5.10-rc2, we have the following changes:

- vas: Disable VAS/NX-842 on some P9 revisions

  VAS/NX-842 are not functional on some P9 revisions, so disable them in hardware and skip creating their device tree nodes.

  Since the intent is to prevent OS from configuring VAS/NX, we remove only the platform device nodes but leave the VAS/NX DT nodes under xscom (i.e we don't skip add_vas_node() in hdata/spira.c)

- phb4: Only escalate freezes on MMIO load where necessary

  In order to work around a hardware issue, MMIO load freezes were escalated to fences on every chip. Now that hardware no longer requires this, restrict escalation to the chips that actually need it.

- pflash: Fix makefile dependency issue
- DT: Add "version" property under ibm, firmware-versions node

  First line of VERSION section in PNOR contains firmware version. Use that to add "version" property under firmware versions dt node.

  Sample output:

```
root@xxx2:/proc/device-tree/ibm,firmware-versions# lsprop
version          "witherspoon-ibm-OP9_v1.19_1.94"
```

- npu2: Disable TVT range check when in bypass mode

  On POWER9 the GPUs need to be able to access the MMIO memory space. Therefore the TVT range check needs to include the MMIO address space. As any possible range check would cover all of memory anyway this patch just disables the TVT range check all together when bypassing the TCE tables.

- hw/npu2: support creset of npu2 devices

  creset calls in the hw procedure that resets the PHY, we don't take them out of reset, just put them in reset.

  this fixes a kexec issue.

- ATTN: Enable flush instruction cache bit in HID register

  In P9, we have to enable "flush the instruction cache" bit along with "attn instruction support" bit to trigger attention.

- capi: Enable channel tag streaming for PHB in CAPP mode

  We re-enable channel tag streaming for PHB in CAPP mode as without it PEC was waiting for cresp for each DMA write command before sending a new DMA write command on the Powerbus. This resulted in much lower DMA write performance than expected.

  The patch updates enable_capi_mode() to remove the masking of channel_streaming_en bit in PBCQ Hardware Configuration Register. Also does some re-factoring of the code that updates this register to use xscom_write_mask instead of xscom_read followed by a xscom_write.

- core/device.c: Fix dt_find_compatible_node

  dt_find_compatible_node() and dt_find_compatible_node_on_chip() are used to find device nodes under a parent/root node with a given compatible property.

  dt_next(root, prev) is used to walk the child nodes of the given parent and takes two arguments - root contains the parent node to walk whilst prev contains the previous child to search from so that it can be used as an iterator over all children nodes.

  The first iteration of dt_find_compatible_node(root, prev) calls dt_next(root, root) which is not a well defined operation as prev is assumed to be child of the root node. The result is that when a node contains no children it will start returning the parent nodes siblings until it hits the top of the tree at which point a NULL derefence is attempted when looking for the root nodes parent.

Dereferencing NULL can result in undesirable data exceptions during system boot and untimely non-hilarious system crashes. dt_next() should not be called with prev == root. Instead we add a check to dt_next() such that passing prev = NULL will cause it to start iterating from the first child node (if any).

- stb: Put correct label (for skiboot) into container

  Hostboot will expect the label field of the stb header to contain "PAYLOAD" for skiboot or it will fail to load and run skiboot.

  The failure looks something like this:

```
53.40896|ISTEP 20. 1 - host_load_payload
53.65840|secure|Secureboot Failure plid = 0x90000755, rc = 0x1E07

53.65881|System shutting down with error status 0x1E07
53.67547|================================================
53.67954|Error reported by secure (0x1E00) PLID 0x90000755
53.67560|  Container's component ID does not match expected component ID
53.67561|  ModuleId   0x09 SECUREBOOT::MOD_SECURE_VERIFY_COMPONENT
53.67845|  ReasonCode 0x1e07 SECUREBOOT::RC_ROM_VERIFY
53.67998|  UserData1   : 0x0000000000000000
53.67999|  UserData2   : 0x0000000000000000
53.67999|------------------------------------------------
53.68000|  Callout type             : Procedure Callout
53.68000|  Procedure                : EPUB_PRC_HB_CODE
53.68001|  Priority                 : SRCI_PRIORITY_HIGH
53.68001|------------------------------------------------
53.68002|  Callout type             : Procedure Callout
53.68003|  Procedure                : EPUB_PRC_FW_VERIFICATION_ERR
53.68003|  Priority                 : SRCI_PRIORITY_HIGH
53.68004|------------------------------------------------
```

- hw/occ: Fix fast-reboot crash in P8 platforms.

  commit 85a1de35cbe4 ("fast-boot: occ: Re-parse the pstate table during fast-boot") breaks the fast-reboot on P8 platforms while reiniting the OCC pstates. On P8 platforms OPAL adds additional two properties #address-cells and #size-cells under ibm,opal/power-mgmt/ DT node. While in fast-reboot same properties adding back to the same node results in Duplicate properties and hence fast-reboot fails with below traces.

```
[  541.410373292,5] OCC: All Chip Rdy after 0 ms
[  541.410488745,3] Duplicate property "#address-cells" in node /ibm,opal/power-
↪mgt
[  541.410694290,0] Aborting!
CPU 0058 Backtrace:
 S: 0000000031d639d0 R: 000000003001367c   .backtrace+0x48
 S: 0000000031d63a60 R: 000000003001a03c   ._abort+0x4c
 S: 0000000031d63ae0 R: 00000000300267d8   .new_property+0xd8
 S: 0000000031d63b70 R: 0000000030026a28   .__dt_add_property_cells+0x30
 S: 0000000031d63c10 R: 000000003003ea3c   .occ_pstates_init+0x984
 S: 0000000031d63d90 R: 00000000300142d8   .load_and_boot_kernel+0x86c
 S: 0000000031d63e70 R: 000000003002586c   .fast_reboot_entry+0x358
 S: 0000000031d63f00 R: 00000000300029f4   fast_reset_entry+0x2c
```

  This patch fixes this issue by removing these two properties on P8 while doing OCC pstates re-init in fast-reboot code path.

5.1.33 skiboot-5.10-rc4

skiboot v5.10-rc4 was released on Wednesday February 21st 2018. It is the fourth release candidate of skiboot 5.10, which will become the new stable release of skiboot following the 5.9 release, first released October 31st 2017.

skiboot v5.10-rc4 contains all bug fixes as of *skiboot-5.9.8* and *skiboot-5.4.9* (the currently maintained stable releases). There may be more 5.9.x stable releases, it will depend on demand.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.10 in February, with skiboot 5.10 being for all POWER8 and POWER9 platforms in op-build v1.21. This release will be targeted to early POWER9 systems.

Over skiboot-5.10-rc3, we have the following changes:

- core: Fix mismatched names between reserved memory nodes & properties

  OPAL exposes reserved memory regions through the device tree in both new (nodes) and old (properties) formats.

  However, the names used for these don't match - we use a generated cell address for the nodes, but the plain region name for the properties.

  This fixes a warning from FWTS

- sensor-groups: occ: Add support to disable/enable sensor group

  This patch adds a new opal call to enable/disable a sensor group. This call is used to select the sensor groups that needs to be copied to main memory by OCC at runtime.

- sensors: occ: Add energy counters

  Export the accumulated power values as energy sensors. The accumulator field of power sensors are used for representing energy counters which can be exported as energy counters in Linux hwmon interface.

- sensors: Support reading u64 sensor values

  This patch adds support to read u64 sensor values. This also adds changes to the core and the backend implementation code to make this API as the base call. Host can use this new API to read sensors upto 64bits.

  This adds a list to store the pointer to the kernel u32 buffer, for older kernels making async sensor u32 reads.

- dt: add /cpus/ibm,powerpc-cpu-features device tree bindings

  This is a new CPU feature advertising interface that is fine-grained, extensible, aware of privilege levels, and gives control of features to all levels of the stack (firmware, hypervisor, and OS).

  The design and binding specification is described in detail in doc/.

- phb3/phb4/p7ioc: Document supported TCE sizes in DT

  Add a new property, "ibm,supported-tce-sizes", to advertise to Linux how big the available TCE sizes are. Each value is a bit shift, from smallest to largest.

- phb4: Fix TCE page size

  The page sizes for TCEs on P9 were inaccurate and just copied from PHB3, so correct them.

- Revert "pci: Shared slot state synchronisation for hot reset"

  An issue was found in shared slot reset where the system can be stuck in an infinite loop, pull the code out until there's a proper fix.

  This reverts commit 1172a6c57ff3c66f6361e572a1790cbcc0e5ff37.

- hdata/iohub: Use only wildcard slots for pluggables

  We don't want to cause a VID:DID check against pluggable devices, as they may use multiple devids.

  Narrow the condition under which VID:DID is listed in the dt, so that we'll end up creating a wildcard slot for these instead.

- increase log verbosity in debug builds
- Add -debug to version on DEBUG builds
- cpu_wait_job: Correctly report time spent waiting for job

5.1.34 skiboot-5.10.1

skiboot 5.10.1 was released on Thursday March 1st, 2018. It replaces *skiboot-5.10* as the current stable release in the 5.10.x series.

Over *skiboot-5.10*, we have an improvement for debugging NPU2/NVLink problems and a bug fix. These changes are:

- NPU2 HMIs: dump out a *LOT* of npu2 registers for debugging
- libflash/blocklevel: Correct miscalculation in blocklevel_smart_erase()

  This fixes a bug in pflash.

  If blocklevel_smart_erase() detects that the smart erase fits entire in one erase block, it has an early bail path. In this path it miscaculates where in the buffer the backend needs to read from to perform the final write.

  Fixes: https://github.com/open-power/skiboot/issues/151

5.1.35 skiboot-5.10.2

skiboot 5.10.2 was released on Tuesday March 6th, 2018. It replaces *skiboot-5.10.1* as the current stable release in the 5.10.x series.

Over *skiboot-5.10.1*, we have one improvement:

- Tie tm-suspend fw-feature and opal_reinit_cpus() together

  Currently opal_reinit_cpus(OPAL_REINIT_CPUS_TM_SUSPEND_DISABLED) always returns OPAL_UNSUPPORTED.

  This ties the tm suspend fw-feature to the opal_reinit_cpus(OPAL_REINIT_CPUS_TM_SUSPEND_DISABLED) so that when tm suspend is disabled, we correctly report it to the kernel. For backwards compatibility, it's assumed tm suspend is available if the fw-feature is not present.

  Currently hostboot will clear fw-feature(TM_SUSPEND_ENABLED) on P9N DD2.1. P9N DD2.2 will set fw-feature(TM_SUSPEND_ENABLED). DD2.0 and below has TM disabled completely (not just suspend).

  We are using opal_reinit_cpus() to determine this setting (rather than the device tree/HDAT) as some future firmware may let us change this dynamically after boot. That is not the case currently though.

5.1.36 skiboot-5.10.3

skiboot 5.10.3 was released on Thursday March 28th, 2018. It replaces *skiboot-5.10.2* as the current stable release in the 5.10.x series.

It is recommended that 5.10.3 be used instead of any previous 5.10.x version due to the bug fixes and debugging enhancements in it.

Over *skiboot-5.10.2*, we have a few improvements and bug fixes:

- NPU2: dump NPU2 registers on npu2 HMI

  Due to the nature of debugging npu2 issues, folk are wanting the full list of NPU2 registers dumped when there's a problem.

  This is different than the solution introduced in 5.10.1 as there we would dump the registers in a way that would trigger a FIR bit that would confuse PRD.

- npu2: Add performance tuning SCOM inits

  Peer-to-peer GPU bandwidth latency testing has produced some tunable values that improve performance. Add them to our device initialization.

  File these under things that need to be cleaned up with nice #defines for the register names and bitfields when we get time.

  A few of the settings are dependent on the system's particular NVLink topology, so introduce a helper to determine how many links go to a single GPU.

- hw/npu2: Assign a unique LPARSHORTID per GPU

  This gets used elsewhere to index items in the XTS tables.

- occ: Set up OCC messaging even if we fail to setup pstates

  This means that we no longer hit this bug if we fail to get valid pstates from the OCC.

```
[console-pexpect]#echo 1 > //sys/firmware/opal/sensor_groups//occ-csm0/clear
echo 1 > //sys/firmware/opal/sensor_groups//occ-csm0/clear
[   94.019971181,5] CPU ATTEMPT TO RE-ENTER FIRMWARE! PIR=083d cpu @0x33cf4000 ->␣
↪pir=083d token=8
[   94.020098392,5] CPU ATTEMPT TO RE-ENTER FIRMWARE! PIR=083d cpu @0x33cf4000 ->␣
↪pir=083d token=8
[   10.318805] Disabling lock debugging due to kernel taint
[   10.318808] Severe Machine check interrupt [Not recovered]
[   10.318812]   NIP [000000003003e434]: 0x3003e434
[   10.318813]   Initiator: CPU
[   10.318815]   Error type: Real address [Load/Store (foreign)]
[   10.318817] opal: Hardware platform error: Unrecoverable Machine Check␣
↪exception
[   10.318821] CPU: 117 PID: 2745 Comm: sh Tainted: G   M             4.15.9-
↪openpower1 #3
[   10.318823] NIP:  000000003003e434 LR: 000000003003025c CTR: 0000000030030240
[   10.318825] REGS: c00000003fa7bd80 TRAP: 0200   Tainted: G   M             (4.
↪15.9-openpower1)
[   10.318826] MSR:  9000000000201002 <SF,HV,ME,RI>  CR: 48002888  XER: 20040000
[   10.318831] CFAR: 0000000030030258 DAR: 394a00147d5a03a6 DSISR: 00000008␣
↪SOFTE: 1
```

- core/fast-reboot: disable fast reboot upon fundamental entry/exit/locking errors

  This disables fast reboot in several more cases where serious errors like lock corruption or call re-entrancy are detected.

- core/opal: allow some re-entrant calls

  This allows a small number of OPAL calls to succeed despite re-entering the firmware, and rejects others rather than aborting.

This allows a system reset interrupt that interrupts OPAL to do something useful. Sreset other CPUs, use the console, which allows xmon to work or stack traces to be printed, reboot the system.

Use OPAL_INTERNAL_ERROR when rejecting, rather than OPAL_BUSY, which is used for many other things that does not mean a serious permanent error.

- core/opal: abort in case of re-entrant OPAL call

  The stack is already destroyed by the time we get here, so there is not much point continuing.

- npu2: Disable fast reboot

  Fast reboot does not yet work right with the NPU. It's been disabled on NVLink and OpenCAPI machines. Do the same for NVLink2.

  This amounts to a port of 3e4577939bbf ("npu: Fix broken fast reset") from the npu code to npu2.

5.1.37 skiboot-5.10.4

skiboot 5.10.4 was released on Wednesday April 4th, 2018. It replaces *skiboot-5.10.3* as the current stable release in the 5.10.x series.

It is recommended that 5.10.3 be used instead of any previous 5.10.x version due to the bug fixes and debugging enhancements in it.

Over *skiboot-5.10.3*, we have one bug fix:

- xive: disable store EOI support

  Hardware has limitations which would require to put a sync after each store EOI to make sure the MMIO operations that change the ESB state are ordered. This is a killer for performance and the PHBs do not support the sync. So remove the store EOI for the moment, until hardware is improved.

  Also, while we are at changing the XIVE source flags, let's fix the settings for the PHB4s which should follow these rules :

  – SHIFT_BUG for DD10

  – STORE_EOI for DD20 and if enabled

  – TRIGGER_PAGE for DDx0 and if not STORE_EOI

5.1.38 skiboot-5.10.5

skiboot 5.10.5 was released on Tuesday April 24th, 2018. It replaces *skiboot-5.10.4* as the current stable release in the 5.10.x series.

It is recommended that 5.10.5 be used instead of any previous 5.10.x version due to the bug fixes and debugging enhancements in it.

Over *skiboot-5.10.4*, we have four bug fixes:

- npu2/hw-procedures: fence bricks on GPU reset

  The NPU workbook defines a way of fencing a brick and getting the brick out of fence state. We do have an implementation of bringing the brick out of fenced/quiesced state. We do the latter in our procedures, but to support run time reset we need to do the former.

  The fencing ensures that access to memory behind the links will not lead to HMI's, but instead SUE's will be populated in cache (in the case of speculation). The expectation is then that prior to and after reset, the operating system components will flush the cache for the region of memory behind the GPU.

This patch does the following:

1. Implements a npu2_dev_fence_brick() function to set/clear fence state
2. Clear FIR bits prior to clearing the fence status
3. Clear's the fence status
4. We take the powerbus out of CQ fence much later now, in credits_check() which is the last hardware procedure called after link training.

- hdata/spira: parse vpd to add part-number and serial-number to xscom@ node

  Expected by FWTS and associates our processor with the part/serial number, which is obviously a good thing for one's own sanity.

- hw/imc: Check for pause_microcode_at_boot() return status

  pause_microcode_at_boot() loops through all the chip's ucode control block and pause the ucode if it is in the running state. But it does not fail if any of the chip's ucode is not initialised.

  Add code to return a failure if ucode is not initialized in any of the chip. Since pause_microcode_at_boot() is called just before attaching the IMC device nodes in imc_init(), add code to check for the function return.

- core/cpufeatures: Fix setting DARN and SCV HWCAP feature bits

  DARN and SCV has been assigned AT_HWCAP2 (32-63) bits:

  ```
  #define PPC_FEATURE2_DARN               0x00200000 /* darn random number insn */
  #define PPC_FEATURE2_SCV                0x00100000 /* scv syscall */
  ```

  A cpufeatures-aware OS will not advertise these to userspace without this patch.

5.1.39 skiboot-5.10.6

skiboot 5.10.6 was released on Monday May 28th, 2018. It replaces *skiboot-5.10.5* as the current stable release in the 5.10.x series.

It is recommended that 5.10.6 be used instead of any previous 5.10.x version, especially due to the locking bug fixes.

It is expected that this will be the final 5.10.x version, with 6.0.x taking over as the main stable branch.

Over *skiboot-5.10.5*, we have the following fixes:

- opal-prd: Do not error out on first failure for soft/hard offline.

  The memory errors (CEs and UEs) that are detected as part of background memory scrubbing are reported by PRD asynchronously to opal-prd along with affected memory ranges. hservice_memory_error() converts these ranges into page granularity before hooking up them to soft/hard offline-ing infrastructure.

  But the current implementation of hservice_memory_error() does not hookup all the pages to soft/hard offline-ing if any of the page offline action fails. e.g hard offline can fail for:

  – Pages that are not part of buddy managed pool.
  – Pages that are reserved by kernel using memblock_reserved()
  – Pages that are in use by kernel.

  But for the pages that are in use by user space application, the hard offline marks the page as hwpoison, sends SIGBUS signal to kill the affected application as recovery action and returns success.

  Hence, It is possible that some of the pages in that memory range are in use by application or free. By stopping on first error we loose the opportunity to hwpoison the subsequent pages which may be free or in use by application. This patch fixes this issue.

- xive: fix missing unlock in error path

  Found with sparse and some added lock annotations.

- OPAL_PCI_SET_POWER_STATE: fix locking in error paths

  Otherwise we could exit OPAL holding locks, potentially leading to all sorts of problems later on.

5.1.40 skiboot-5.11

skiboot v5.11 was released on Friday April 6th 2018. It is the first release of skiboot 5.11, which is now the new stable release of skiboot following the 5.10 release, first released February 23rd 2018.

It is *not* expected to keep the 5.11 branch around for long, and instead quickly move onto a 6.0, which will mark the basis for op-build v2.0 and will be required for POWER9 systems.

It is expected that skiboot 6.0 will follow very shortly. Consider 5.11 more of a beta release to 6.0 than anything. For POWER9 systems it should certainly be more solid than previous releases though.

skiboot v5.11 contains all bug fixes as of *skiboot-5.10.4* and *skiboot-5.4.9* (the currently maintained stable releases). There may be more 5.10.x stable releases, it will depend on demand.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

Over skiboot-5.10, we have the following changes:

New Platforms

- Add VESNIN platform support

  The Vesnin platform from YADRO is a 4 socked POWER8 system with up to 8TB of memory with 460GB/s of memory bandwidth in only 2U. Many kudos to the team from Yadro for submitting their code upstream!

New Features

- fast-reboot: enable by default for POWER9
  - Fast reboot is disabled if NPU2 is present or CAPI2/OpenCAPI is used
- PCI tunneled operations on PHB4
  - phb4: set PBCQ Tunnel BAR for tunneled operations

    P9 supports PCI tunneled operations (atomics and as_notify) that are initiated by devices.

    A subset of the tunneled operations require a response, that must be sent back from the host to the device. For example, an atomic compare and swap will return the compare status, as swap will only performed in case of success. Similarly, as_notify reports if the target thread has been woken up or not, because the operation may fail.

    To enable tunneled operations, a device driver must tell the host where it expects tunneled operation responses, by setting the PBCQ Tunnel BAR Response register with a specific value within the range of its BARs.

    This register is currently initialized by enable_capi_mode(). But, as tunneled operations may also operate in PCI mode, a new API is required to set the PBCQ Tunnel BAR Response register, without switching to CAPI mode.

    This patch provides two new OPAL calls to get/set the PBCQ Tunnel BAR Response register.

Note: as there is only one PBCQ Tunnel BAR register, shared between all the devices connected to the same PHB, only one of these devices will be able to use tunneled operations, at any time.

 – phb4: set PHB CMPM registers for tunneled operations

 P9 supports PCI tunneled operations (atomics and as_notify) that require setting the PHB ASN Compare/Mask register with a 16-bit indication.

 This register is currently initialized by enable_capi_mode(). But, as tunneled operations may also work in PCI mode, the ASN Compare/Mask register should rather be initialized in phb4_init_ioda3().

 This patch also adds "ibm,phb-indications" to the device tree, to tell Linux the values of CAPI, ASN, and NBW indications, when supported.

 Tunneled operations tested by IBM in CAPI mode, by Mellanox Technologies in PCI mode.

- Tie tm-suspend fw-feature and opal_reinit_cpus() together

 Currently opal_reinit_cpus(OPAL_REINIT_CPUS_TM_SUSPEND_DISABLED) always returns OPAL_UNSUPPORTED.

 This ties the tm suspend fw-feature to the opal_reinit_cpus(OPAL_REINIT_CPUS_TM_SUSPEND_DISABLED) so that when tm suspend is disabled, we correctly report it to the kernel. For backwards compatibility, it's assumed tm suspend is available if the fw-feature is not present.

 Currently hostboot will clear fw-feature(TM_SUSPEND_ENABLED) on P9N DD2.1. P9N DD2.2 will set fw-feature(TM_SUSPEND_ENABLED). DD2.0 and below has TM disabled completely (not just suspend).

 We are using opal_reinit_cpus() to determine this setting (rather than the device tree/HDAT) as some future firmware may let us change this dynamically after boot. That is not the case currently though.

Power Management

- SLW: Increase stop4-5 residency by 10x

 Using DGEMM benchmark we observed there was a drop of 5-9% throughput with and without stop4/5. In this benchmark the GPU waits on the cpu to wakeup and provide the subsequent data block to compute. The wakup latency accumulates over the run and shows up as a performance drop.

 Linux enters stop4/5 more aggressively for its wakeup latency. Increasing the residency from 1ms to 10ms makes the performance drop <1%

- occ: Set up OCC messaging even if we fail to setup pstates

 This means that we no longer hit this bug if we fail to get valid pstates from the OCC.

```
[console-pexpect]#echo 1 > //sys/firmware/opal/sensor_groups//occ-csm0/clear
echo 1 > //sys/firmware/opal/sensor_groups//occ-csm0/clear
[   94.019971181,5] CPU ATTEMPT TO RE-ENTER FIRMWARE! PIR=083d cpu @0x33cf4000 ->␣
↪pir=083d token=8
[   94.020098392,5] CPU ATTEMPT TO RE-ENTER FIRMWARE! PIR=083d cpu @0x33cf4000 ->␣
↪pir=083d token=8
[   10.318805] Disabling lock debugging due to kernel taint
[   10.318808] Severe Machine check interrupt [Not recovered]
[   10.318812]   NIP [000000003003e434]: 0x3003e434
[   10.318813]   Initiator: CPU
[   10.318815]   Error type: Real address [Load/Store (foreign)]
[   10.318817] opal: Hardware platform error: Unrecoverable Machine Check␣
↪exception
[   10.318821] CPU: 117 PID: 2745 Comm: sh Tainted: G   M             4.15.9-
↪openpower1 #3
```

(continues on next page)

(continued from previous page)

```
[   10.318823] NIP:  000000003003e434 LR: 000000003003025c CTR: 0000000030030240
[   10.318825] REGS: c00000003fa7bd80 TRAP: 0200   Tainted: G   M            (4.
↪15.9-openpower1)
[   10.318826] MSR:  9000000000201002 <SF,HV,ME,RI>  CR: 48002888  XER: 20040000
[   10.318831] CFAR: 0000000030030258 DAR: 394a00147d5a03a6 DSISR: 00000008␣
↪SOFTE: 1
```

mbox based platforms

For platforms using the mbox protocol for host flash access (all BMC based OpenPOWER systems, most OpenBMC based systems) there have been some hardening efforts in the event of the BMC being poorly behaved.

- mbox: Reduce default BMC timeouts

  Rebooting a BMC can take 70 seconds. Skiboot cannot possibly spin for 70 seconds waiting for a BMC to come back. This also makes the current default of 30 seconds a bit pointless, is it far too short to be a worse case wait time but too long to avoid hitting hardlockup detectors and wrecking havoc inside host linux.

  Just change it to three seconds so that host linux will survive and that, reads and writes will fail but at least the host stays up.

  Also refactored the waiting loop just a bit so that it's easier to read.

- mbox: Harden against BMC daemon errors

  Bugs present in the BMC daemon mean that skiboot gets presented with mbox windows of size zero. These windows cannot be valid and skiboot already detects these conditions.

  Currently skiboot warns quite strongly about the occurrence of these problems. The problem for skiboot is that it doesn't take any action. Initially I wanting to avoid putting policy like this into skiboot but since these bugs aren't going away and skiboot barfing is leading to lockups and ultimately the host going down something needs to be done.

  I propose that when we detect the problem we fail the mbox call and punt the problem back up to Linux. I don't like it but at least it will cause errors to cascade and won't bring the host down. I'm not sure how Linux is supposed to detect this or what it can even do but this is better than a crash.

  Diagnosing a failure to boot if skiboot its self fails to read flash may be marginally more difficult with this patch. This is because skiboot will now only print one warning about the zero sized window rather than continuously spitting it out.

Fast Reboot Improvements

Around fast-reboot we have made several improvements to harden the fast reboot code paths and resort to a full IPL if something doesn't look right.

- core/fast-reboot: zero memory after fast reboot

  This improves the security and predictability of the fast reboot environment.

  There can not be a secure fence between fast reboots, because a malicious OS can modify the firmware itself. However a well-behaved OS can have a reasonable expectation that OS memory regions it has modified will be cleared upon fast reboot.

  The memory is zeroed after all other CPUs come up from fast reboot, just before the new kernel is loaded and booted into. This allows image preloading to run concurrently, and will allow parallelisation of the clearing in future.

- core/fast-reboot: verify mem regions before fast reboot

  Run the mem_region sanity checkers before proceeding with fast reboot.

  This is the beginning of proactive sanity checks on opal data for fast reboot (with complements the reactive disable_fast_reboot cases). This is encouraged to re-use and share any kind of debug code and unit test code.

- fast-reboot: occ: Only delete /ibm, opal/power-mgt nodes if they exist
- core/fast-reboot: disable fast reboot upon fundamental entry/exit/locking errors

  This disables fast reboot in several more cases where serious errors like lock corruption or call re-entrancy are detected.

- capp: Disable fast-reboot whenever enable_capi_mode() is called

  This patch updates phb4_set_capi_mode() to disable fast-reboot whenever enable_capi_mode() is called, irrespective to its return value. This should prevent against a possibility of not disabling fast-reboot when some changes to enable_capi_mode() causing return of an error and leaving CAPP in enabled mode.

- fast-reboot: occ: Delete OCC child nodes in /ibm, opal/power-mgt

  Fast-reboot in P8 fails to re-init OCC data as there are chipwise OCC nodes which are already present in the /ibm,opal/power-mgt node. These per-chip nodes hold the voltage IDs for each pstate and these can be changed on OCC pstate table biasing. So delete these before calling the re-init code to re-parse and populate the pstate data.

Debugging/SRESET improvemens

Since *skiboot-5.11-rc1*:

- core/cpu: Prevent clobbering of stack guard for boot-cpu

  Commit 90d53934c2da ("core/cpu: discover stack region size before initialising memory regions") introduced memzero for struct cpu_thread in init_cpu_thread(). This has an unintended side effect of clobbering the stack-guard cannery of the boot_cpu stack. This results in opal failing to init with this failure message:

  ```
  CPU: P9 generation processor (max 4 threads/core)
  CPU: Boot CPU PIR is 0x0004 PVR is 0x004e1200
  Guard skip = 0
  Stack corruption detected !
  Aborting!
  CPU 0004 Backtrace:
   S: 0000000031c13ab0 R: 0000000030013b0c   .backtrace+0x5c
   S: 0000000031c13b50 R: 000000003001bd18   ._abort+0x60
   S: 0000000031c13be0 R: 0000000030013bbc   .__stack_chk_fail+0x54
   S: 0000000031c13c60 R: 00000000300c5b70   .memset+0x12c
   S: 0000000031c13d00 R: 0000000030019aa8   .init_cpu_thread+0x40
   S: 0000000031c13d90 R: 000000003001b520   .init_boot_cpu+0x188
   S: 0000000031c13e30 R: 0000000030015050   .main_cpu_entry+0xd0
   S: 0000000031c13f00 R: 0000000030002700   boot_entry+0x1c0
  ```

  So the patch provides a fix by tweaking the memset() call in init_cpu_thread() to skip over the stack-guard cannery.

- core/lock.c: ensure valid start value for lock spin duration warning

  The previous fix in a8e6cc3f4 only addressed half of the problem, as we could also get an invalid value for start, causing us to fail in a weird way.

  This was caught by the testcases.OpTestHMIHandling.HMI_TFMR_ERRORS test in op-test-framework.

You'd get to this part of the test and get the erroneous lock spinning warnings:

```
PATH=/usr/local/sbin:$PATH putscom -c 00000000 0x2b010a84 0003080000000000
0000080000000000
[  790.140976993,4] WARNING: Lock has been spinning for 790275ms
[  790.140976993,4] WARNING: Lock has been spinning for 790275ms
[  790.140976918,4] WARNING: Lock has been spinning for 790275ms
```

This patch checks the validity of timebase before setting start, and only checks the lock timeout if we got a valid start value.

Since *skiboot-5.10*:

- core/opal: allow some re-entrant calls

  This allows a small number of OPAL calls to succeed despite re-entering the firmware, and rejects others rather than aborting.

  This allows a system reset interrupt that interrupts OPAL to do something useful. Sreset other CPUs, use the console, which allows xmon to work or stack traces to be printed, reboot the system.

  Use OPAL_INTERNAL_ERROR when rejecting, rather than OPAL_BUSY, which is used for many other things that does not mean a serious permanent error.

- core/opal: abort in case of re-entrant OPAL call

  The stack is already destroyed by the time we get here, so there is not much point continuing.

- core/lock: Add lock timeout warnings

  There are currently no timeout warnings for locks in skiboot. We assume that the lock will eventually become free, which may not always be the case.

  This patch adds timeout warnings for locks. Any lock which spins for more than 5 seconds will throw a warning and stacktrace for that thread. This is useful for debugging siturations where a lock which hang, waiting for the lock to be freed.

- core/lock: Add deadlock detection

  This adds simple deadlock detection. The detection looks for circular dependencies in the lock requests. It will abort and display a stack trace when a deadlock occurs. The detection is enabled by DEBUG_LOCKS (enabled by default). While the detection may have a slight performance overhead, as there are not a huge number of locks in skiboot this overhead isn't significant.

- core/hmi: report processor recovery reason from core FIR bits on P9

  When an error is encountered that causes processor recovery, HMI is generated if the recovery was successful. The reason is recorded in the core FIR, which gets copied into the WOF.

  In this case dump the WOF register and an error string into the OPAL msglog.

  A broken init setting led to HMIs reported in Linux as:

```
[    3.591547] Harmless Hypervisor Maintenance interrupt [Recovered]
[    3.591648]  Error detail: Processor Recovery done
[    3.591714]  HMER: 2040000000000000
```

  This patch would have been useful because it tells us exactly that the problem is in the d-side ERAT:

```
[  414.489690798,7] HMI: Received HMI interrupt: HMER = 0x2040000000000000
[  414.489693339,7] HMI: [Loc: UOPWR.0000000-Node0-Proc0]: P:0 C:1 T:1: Processor␣
↪recovery occurred.
[  414.489699837,7] HMI: Core WOF = 0x0000000410000000 recovered error:
```

(continues on next page)

(continued from previous page)

```
[  414.489701543,7] HMI: LSU - SRAM (DCACHE parity, etc)
[  414.489702341,7] HMI: LSU - ERAT multi hit
```

In future it will be good to unify this reporting, so Linux could print something more useful. Until then, this gives some good data.

NPU2/NVLink2 Fixes

- npu2: Add performance tuning SCOM inits

  Peer-to-peer GPU bandwidth latency testing has produced some tunable values that improve performance. Add them to our device initialization.

  File these under things that need to be cleaned up with nice #defines for the register names and bitfields when we get time.

  A few of the settings are dependent on the system's particular NVLink topology, so introduce a helper to determine how many links go to a single GPU.

- hw/npu2: Assign a unique LPARSHORTID per GPU

  This gets used elsewhere to index items in the XTS tables.

- NPU2: dump NPU2 registers on npu2 HMI

  Due to the nature of debugging npu2 issues, folk are wanting the full list of NPU2 registers dumped when there's a problem.

- npu2: Remove DD1 support

  Major changes in the NPU between DD1 and DD2 necessitated a fair bit of revision-specific code.

  Now that all our lab machines are DD2, we no longer test anything on DD1 and it's time to get rid of it.

  Remove DD1-specific code and abort probe if we're running on a DD1 machine.

- npu2: Disable fast reboot

  Fast reboot does not yet work right with the NPU. It's been disabled on NVLink and OpenCAPI machines. Do the same for NVLink2.

  This amounts to a port of 3e4577939bbf ("npu: Fix broken fast reset") from the npu code to npu2.

- npu2: Use unfiltered mode in XTS tables

  The XTS_PID context table is limited to 256 possible pids/contexts. To relieve this limitation, make use of "unfiltered mode" instead.

  If an entry in the XTS_BDF table has the bit for unfiltered mode set, we can just use one context for that entire bdf/lpar, regardless of pid. Instead of of searching the XTS_PID table, the NMMU checkout request will simply use the entry indexed by lparshort id instead.

  Change opal_npu_init_context() to create these lparshort-indexed wildcard entries (0-15) instead of allocating one for each pid. Check that multiple calls for the same bdf all specify the same msr value.

  In opal_npu_destroy_context(), continue validating the bdf argument, ensuring that it actually maps to an lpar, but no longer remove anything from the XTS_PID table. If/when we start supporting virtualized GPUs, we might consider actually removing these wildcard entries by keeping a refcount, but keep things simple for now.

CAPI/OpenCAPI

Since *skiboot-5.11-rc1*:

- capi: Poll Err/Status register during CAPP recovery

  This patch updates do_capp_recovery_scoms() to poll the CAPP Err/Status control register, check for CAPP-Recovery to complete/fail based on indications of BITS-1,5,9 and then proceed with the CAPP-Recovery scoms iif recovery completed successfully. This would prevent cases where we bring-up the PCIe link while recovery sequencer on CAPP is still busy with casting out cache lines.

  In case CAPP-Recovery didn't complete successfully an error is returned from do_capp_recovery_scoms() asking phb4_creset() to keep the phb4 fenced and mark it as broken.

  The loop that implements polling of Err/Status register will also log an error on the PHB when it continues for more than 168ms which is the max time to failure for CAPP-Recovery.

Since *skiboot-5.10*:

- npu2-opencapi: Add OpenCAPI OPAL API calls

  Add three OPAL API calls that are required by the ocxl driver.

  – OPAL_NPU_SPA_SETUP

    The Shared Process Area (SPA) is a table containing one entry (a "Process Element") per memory context which can be accessed by the OpenCAPI device.

  – OPAL_NPU_SPA_CLEAR_CACHE

    The NPU keeps a cache of recently accessed memory contexts. When a Process Element is removed from the SPA, the cache for the link must be cleared.

  – OPAL_NPU_TL_SET

    The Transaction Layer specification defines several templates for messages to be exchanged on the link. During link setup, the host and device must negotiate what templates are supported on both sides and at what rates those messages can be sent.

- npu2-opencapi: Train OpenCAPI links and setup devices

  Scan the OpenCAPI links under the NPU, and for each link, reset the card, set up a device, train the link and register a PHB.

  Implement the necessary operations for the OpenCAPI PHB type.

  For bringup, test and debug purposes, we allow an NVRAM setting, "opencapi-link-training" that can be set to either disable link training completely or to use the prbs31 test pattern.

  To disable link training:

  ```
  nvram -p ibm,skiboot --update-config opencapi-link-training=none
  ```

  To use prbs31:

  ```
  nvram -p ibm,skiboot --update-config opencapi-link-training=prbs31
  ```

- npu2-hw-procedures: Add support for OpenCAPI PHY link training

  Unlike NVLink, which uses the pci-virt framework to fake a PCI configuration space for NVLink devices, the OpenCAPI device model presents us with a real configuration space handled by the device over the OpenCAPI link.

  As a result, we have to train the OpenCAPI link in skiboot before we do PCI probing, so that config space can be accessed, rather than having link training being triggered by the Linux driver.

- npu2-opencapi: Configure NPU for OpenCAPI

  Scan the device tree for NPUs with OpenCAPI links and configure the NPU per the initialisation sequence in the NPU OpenCAPI workbook.

- capp: Make error in capp timebase sync a non-fatal error

  Presently when we encounter an error while synchronizing capp timebase with chip-tod at the end of enable_capi_mode() we return an error. This has an to unintended consequences. First this will prevent disabling of fast-reboot even though CAPP is already enabled by this point. Secondly, failure during timebase sync is a non fatal error or capp initialization as CAPP/PSL can continue working after this and an AFU will only see an error when it tries to read the timebase value from PSL.

  So this patch updates enable_capi_mode() to not return an error in case call to chiptod_capp_timebase_sync() fails. The function will now just log an error and continue further with capp init sequence. This make the current implementation align with the one in kernel 'cxl' driver which also assumes the PSL timebase sync errors as non-fatal init error.

- npu2-opencapi: Fix assert on link reset during init

  We don't support resetting an opencapi link yet.

  Commit fe6d86b9 ("pci: Make fast reboot creset PHBs in parallel") tries resetting any PHB whose slot defines a 'run_sm' callback. It raises an assert when applied to an opencapi PHB, as 'run_sm' calls the 'freset' callback, which is not yet defined for opencapi.

  Fix it for now by removing the currently useless definition of 'run_sm' on the opencapi slot. It will print a message in the skiboot log because the PHB cannot be reset, which is correct. It will all go away when we add support for resetting an opencapi link.

- capp: Add lid definition for P9 DD-2.2

  Update fsp_lid_map to include CAPP ucode lid for phb4-chipid == 0x202d1 that corresponds to P9 DD-2.2 chip.

- capp: Disable fast-reboot when capp is enabled

PCI

Since *skiboot-5.11-rc1*:

- phb4: Reset FIR/NFIR registers before PHB4 probe

  The function phb4_probe_stack() resets "ETU Reset Register" to unfreeze the PHB before it performs mmio access on the PHB. However in case the FIR/NFIR registers are set while entering this function, the reset of "ETU Reset Register" wont unfreeze the PHB and it will remain fenced. This leads to failure during initial CRESET of the PHB as mmio access is still not enabled and an error message of the form below is logged:

  ```
  PHB#0000[0:0]: Initializing PHB4...
  PHB#0000[0:0]: Default system config: 0xffffffffffffffff
  PHB#0000[0:0]: New system config    : 0xffffffffffffffff
  PHB#0000[0:0]: Initial PHB CRESET is 0xffffffffffffffff
  PHB#0000[0:0]: Waiting for DLP PG reset to complete...
  <snip>
  PHB#0000[0:0]: Timeout waiting for DLP PG reset !
  PHB#0000[0:0]: Initialization failed
  ```

  This is especially seen happening during the MPIPL flow where SBE would quiesces and fence the PHB so that it doesn't stomp on the main memory. However when skiboot enters phb4_probe_stack() after MPIPL, the FIR/NFIR registers are set forcing PHB to re-enter fence after ETU reset is done.

So to fix this issue the patch introduces new xscom writes to phb4_probe_stack() to reset the FIR/NFIR registers before performing ETU reset to enable mmio access to the PHB.

Since *skiboot-5.10*:

- pci: Reduce log level of error message

  If a link doesn't train, we can end up with error messages like this:

  ```
  [   63.027261959,3] PHB#0032[8:2]: LINK: Timeout waiting for electrical link
  [   63.027265573,3] PHB#0032:00:00.0 Error -6 resetting
  ```

  The first message is useful but the second message is just debug from the core PCI code and is confusing to print to the console.

  This reduces the second print to debug level so it's not seen by the console by default.

- Revert "platforms/astbmc/slots.c: Allow comparison of bus numbers when matching slots"

  This reverts commit bda7cc4d0354eb3f66629d410b2afc08c79f795f.

  Ben says: It's on purpose that we do NOT compare the bus numbers, they are always 0 in the slot table we do a hierarchical walk of the tree, matching only the devfn's along the way bcs the bus numbering isn't fixed this breaks all slot naming etc... stuff on anything using the "skiboot" slot tables (P8 opp typically)

- core/pci-dt-slot: Fix booting with no slot map

  Currently if you don't have a slot map in the device tree in /ibm,pcie-slots, you can crash with a back trace like this:

  ```
  CPU 0034 Backtrace:
   S: 0000000031cd3370 R: 000000003001362c   .backtrace+0x48
   S: 0000000031cd3410 R: 0000000030019e38   ._abort+0x4c
   S: 0000000031cd3490 R: 000000003002760c   .exception_entry+0x180
   S: 0000000031cd3670 R: 0000000000001f10 *
   S: 0000000031cd3850 R: 00000000300b4f3e * cpu_features_table+0x1d9e
   S: 0000000031cd38e0 R: 000000003002682c   .dt_node_is_compatible+0x20
   S: 0000000031cd3960 R: 0000000030030e08   .map_pci_dev_to_slot+0x16c
   S: 0000000031cd3a30 R: 0000000030091054   .dt_slot_get_slot_info+0x28
   S: 0000000031cd3ac0 R: 000000003001e27c   .pci_scan_one+0x2ac
   S: 0000000031cd3ba0 R: 000000003001e588   .pci_scan_bus+0x70
   S: 0000000031cd3cb0 R: 000000003001ee74   .pci_scan_phb+0x100
   S: 0000000031cd3d40 R: 0000000030017ff0   .cpu_process_jobs+0xdc
   S: 0000000031cd3e00 R: 0000000030014cb0   .__secondary_cpu_entry+0x44
   S: 0000000031cd3e80 R: 0000000030014d04   .secondary_cpu_entry+0x34
   S: 0000000031cd3f00 R: 0000000030002770   secondary_wait+0x8c
  [   73.016947149,3] Fatal MCE at 0000000030026054   .dt_find_property+0x30
  [   73.017073254,3] CFAR : 0000000030026040
  [   73.017138048,3] SRR0 : 0000000030026054 SRR1 : 9000000000201000
  [   73.017198375,3] HSRR0: 0000000000000000 HSRR1: 0000000000000000
  [   73.017263210,3] DSISR: 00000008         DAR  : 7c7b1b7848002524
  [   73.017352517,3] LR   : 000000003002602c CTR  : 000000003009102c
  [   73.017419778,3] CR   : 20004204         XER  : 20040000
  [   73.017502425,3] GPR00: 000000003002682c GPR16: 0000000000000000
  [   73.017586924,3] GPR01: 0000000031c23670 GPR17: 0000000000000000
  [   73.017643873,3] GPR02: 00000000300fd500 GPR18: 0000000000000000
  [   73.017767091,3] GPR03: fffffffffffffff8 GPR19: 0000000000000000
  [   73.017855707,3] GPR04: 00000000300b3dc6 GPR20: 0000000000000000
  [   73.017943944,3] GPR05: 0000000000000000 GPR21: 00000000300bb6d2
  [   73.018024709,3] GPR06: 0000000031c23910 GPR22: 0000000000000000
  [   73.018117716,3] GPR07: 0000000031c23930 GPR23: 0000000000000000
  ```

(continues on next page)

(continued from previous page)

```
[   73.018195974,3] GPR08: 0000000000000000 GPR24: 0000000000000000
[   73.018278350,3] GPR09: 0000000000000000 GPR25: 0000000000000000
[   73.018353795,3] GPR10: 0000000000000028 GPR26: 00000000300be6fb
[   73.018424362,3] GPR11: 0000000000000000 GPR27: 0000000000000000
[   73.018533159,3] GPR12: 0000000020004208 GPR28: 0000000030767d38
[   73.018642725,3] GPR13: 0000000031c20000 GPR29: 00000000300b3dc6
[   73.018737925,3] GPR14: 0000000000000000 GPR30: 0000000000000010
[   73.018794428,3] GPR15: 0000000000000000 GPR31: 7c7b1b7848002514
```

This has been seen in the lab on a witherspoon using the device tree entry point (ie. no HDAT).

This fixes the null pointer deref.

Bugs Fixed

Since *skiboot-5.11-rc1*:

- cpufeatures: Fix setting DARN and SCV HWCAP feature bits

  DARN and SCV has been assigned AT_HWCAP2 (32-63) bits:

  ```
  #define PPC_FEATURE2_DARN                0x00200000 /* darn random number insn */
  #define PPC_FEATURE2_SCV                 0x00100000 /* scv syscall */
  ```

  A cpufeatures-aware OS will not advertise these to userspace without this patch.

- xive: disable store EOI support

  Hardware has limitations which would require to put a sync after each store EOI to make sure the MMIO operations that change the ESB state are ordered. This is a killer for performance and the PHBs do not support the sync. So remove the store EOI for the moment, until hardware is improved.

  Also, while we are at changing the XIVE source flags, let's fix the settings for the PHB4s which should follow these rules :

  – SHIFT_BUG for DD10
  – STORE_EOI for DD20 and if enabled
  – TRIGGER_PAGE for DDx0 and if not STORE_EOI

Since *skiboot-5.10*:

- xive: fix opal_xive_set_vp_info() error path

  In case of error, opal_xive_set_vp_info() will return without unlocking the xive object. This is most certainly a typo.

- hw/imc: don't access homer memory if it was not initialised

  This can happen under mambo, at least.

- nvram: run nvram_validate() after nvram_reformat()

  nvram_reformat() sets nvram_valid = true, but it does not set skiboot_part_hdr. Call nvram_validate() instead, which sets everything up properly.

- dts: Zero struct to avoid using uninitialised value
- hw/imc: Don't dereference possible NULL
- libstb/create-container: munmap() signature file address

- npu2-opencapi: Fix memory leak
- npu2: Fix possible NULL dereference
- occ-sensors: Remove NULL checks after dereference
- core/ipmi-opal: Add interrupt-parent property for ipmi node on P9 and above.

  dtc complains below warning with newer 4.2+ kernels.

  ```
  dts: Warning (interrupts_property): Missing interrupt-parent for /ibm,opal/ipmi
  ```

  This fix adds interrupt-parent property under /ibm,opal/ipmi DT node on P9 and above, which allows ipmi-opal to properly use the OPAL irqchip.

Other fixes and improvements

- core/cpu: discover stack region size before initialising memory regions

  Stack allocation first allocates a memory region sized to hold stacks for all possible CPUs up to the maximum PIR of the architecture, zeros the region, then initialises all stacks. Max PIR is 32768 on POWER9, which is 512MB for stacks.

  The stack region is then shrunk after CPUs are discovered, but this is a bit of a hack, and it leaves a hole in the memory allocation regions as it's done after mem regions are initialised.

  ```
  0x000000000000..00002fffffff : ibm,os-reserve - OS
  0x000030000000..0000303fffff : ibm,firmware-code - OPAL
  0x000030400000..000030ffffff : ibm,firmware-heap - OPAL
  0x000031000000..000031bfffff : ibm,firmware-data - OPAL
  0x000031c00000..000031c0ffff : ibm,firmware-stacks - OPAL
  *** gap ***
  0x000051c00000..000051d01fff : ibm,firmware-allocs-memory@0 - OPAL
  0x000051d02000..00007fffffff : ibm,firmware-allocs-memory@0 - OS
  0x000080000000..000080b3cdff : initramfs - OPAL
  0x000080b3ce00..000080b7cdff : ibm,fake-nvram - OPAL
  0x000080b7ce00..0000ffffffff : ibm,firmware-allocs-memory@0 - OS
  ```

  This change moves zeroing into the per-cpu stack setup. The boot CPU stack is set up based on the current PIR. Then the size of the stack region is set, by discovering the maximum PIR of the system from the device tree, before mem regions are intialised.

  This results in all memory being accounted within memory regions, and less memory fragmentation of OPAL allocations.

- Make gard display show that a record is cleared

  When clearing gard records, Hostboot only modifies the record_id portion to be 0xFFFFFFFF. The remainder of the entry remains. Without this change it can be confusing to users to know that the record they are looking at is no longer valid.

- Reserve OPAL API number for opal_handle_hmi2 function.
- dts: spl_wakeup: Remove all workarounds in the spl wakeup logic

  We coded few workarounds in special wakeup logic to handle the buggy firmware. Now that is fixed remove them as they break the special wakeup protocol. As per the spec we should not de-assert beofre assert is complete. So follow this protocol.

- build: use thin archives rather than incremental linking

This changes to build system to use thin archives rather than incremental linking for built-in.o, similar to recent change to Linux. built-in.o is renamed to built-in.a, and is created as a thin archive with no index, for speed and size. All built-in.a are aggregated into a skiboot.tmp.a which is a thin archive built with an index, making it suitable or linking. This is input into the final link.

The advantags of build size and linker code placement flexibility are not as great with skiboot as a bigger project like Linux, but it's a conceptually better way to build, and is more compatible with link time optimisation in toolchains which might be interesting for skiboot particularly for size reductions.

Size of build tree before this patch is 34.4MB, afterwards 23.1MB.

- core/init: Assert when kernel not found

  If the kernel doesn't load out of flash or there is nothing at KERNEL_LOAD_BASE, we end up with an esoteric message as we try to branch to out of skiboot into nothing

```
[    0.007197688,3] INIT: ELF header not found. Assuming raw binary.
[    0.014035267,5] INIT: Starting kernel at 0x0, fdt at 0x3044ad90 13029
[    0.014042254,3] ***********************************************
[    0.014069947,3] Fatal Exception 0xe40 at 0000000000000000
[    0.014085574,3] CFAR : 00000000300051c4
[    0.014090118,3] SRR0 : 0000000000000000 SRR1 : 0000000000000000
[    0.014096243,3] HSRR0: 0000000000000000 HSRR1: 9000000000001000
[    0.014102546,3] DSISR: 00000000         DAR  : 0000000000000000
[    0.014108538,3] LR   : 00000000300144c8 CTR  : 0000000000000000
[    0.014114756,3] CR   : 40002202         XER  : 00000000
[    0.014120301,3] GPR00: 000000003001447c GPR16: 0000000000000000
```

  This improves the message and asserts in this case:

```
[    0.014042685,5] INIT: Starting kernel at 0x0, fdt at 0x3044ad90 13049 bytes)
[    0.014049556,0] FATAL: Kernel is zeros, can't execute!
[    0.014054237,0] Assert fail: core/init.c:566:0
[    0.014060472,0] Aborting!
```

- core: Fix 'opal-runtime-size' property

  We are populating 'opal-runtime-size' before calculating actual stack size. Hence we endup having wrong runtime size (ex: on P9 it shows ~540MB while actual size is around ~40MB). Note that only device tree property is shows wrong value, but reserved-memory reflects correct size.

  init_all_cpus() calculates and updates actual stack size. Hence move this function call before add_opal_node().

- mambo: Add fw-feature flags for security related settings

  Newer firmwares report some feature flags related to security settings via HDAT. On real hardware skiboot translates these into device tree properties. For testing purposes just create the properties manually in the tcl.

  These values don't exactly match any actual chip revision, but the code should not rely on any exact set of values anyway. We just define the most interesting flags, that if toggled to "disable" will change Linux behaviour. You can see the actual values in the hostboot source in src/usr/hdat/hdatiplparms.H.

  Also add an environment variable for easily toggling the top-level "security on" setting.

- direct-controls: mambo fix for multiple chips
- libflash/blocklevel: Correct miscalculation in blocklevel_smart_erase()

  If blocklevel_smart_erase() detects that the smart erase fits entire in one erase block, it has an early bail path. In this path it miscaculates where in the buffer the backend needs to read from to perform the final write.

- libstb/secureboot: Fix logging of secure verify messages.

Currently we are logging secure verify/enforce messages in PR_EMERG level even when there is no secureboot mode enabled. So reduce the log level to PR_ERR when secureboot mode is OFF.

Testing / Code coverage improvements

Improvements in gcov support include support for newer GCCs as well as easily exporting the area of memory you need to dump to feed to *extract-gcov*.

- cpu_idle_job: relax a bit

  This *dramatically* improves kernel boot time with GCOV builds

  from ~3minutes between loading kernel and switching the HILE bit down to around 10 seconds.

- gcov: Another GCC, another gcov tweak
- Keep constructors with priorities

  Fixes GCOV builds with gcc7, which uses this.

- gcov: Add gcov data struct to sysfs

  Extracting the skiboot gcov data is currently a tedious process which involves taking a mem dump of skiboot and searching for the gcov_info struct. This patch adds the gcov struct to sysfs under /opal/exports. Allowing the data to be copied directly into userspace and processed.

5.1.41 skiboot-5.11-rc1

skiboot v5.11-rc1 was released on Wednesday March 28th 2018. It is the first release candidate of skiboot 5.11, which will become the new stable release of skiboot following the 5.10 release, first released February 23rd 2018.

It is not expected to keep the 5.11 branch around for long, and instead quickly move onto a 6.0, which will mark the basis for op-build v2.0 and will be required for POWER9 systems.

skiboot v5.11-rc1 contains all bug fixes as of *skiboot-5.10.3* and *skiboot-5.4.9* (the currently maintained stable releases). There may be more 5.10.x stable releases, it will depend on demand.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.11 in March, with skiboot 5.11 being for all POWER8 and POWER9 platforms in op-build v1.22. This release is targeted to early POWER9 systems.

Over skiboot-5.10, we have the following changes:

New Platforms

- Add VESNIN platform support

  The Vesnin platform from YADRO is a 4 socked POWER8 system with up to 8TB of memory with 460GB/s of memory bandwidth in only 2U. Many kudos to the team from Yadro for submitting their code upstream!

New Features

- fast-reboot: enable by default for POWER9
  - Fast reboot is disabled if NPU2 is present or CAPI2/OpenCAPI is used
- PCI tunneled operations on PHB4

  - phb4: set PBCQ Tunnel BAR for tunneled operations

    P9 supports PCI tunneled operations (atomics and as_notify) that are initiated by devices.

    A subset of the tunneled operations require a response, that must be sent back from the host to the device. For example, an atomic compare and swap will return the compare status, as swap will only performed in case of success. Similarly, as_notify reports if the target thread has been woken up or not, because the operation may fail.

    To enable tunneled operations, a device driver must tell the host where it expects tunneled operation responses, by setting the PBCQ Tunnel BAR Response register with a specific value within the range of its BARs.

    This register is currently initialized by enable_capi_mode(). But, as tunneled operations may also operate in PCI mode, a new API is required to set the PBCQ Tunnel BAR Response register, without switching to CAPI mode.

    This patch provides two new OPAL calls to get/set the PBCQ Tunnel BAR Response register.

    Note: as there is only one PBCQ Tunnel BAR register, shared between all the devices connected to the same PHB, only one of these devices will be able to use tunneled operations, at any time.

  - phb4: set PHB CMPM registers for tunneled operations

    P9 supports PCI tunneled operations (atomics and as_notify) that require setting the PHB ASN Compare/Mask register with a 16-bit indication.

    This register is currently initialized by enable_capi_mode(). But, as tunneled operations may also work in PCI mode, the ASN Compare/Mask register should rather be initialized in phb4_init_ioda3().

    This patch also adds "ibm,phb-indications" to the device tree, to tell Linux the values of CAPI, ASN, and NBW indications, when supported.

    Tunneled operations tested by IBM in CAPI mode, by Mellanox Technologies in PCI mode.

- Tie tm-suspend fw-feature and opal_reinit_cpus() together

  Currently opal_reinit_cpus(OPAL_REINIT_CPUS_TM_SUSPEND_DISABLED) always returns OPAL_UNSUPPORTED.

  This ties the tm suspend fw-feature to the opal_reinit_cpus(OPAL_REINIT_CPUS_TM_SUSPEND_DISABLED) so that when tm suspend is disabled, we correctly report it to the kernel. For backwards compatibility, it's assumed tm suspend is available if the fw-feature is not present.

  Currently hostboot will clear fw-feature(TM_SUSPEND_ENABLED) on P9N DD2.1. P9N DD2.2 will set fw-feature(TM_SUSPEND_ENABLED). DD2.0 and below has TM disabled completely (not just suspend).

  We are using opal_reinit_cpus() to determine this setting (rather than the device tree/HDAT) as some future firmware may let us change this dynamically after boot. That is not the case currently though.

Power Management

- SLW: Increase stop4-5 residency by 10x

  Using DGEMM benchmark we observed there was a drop of 5-9% throughput with and without stop4/5. In this benchmark the GPU waits on the cpu to wakeup and provide the subsequent data block to compute. The wakup latency accumulates over the run and shows up as a performance drop.

  Linux enters stop4/5 more aggressively for its wakeup latency. Increasing the residency from 1ms to 10ms makes the performance drop <1%

- occ: Set up OCC messaging even if we fail to setup pstates

  This means that we no longer hit this bug if we fail to get valid pstates from the OCC.

```
[console-pexpect]#echo 1 > //sys/firmware/opal/sensor_groups//occ-csm0/clear
echo 1 > //sys/firmware/opal/sensor_groups//occ-csm0/clear
[   94.019971181,5] CPU ATTEMPT TO RE-ENTER FIRMWARE! PIR=083d cpu @0x33cf4000 ->␣
↪pir=083d token=8
[   94.020098392,5] CPU ATTEMPT TO RE-ENTER FIRMWARE! PIR=083d cpu @0x33cf4000 ->␣
↪pir=083d token=8
[   10.318805] Disabling lock debugging due to kernel taint
[   10.318808] Severe Machine check interrupt [Not recovered]
[   10.318812]   NIP [000000003003e434]: 0x3003e434
[   10.318813]   Initiator: CPU
[   10.318815]   Error type: Real address [Load/Store (foreign)]
[   10.318817] opal: Hardware platform error: Unrecoverable Machine Check␣
↪exception
[   10.318821] CPU: 117 PID: 2745 Comm: sh Tainted: G   M             4.15.9-
↪openpower1 #3
[   10.318823] NIP:  000000003003e434 LR: 000000003003025c CTR: 0000000030030240
[   10.318825] REGS: c00000003fa7bd80 TRAP: 0200   Tainted: G   M              (4.
↪15.9-openpower1)
[   10.318826] MSR:  9000000000201002 <SF,HV,ME,RI>  CR: 48002888  XER: 20040000
[   10.318831] CFAR: 0000000030030258 DAR: 394a00147d5a03a6 DSISR: 00000008␣
↪SOFTE: 1
```

mbox based platforms

For platforms using the mbox protocol for host flash access (all BMC based OpenPOWER systems, most OpenBMC based systems) there have been some hardening efforts in the event of the BMC being poorly behaved.

- mbox: Reduce default BMC timeouts

  Rebooting a BMC can take 70 seconds. Skiboot cannot possibly spin for 70 seconds waiting for a BMC to come back. This also makes the current default of 30 seconds a bit pointless, is it far too short to be a worse case wait time but too long to avoid hitting hardlockup detectors and wrecking havoc inside host linux.

  Just change it to three seconds so that host linux will survive and that, reads and writes will fail but at least the host stays up.

  Also refactored the waiting loop just a bit so that it's easier to read.

- mbox: Harden against BMC daemon errors

  Bugs present in the BMC daemon mean that skiboot gets presented with mbox windows of size zero. These windows cannot be valid and skiboot already detects these conditions.

  Currently skiboot warns quite strongly about the occurrence of these problems. The problem for skiboot is that it doesn't take any action. Initially I wanting to avoid putting policy like this into skiboot but since these bugs aren't going away and skiboot barfing is leading to lockups and ultimately the host going down something needs to be done.

  I propose that when we detect the problem we fail the mbox call and punt the problem back up to Linux. I don't like it but at least it will cause errors to cascade and won't bring the host down. I'm not sure how Linux is supposed to detect this or what it can even do but this is better than a crash.

  Diagnosing a failure to boot if skiboot its self fails to read flash may be marginally more difficult with this patch. This is because skiboot will now only print one warning about the zero sized window rather than continuously spitting it out.

Fast Reboot Improvements

Around fast-reboot we have made several improvements to harden the fast reboot code paths and resort to a full IPL if something doesn't look right.

- core/fast-reboot: zero memory after fast reboot

  This improves the security and predictability of the fast reboot environment.

  There can not be a secure fence between fast reboots, because a malicious OS can modify the firmware itself. However a well-behaved OS can have a reasonable expectation that OS memory regions it has modified will be cleared upon fast reboot.

  The memory is zeroed after all other CPUs come up from fast reboot, just before the new kernel is loaded and booted into. This allows image preloading to run concurrently, and will allow parallelisation of the clearing in future.

- core/fast-reboot: verify mem regions before fast reboot

  Run the mem_region sanity checkers before proceeding with fast reboot.

  This is the beginning of proactive sanity checks on opal data for fast reboot (with complements the reactive disable_fast_reboot cases). This is encouraged to re-use and share any kind of debug code and unit test code.

- fast-reboot: occ: Only delete /ibm, opal/power-mgt nodes if they exist
- core/fast-reboot: disable fast reboot upon fundamental entry/exit/locking errors

  This disables fast reboot in several more cases where serious errors like lock corruption or call re-entrancy are detected.

- capp: Disable fast-reboot whenever enable_capi_mode() is called

  This patch updates phb4_set_capi_mode() to disable fast-reboot whenever enable_capi_mode() is called, irrespective to its return value. This should prevent against a possibility of not disabling fast-reboot when some changes to enable_capi_mode() causing return of an error and leaving CAPP in enabled mode.

- fast-reboot: occ: Delete OCC child nodes in /ibm, opal/power-mgt

  Fast-reboot in P8 fails to re-init OCC data as there are chipwise OCC nodes which are already present in the /ibm,opal/power-mgt node. These per-chip nodes hold the voltage IDs for each pstate and these can be changed on OCC pstate table biasing. So delete these before calling the re-init code to re-parse and populate the pstate data.

Debugging/SRESET improvemens

- core/opal: allow some re-entrant calls

  This allows a small number of OPAL calls to succeed despite re-entering the firmware, and rejects others rather than aborting.

  This allows a system reset interrupt that interrupts OPAL to do something useful. Sreset other CPUs, use the console, which allows xmon to work or stack traces to be printed, reboot the system.

  Use OPAL_INTERNAL_ERROR when rejecting, rather than OPAL_BUSY, which is used for many other things that does not mean a serious permanent error.

- core/opal: abort in case of re-entrant OPAL call

  The stack is already destroyed by the time we get here, so there is not much point continuing.

- core/lock: Add lock timeout warnings

  There are currently no timeout warnings for locks in skiboot. We assume that the lock will eventually become free, which may not always be the case.

  This patch adds timeout warnings for locks. Any lock which spins for more than 5 seconds will throw a warning and stacktrace for that thread. This is useful for debugging siturations where a lock which hang, waiting for the lock to be freed.

- core/lock: Add deadlock detection

  This adds simple deadlock detection. The detection looks for circular dependencies in the lock requests. It will abort and display a stack trace when a deadlock occurs. The detection is enabled by DEBUG_LOCKS (enabled by default). While the detection may have a slight performance overhead, as there are not a huge number of locks in skiboot this overhead isn't significant.

- core/hmi: report processor recovery reason from core FIR bits on P9

  When an error is encountered that causes processor recovery, HMI is generated if the recovery was successful. The reason is recorded in the core FIR, which gets copied into the WOF.

  In this case dump the WOF register and an error string into the OPAL msglog.

  A broken init setting led to HMIs reported in Linux as:

  ```
  [    3.591547] Harmless Hypervisor Maintenance interrupt [Recovered]
  [    3.591648]  Error detail: Processor Recovery done
  [    3.591714]  HMER: 2040000000000000
  ```

  This patch would have been useful because it tells us exactly that the problem is in the d-side ERAT:

  ```
  [  414.489690798,7] HMI: Received HMI interrupt: HMER = 0x2040000000000000
  [  414.489693339,7] HMI: [Loc: UOPWR.0000000-Node0-Proc0]: P:0 C:1 T:1: Processor␣
  ↪recovery occurred.
  [  414.489699837,7] HMI: Core WOF = 0x0000000410000000 recovered error:
  [  414.489701543,7] HMI: LSU - SRAM (DCACHE parity, etc)
  [  414.489702341,7] HMI: LSU - ERAT multi hit
  ```

  In future it will be good to unify this reporting, so Linux could print something more useful. Until then, this gives some good data.

NPU2/NVLink2 Fixes

- npu2: Add performance tuning SCOM inits

  Peer-to-peer GPU bandwidth latency testing has produced some tunable values that improve performance. Add them to our device initialization.

  File these under things that need to be cleaned up with nice #defines for the register names and bitfields when we get time.

  A few of the settings are dependent on the system's particular NVLink topology, so introduce a helper to determine how many links go to a single GPU.

- hw/npu2: Assign a unique LPARSHORTID per GPU

  This gets used elsewhere to index items in the XTS tables.

- NPU2: dump NPU2 registers on npu2 HMI

  Due to the nature of debugging npu2 issues, folk are wanting the full list of NPU2 registers dumped when there's a problem.

- npu2: Remove DD1 support

  Major changes in the NPU between DD1 and DD2 necessitated a fair bit of revision-specific code.

  Now that all our lab machines are DD2, we no longer test anything on DD1 and it's time to get rid of it.

  Remove DD1-specific code and abort probe if we're running on a DD1 machine.

- npu2: Disable fast reboot

  Fast reboot does not yet work right with the NPU. It's been disabled on NVLink and OpenCAPI machines. Do the same for NVLink2.

  This amounts to a port of 3e4577939bbf ("npu: Fix broken fast reset") from the npu code to npu2.

- npu2: Use unfiltered mode in XTS tables

  The XTS_PID context table is limited to 256 possible pids/contexts. To relieve this limitation, make use of "unfiltered mode" instead.

  If an entry in the XTS_BDF table has the bit for unfiltered mode set, we can just use one context for that entire bdf/lpar, regardless of pid. Instead of of searching the XTS_PID table, the NMMU checkout request will simply use the entry indexed by lparshort id instead.

  Change opal_npu_init_context() to create these lparshort-indexed wildcard entries (0-15) instead of allocating one for each pid. Check that multiple calls for the same bdf all specify the same msr value.

  In opal_npu_destroy_context(), continue validating the bdf argument, ensuring that it actually maps to an lpar, but no longer remove anything from the XTS_PID table. If/when we start supporting virtualized GPUs, we might consider actually removing these wildcard entries by keeping a refcount, but keep things simple for now.

CAPI/OpenCAPI

- npu2-opencapi: Add OpenCAPI OPAL API calls

  Add three OPAL API calls that are required by the ocxl driver.

  - OPAL_NPU_SPA_SETUP

    The Shared Process Area (SPA) is a table containing one entry (a "Process Element") per memory context which can be accessed by the OpenCAPI device.

  - OPAL_NPU_SPA_CLEAR_CACHE

    The NPU keeps a cache of recently accessed memory contexts. When a Process Element is removed from the SPA, the cache for the link must be cleared.

  - OPAL_NPU_TL_SET

    The Transaction Layer specification defines several templates for messages to be exchanged on the link. During link setup, the host and device must negotiate what templates are supported on both sides and at what rates those messages can be sent.

- npu2-opencapi: Train OpenCAPI links and setup devices

  Scan the OpenCAPI links under the NPU, and for each link, reset the card, set up a device, train the link and register a PHB.

  Implement the necessary operations for the OpenCAPI PHB type.

  For bringup, test and debug purposes, we allow an NVRAM setting, "opencapi-link-training" that can be set to either disable link training completely or to use the prbs31 test pattern.

  To disable link training:

```
nvram -p ibm,skiboot --update-config opencapi-link-training=none
```

To use prbs31:

```
nvram -p ibm,skiboot --update-config opencapi-link-training=prbs31
```

- npu2-hw-procedures: Add support for OpenCAPI PHY link training

  Unlike NVLink, which uses the pci-virt framework to fake a PCI configuration space for NVLink devices, the OpenCAPI device model presents us with a real configuration space handled by the device over the OpenCAPI link.

  As a result, we have to train the OpenCAPI link in skiboot before we do PCI probing, so that config space can be accessed, rather than having link training being triggered by the Linux driver.

- npu2-opencapi: Configure NPU for OpenCAPI

  Scan the device tree for NPUs with OpenCAPI links and configure the NPU per the initialisation sequence in the NPU OpenCAPI workbook.

- capp: Make error in capp timebase sync a non-fatal error

  Presently when we encounter an error while synchronizing capp timebase with chip-tod at the end of enable_capi_mode() we return an error. This has an to unintended consequences. First this will prevent disabling of fast-reboot even though CAPP is already enabled by this point. Secondly, failure during timebase sync is a non fatal error or capp initialization as CAPP/PSL can continue working after this and an AFU will only see an error when it tries to read the timebase value from PSL.

  So this patch updates enable_capi_mode() to not return an error in case call to chiptod_capp_timebase_sync() fails. The function will now just log an error and continue further with capp init sequence. This make the current implementation align with the one in kernel 'cxl' driver which also assumes the PSL timebase sync errors as non-fatal init error.

- npu2-opencapi: Fix assert on link reset during init

  We don't support resetting an opencapi link yet.

  Commit fe6d86b9 ("pci: Make fast reboot creset PHBs in parallel") tries resetting any PHB whose slot defines a 'run_sm' callback. It raises an assert when applied to an opencapi PHB, as 'run_sm' calls the 'freset' callback, which is not yet defined for opencapi.

  Fix it for now by removing the currently useless definition of 'run_sm' on the opencapi slot. It will print a message in the skiboot log because the PHB cannot be reset, which is correct. It will all go away when we add support for resetting an opencapi link.

- capp: Add lid definition for P9 DD-2.2

  Update fsp_lid_map to include CAPP ucode lid for phb4-chipid == 0x202d1 that corresponds to P9 DD-2.2 chip.

- capp: Disable fast-reboot when capp is enabled

PCI

- pci: Reduce log level of error message

  If a link doesn't train, we can end up with error messages like this:

```
[   63.027261959,3] PHB#0032[8:2]: LINK: Timeout waiting for electrical link
[   63.027265573,3] PHB#0032:00:00.0 Error -6 resetting
```

The first message is useful but the second message is just debug from the core PCI code and is confusing to print to the console.

This reduces the second print to debug level so it's not seen by the console by default.

- Revert "platforms/astbmc/slots.c: Allow comparison of bus numbers when matching slots"

  This reverts commit bda7cc4d0354eb3f66629d410b2afc08c79f795f.

  Ben says: It's on purpose that we do NOT compare the bus numbers, they are always 0 in the slot table we do a hierarchical walk of the tree, matching only the devfn's along the way bcs the bus numbering isn't fixed this breaks all slot naming etc… stuff on anything using the "skiboot" slot tables (P8 opp typically)

- core/pci-dt-slot: Fix booting with no slot map

  Currently if you don't have a slot map in the device tree in /ibm,pcie-slots, you can crash with a back trace like this:

```
CPU 0034 Backtrace:
 S: 0000000031cd3370 R: 000000003001362c   .backtrace+0x48
 S: 0000000031cd3410 R: 0000000030019e38   ._abort+0x4c
 S: 0000000031cd3490 R: 000000003002760c   .exception_entry+0x180
 S: 0000000031cd3670 R: 0000000000001f10 *
 S: 0000000031cd3850 R: 00000000300b4f3e * cpu_features_table+0x1d9e
 S: 0000000031cd38e0 R: 000000003002682c   .dt_node_is_compatible+0x20
 S: 0000000031cd3960 R: 0000000030030e08   .map_pci_dev_to_slot+0x16c
 S: 0000000031cd3a30 R: 0000000030091054   .dt_slot_get_slot_info+0x28
 S: 0000000031cd3ac0 R: 000000003001e27c   .pci_scan_one+0x2ac
 S: 0000000031cd3ba0 R: 000000003001e588   .pci_scan_bus+0x70
 S: 0000000031cd3cb0 R: 000000003001ee74   .pci_scan_phb+0x100
 S: 0000000031cd3d40 R: 0000000030017ff0   .cpu_process_jobs+0xdc
 S: 0000000031cd3e00 R: 0000000030014cb0   .__secondary_cpu_entry+0x44
 S: 0000000031cd3e80 R: 0000000030014d04   .secondary_cpu_entry+0x34
 S: 0000000031cd3f00 R: 0000000030002770   secondary_wait+0x8c
[   73.016947149,3] Fatal MCE at 0000000030026054   .dt_find_property+0x30
[   73.017073254,3] CFAR : 0000000030026040
[   73.017138048,3] SRR0 : 0000000030026054 SRR1 : 9000000000201000
[   73.017198375,3] HSRR0: 0000000000000000 HSRR1: 0000000000000000
[   73.017263210,3] DSISR: 00000008         DAR  : 7c7b1b7848002524
[   73.017352517,3] LR   : 000000003002602c CTR  : 000000003009102c
[   73.017419778,3] CR   : 20004204         XER  : 20040000
[   73.017502425,3] GPR00: 000000003002682c GPR16: 0000000000000000
[   73.017586924,3] GPR01: 0000000031c23670 GPR17: 0000000000000000
[   73.017643873,3] GPR02: 00000000300fd500 GPR18: 0000000000000000
[   73.017767091,3] GPR03: fffffffffffffff8 GPR19: 0000000000000000
[   73.017855707,3] GPR04: 00000000300b3dc6 GPR20: 0000000000000000
[   73.017943944,3] GPR05: 0000000000000000 GPR21: 00000000300bb6d2
[   73.018024709,3] GPR06: 0000000031c23910 GPR22: 0000000000000000
[   73.018117716,3] GPR07: 0000000031c23930 GPR23: 0000000000000000
[   73.018195974,3] GPR08: 0000000000000000 GPR24: 0000000000000000
[   73.018278350,3] GPR09: 0000000000000000 GPR25: 0000000000000000
[   73.018353795,3] GPR10: 0000000000000028 GPR26: 00000000300be6fb
[   73.018424362,3] GPR11: 0000000000000000 GPR27: 0000000000000000
[   73.018533159,3] GPR12: 0000000020004208 GPR28: 0000000030767d38
[   73.018642725,3] GPR13: 0000000031c20000 GPR29: 00000000300b3dc6
[   73.018737925,3] GPR14: 0000000000000000 GPR30: 0000000000000010
[   73.018794428,3] GPR15: 0000000000000000 GPR31: 7c7b1b7848002514
```

  This has been seen in the lab on a witherspoon using the device tree entry point (ie. no HDAT).

  This fixes the null pointer deref.

Bugs Fixed

- xive: fix opal_xive_set_vp_info() error path

  In case of error, opal_xive_set_vp_info() will return without unlocking the xive object. This is most certainly a typo.

- hw/imc: don't access homer memory if it was not initialised

  This can happen under mambo, at least.

- nvram: run nvram_validate() after nvram_reformat()

  nvram_reformat() sets nvram_valid = true, but it does not set skiboot_part_hdr. Call nvram_validate() instead, which sets everything up properly.

- dts: Zero struct to avoid using uninitialised value
- hw/imc: Don't dereference possible NULL
- libstb/create-container: munmap() signature file address
- npu2-opencapi: Fix memory leak
- npu2: Fix possible NULL dereference
- occ-sensors: Remove NULL checks after dereference
- core/ipmi-opal: Add interrupt-parent property for ipmi node on P9 and above.

  dtc complains below warning with newer 4.2+ kernels.

  ```
  dts: Warning (interrupts_property): Missing interrupt-parent for /ibm,opal/ipmi
  ```

  This fix adds interrupt-parent property under /ibm,opal/ipmi DT node on P9 and above, which allows ipmi-opal to properly use the OPAL irqchip.

Other fixes and improvements

- core/cpu: discover stack region size before initialising memory regions

  Stack allocation first allocates a memory region sized to hold stacks for all possible CPUs up to the maximum PIR of the architecture, zeros the region, then initialises all stacks. Max PIR is 32768 on POWER9, which is 512MB for stacks.

  The stack region is then shrunk after CPUs are discovered, but this is a bit of a hack, and it leaves a hole in the memory allocation regions as it's done after mem regions are initialised.

  ```
  0x000000000000..00002fffffff : ibm,os-reserve - OS
  0x000030000000..0000303fffff : ibm,firmware-code - OPAL
  0x000030400000..000030ffffff : ibm,firmware-heap - OPAL
  0x000031000000..000031bfffff : ibm,firmware-data - OPAL
  0x000031c00000..000031c0ffff : ibm,firmware-stacks - OPAL
  *** gap ***
  0x000051c00000..000051d01fff : ibm,firmware-allocs-memory@0 - OPAL
  0x000051d02000..00007fffffff : ibm,firmware-allocs-memory@0 - OS
  0x000080000000..000080b3cdff : initramfs - OPAL
  0x000080b3ce00..000080b7cdff : ibm,fake-nvram - OPAL
  0x000080b7ce00..0000ffffffff : ibm,firmware-allocs-memory@0 - OS
  ```

This change moves zeroing into the per-cpu stack setup. The boot CPU stack is set up based on the current PIR. Then the size of the stack region is set, by discovering the maximum PIR of the system from the device tree, before mem regions are intialised.

This results in all memory being accounted within memory regions, and less memory fragmentation of OPAL allocations.

- Make gard display show that a record is cleared

  When clearing gard records, Hostboot only modifies the record_id portion to be 0xFFFFFFFF. The remainder of the entry remains. Without this change it can be confusing to users to know that the record they are looking at is no longer valid.

- Reserve OPAL API number for opal_handle_hmi2 function.

- dts: spl_wakeup: Remove all workarounds in the spl wakeup logic

  We coded few workarounds in special wakeup logic to handle the buggy firmware. Now that is fixed remove them as they break the special wakeup protocol. As per the spec we should not de-assert beofre assert is complete. So follow this protocol.

- build: use thin archives rather than incremental linking

  This changes to build system to use thin archives rather than incremental linking for built-in.o, similar to recent change to Linux. built-in.o is renamed to built-in.a, and is created as a thin archive with no index, for speed and size. All built-in.a are aggregated into a skiboot.tmp.a which is a thin archive built with an index, making it suitable or linking. This is input into the final link.

  The advantags of build size and linker code placement flexibility are not as great with skiboot as a bigger project like Linux, but it's a conceptually better way to build, and is more compatible with link time optimisation in toolchains which might be interesting for skiboot particularly for size reductions.

  Size of build tree before this patch is 34.4MB, afterwards 23.1MB.

- core/init: Assert when kernel not found

  If the kernel doesn't load out of flash or there is nothing at KERNEL_LOAD_BASE, we end up with an esoteric message as we try to branch to out of skiboot into nothing

```
[    0.007197688,3] INIT: ELF header not found. Assuming raw binary.
[    0.014035267,5] INIT: Starting kernel at 0x0, fdt at 0x3044ad90 13029
[    0.014042254,3] ***********************************************
[    0.014069947,3] Fatal Exception 0xe40 at 0000000000000000
[    0.014085574,3] CFAR : 00000000300051c4
[    0.014090118,3] SRR0 : 0000000000000000 SRR1 : 0000000000000000
[    0.014096243,3] HSRR0: 0000000000000000 HSRR1: 9000000000001000
[    0.014102546,3] DSISR: 00000000         DAR  : 0000000000000000
[    0.014108538,3] LR   : 00000000300144c8 CTR  : 0000000000000000
[    0.014114756,3] CR   : 40002202         XER  : 00000000
[    0.014120301,3] GPR00: 000000003001447c GPR16: 0000000000000000
```

  This improves the message and asserts in this case:

```
[    0.014042685,5] INIT: Starting kernel at 0x0, fdt at 0x3044ad90 13049 bytes)
[    0.014049556,0] FATAL: Kernel is zeros, can't execute!
[    0.014054237,0] Assert fail: core/init.c:566:0
[    0.014060472,0] Aborting!
```

- core: Fix 'opal-runtime-size' property

  We are populating 'opal-runtime-size' before calculating actual stack size. Hence we endup having wrong runtime size (ex: on P9 it shows ~540MB while actual size is around ~40MB). Note that only device tree property is shows wrong value, but reserved-memory reflects correct size.

init_all_cpus() calculates and updates actual stack size. Hence move this function call before add_opal_node().

- mambo: Add fw-feature flags for security related settings

  Newer firmwares report some feature flags related to security settings via HDAT. On real hardware skiboot translates these into device tree properties. For testing purposes just create the properties manually in the tcl.

  These values don't exactly match any actual chip revision, but the code should not rely on any exact set of values anyway. We just define the most interesting flags, that if toggled to "disable" will change Linux behaviour. You can see the actual values in the hostboot source in src/usr/hdat/hdatiplparms.H.

  Also add an environment variable for easily toggling the top-level "security on" setting.

- direct-controls: mambo fix for multiple chips
- libflash/blocklevel: Correct miscalculation in blocklevel_smart_erase()

  If blocklevel_smart_erase() detects that the smart erase fits entire in one erase block, it has an early bail path. In this path it miscaculates where in the buffer the backend needs to read from to perform the final write.

- libstb/secureboot: Fix logging of secure verify messages.

  Currently we are logging secure verify/enforce messages in PR_EMERG level even when there is no secureboot mode enabled. So reduce the log level to PR_ERR when secureboot mode is OFF.

Testing / Code coverage improvements

Improvements in gcov support include support for newer GCCs as well as easily exporting the area of memory you need to dump to feed to *extract-gcov*.

- cpu_idle_job: relax a bit

  This *dramatically* improves kernel boot time with GCOV builds

  from ~3minutes between loading kernel and switching the HILE bit down to around 10 seconds.

- gcov: Another GCC, another gcov tweak
- Keep constructors with priorities

  Fixes GCOV builds with gcc7, which uses this.

- gcov: Add gcov data struct to sysfs

  Extracting the skiboot gcov data is currently a tedious process which involves taking a mem dump of skiboot and searching for the gcov_info struct. This patch adds the gcov struct to sysfs under /opal/exports. Allowing the data to be copied directly into userspace and processed.

5.1.42 skiboot-5.2.0

skiboot-5.2.0 was released on Wednesday March 16th, 2016.

skiboot-5.2.0 is the first stable release of skiboot 5.2, the new stable release of skiboot, which will take over from the 5.1.x series which was first released August 17th, 2015.

skiboot-5.2.0 contains all bug fixes as of skiboot-5.1.15.

This is the second release that will follow the (now documented) Skiboot stable rules - see *Skiboot stable tree rules and releases*.

Changes since rc2

Over skiboot-5.2.0-rc2, the following fixes are included:

- Include 'extract-gcov' in make clean.
- ipmi-sel: Fix esel event logger to handle early boot PANIC events
- IPMI: Enable synchronous eSEL logging option (for PANIC events)
- libflash/libffs: Reporting seeing all 0xFF bytes during init.
- ipmi-sel: Fix memory leak in error path

Changes since rc1

Over skiboot-5.2.0-rc1, we have the following changes:

- Add Barreleye platform

Generic

- hw/p8-i2c: Speed up SMBUS_WRITE
- Fix early backtraces

FSP Platforms

- fsp-sensor: rework device tree for sensors
- platforms/firenze: Fix I2C clock source frequency

Simics simulator

- Enable Simics UART console

Mambo simulator

- platforms/mambo: Add terminate callback
  - fix hang in multi-threaded mambo
  - add multithreaded mambo tests

IPMI

- hw/ipmi: fix event data 1 for System Firmware Progress sensor
- ipmi: Log exact NetFn value in OPAL logs

AST BMC based platforms

- hw/bt: allow BT driver to use different buffer size

opal-prd utility

- **opal-prd: Add debug output for firmware-driven OCC events** We indicate when we have a user-driven event, so add corresponding outputs for firmware-driven ones too.

getscom utility

- Add Naples chip support

New Features

Over skiboot-5.1, the following features have been added:

- Naples (P8', i.e. P8 with NVLINK) processor support, including NVLINK.
- Improvements in gard, libflash/pflash and opal-prd utilities
  - increased testing
  - increased usability
  - systemd scripts for opal-prd
  - pflash can now use the /dev/mtd device to access BMC flash rather than accessing it directly. It is *important* that you use –mtd if your BMC may otherwise know how to interact with its own flash.
- support for Micron N25Q256Ax and N25Qx256Ax NOR flash.
- support for Winbond W25Q256BV NOR flash
- support for an emulated ("fake") RTC clock, useful in simulators and during bringup
- Explicit 1:1 mapping in ranges properties have been added to PCI bridges. This allows a neat trick with offb and VGA ports that should probably not be told to young children.
- Added support to read the V2 format of the OCC-OPAL memory region, which supports Workload Optimized Frequency (WOF)

Changes in behavior

- Assigning OPAL IDs to PHBs is now fixed and based on the chip id and PHB index on that chip. On POWER7, we continue to use allocated numbers.
- We now query the BMC for BT capabilities rather than making assumptions

Removed support

- p5ioc2 is no longer supported. This affects a grand total of two POWER7 systems in the world.

NOTE: It is planned that skiboot-5.2 will be the last release supporting POWER7 machines.

Bugs fixed

- PHB3: Fix unexpected ER (all) on errinjct by PCI config
- hw/bt: timeout messages when BT interface isn't functional
- On Habanero, Slot3 should have been "Slot 3".
- We now completely flush the console buffer before power down and reboot
- For chips with ibm,occ-functional-state set to false, we don't wait for the OCC to start. This caused needless delay in booting on simulators which did not simulate OCCs.
- Change OCC reset order to always reset slave OCCs first.
- slw: Remove overwrites for EX_PM_CORE_ECO_VRET and EX_PM_CORE_PFET_VRET (these were already initialized in hostboot)
- p8-i2c: send stop bit on timeouts. Some devices can otherwise leave the bus in a held state.

Other improvements

- many fixes of compiler and static analysis warnings
- increased unit test coverage
- Unit test of "boot debian jessie installer"
- ability to plug in other simulators to run existing tests (e.g. simulator for non pegasus p8)
- Support using (patched) Qemu with PowerNV platform support for running unit tests.
- increased support for running with sparse
- We now build with -fstack-protector-strong if supported by the compiler
- We now build with -Werror for -Wformat
- pflash is now built as part of travis-ci and for Coverity Scan.
- There is now a RPM SPEC file that can be used as the basis for packaging skiboot and associated utilities.

Contributors

We have had a number of improvements in workflow over skiboot-5.1.0. Looking back, we have roughly the same number of changesets (372 for 5.1.0, 334 for 5.2.0-rc1 - even closer for 5.1.0-beta1) which indicates a relatively stable rate of development.

Complete statistics are included below (generated by gitdm), but I'd like to draw attention to a couple of stats:

Release	csets	Ack	Reviews	Tested	Reported
5.0	329	15	20	1	0
5.1	372	13	38	1	4
5.2-rc1	334	20	34	6	11

Overall, it looks like we're on the right trajectory for increasing the number of eyeballs looking at code before it heads in tree, especially around testing. Largely, this increase in Tested-by can be attributed to encouraging the existing test teams to start commenting on the patches themselves.

Anyway, here's the full stats from skiboot 5.1.0 to 5.2.0-rc1:

Processed 334 csets from 27 developers 2 employers found A total of 46172 lines added, 23274 removed (delta 22898)

Developers with the most changesets

Stewart Smith	146 (43.7%)
Cyril Bur	52 (15.6%)
Benjamin Herrenschmidt	15 (4.5%)
Joel Stanley	12 (3.6%)
Gavin Shan	12 (3.6%)
Alistair Popple	10 (3.0%)
Vasant Hegde	10 (3.0%)
Michael Neuling	10 (3.0%)
Russell Currey	9 (2.7%)
Cédric Le Goater	8 (2.4%)
Jeremy Kerr	8 (2.4%)
Samuel Mendoza-Jonas	6 (1.8%)
Neelesh Gupta	6 (1.8%)
Shilpasri G Bhat	4 (1.2%)
Oliver O'Halloran	4 (1.2%)
Mahesh Salgaonkar	4 (1.2%)
Vipin K Parashar	3 (0.9%)
Daniel Axtens	3 (0.9%)
Andrew Donnellan	2 (0.6%)
Philippe Bergheaud	2 (0.6%)
Ananth N Mavinakayanahalli	2 (0.6%)
Vaibhav Jain	1 (0.3%)
Sam Mendoza-Jonas	1 (0.3%)
Adriana Kobylak	1 (0.3%)
Shreyas B. Prabhu	1 (0.3%)
Vaidyanathan Srinivasan	1 (0.3%)
Ian Munsie	1 (0.3%)

Developers with the most changed lines

Stewart Smith	19533 (39.4%)
Oliver O'Halloran	17920 (36.1%)
Alistair Popple	3285 (6.6%)
Daniel Axtens	2154 (4.3%)
Cyril Bur	2028 (4.1%)
Benjamin Herrenschmidt	941 (1.9%)
Neelesh Gupta	434 (0.9%)
Gavin Shan	294 (0.6%)
Russell Currey	261 (0.5%)
Vasant Hegde	245 (0.5%)
Cédric Le Goater	209 (0.4%)
Vipin K Parashar	155 (0.3%)
Shilpasri G Bhat	153 (0.3%)
Joel Stanley	140 (0.3%)
Vaidyanathan Srinivasan	135 (0.3%)
Michael Neuling	111 (0.2%)
Samuel Mendoza-Jonas	81 (0.2%)
Jeremy Kerr	60 (0.1%)
Mahesh Salgaonkar	58 (0.1%)
Vaibhav Jain	50 (0.1%)
Ananth N Mavinakayanahalli	43 (0.1%)
Shreyas B. Prabhu	17 (0.0%)
Sam Mendoza-Jonas	12 (0.0%)
Andrew Donnellan	10 (0.0%)
Ian Munsie	8 (0.0%)
Philippe Bergheaud	6 (0.0%)
Adriana Kobylak	6 (0.0%)

Developers with the most lines removed

Daniel Axtens	2149 (9.2%)
Shreyas B. Prabhu	17 (0.1%)
Andrew Donnellan	9 (0.0%)
Vipin K Parashar	2 (0.0%)

Developers with the most signoffs (total 190)

Stewart Smith	188 (98.9%)
Gavin Shan	1 (0.5%)
Neelesh Gupta	1 (0.5%)

Developers with the most reviews (total 34)

Patrick Williams	5 (14.7%)
Joel Stanley	5 (14.7%)
Cédric Le Goater	5 (14.7%)
Vasant Hegde	4 (11.8%)
Alistair Popple	4 (11.8%)
Sam Mendoza-Jonas	3 (8.8%)
Samuel Mendoza-Jonas	3 (8.8%)
Andrew Donnellan	2 (5.9%)
Cyril Bur	2 (5.9%)
Vaibhav Jain	1 (2.9%)

Developers with the most test credits (total 6)

Vipin K Parashar	3 (50.0%)
Vaibhav Jain	2 (33.3%)
Gajendra B Bandhu1	1 (16.7%)

Developers who gave the most tested-by credits (total 6)

Gavin Shan	2 (33.3%)
Ananth N Mavinakayanahalli	2 (33.3%)
Alistair Popple	1 (16.7%)
Stewart Smith	1 (16.7%)

Developers with the most report credits (total 11)

Vaibhav Jain	2 (18.2%)
Paul Nguyen	2 (18.2%)
Alistair Popple	1 (9.1%)
Cédric Le Goater	1 (9.1%)
Aneesh Kumar K.V	1 (9.1%)
Dionysius d. Bell	1 (9.1%)
Pradeep Ramanna	1 (9.1%)
John Walthour	1 (9.1%)
Benjamin Herrenschmidt	1 (9.1%)

Developers who gave the most report credits (total 11)

Gavin Shan	6 (54.5%)
Stewart Smith	3 (27.3%)
Samuel Mendoza-Jonas	1 (9.1%)
Shilpasri G Bhat	1 (9.1%)

5.1.43 skiboot-5.2.0-rc1

skiboot-5.2.0-rc1 was released on Friday Feb 26th, 2016.

skiboot-5.2.0-rc1 is the first release candidate of skiboot 5.2, which will become the new stable release of skiboot following the 5.1 release, first released August 17th, 2015.

skiboot-5.2.0-rc1 contains all bug fixes as of skiboot-5.1.13.

This is the second release that will follow the (now documented) Skiboot stable rules - see *Skiboot stable tree rules and releases*.

The current plan is to release skiboot-5.2.0 mid-March 2016, with a focus on bug fixing for future 5.2.0-rc releases.

New Features

Over skiboot-5.1, the following features have been added:

- Naples (P8', i.e. P8 with NVLINK) processor support, including NVLINK.
- Improvements in gard, libflash/pflash and opal-prd utilities
  - increased testing
  - increased usability
  - systemd scripts for opal-prd
  - pflash can now use the /dev/mtd device to access BMC flash rather than accessing it directly. It is *important* that you use –mtd if your BMC may otherwise know how to interact with its own flash.
- support for Micron N25Q256Ax and N25Qx256Ax NOR flash.
- support for Winbond W25Q256BV NOR flash
- support for an emulated ("fake") RTC clock, useful in simulators and during bringup
- Explicit 1:1 mapping in ranges properties have been added to PCI bridges. This allows a neat trick with offb and VGA ports that should probably not be told to young children.
- Added support to read the V2 format of the OCC-OPAL memory region, which supports Workload Optimized Frequency (WOF)

Changes in behavior

- Assigning OPAL IDs to PHBs is now fixed and based on the chip id and PHB index on that chip. On POWER7, we continue to use allocated numbers.
- We now query the BMC for BT capabilities rather than making assumptions

Removed support

- p5ioc2 is no longer supported. This affects a grand total of two POWER7 systems in the world.

NOTE: It is planned that skiboot-5.2 will be the last release supporting POWER7 machines.

Bugs fixed

- PHB3: Fix unexpected ER (all) on errinjct by PCI config
- hw/bt: timeout messages when BT interface isn't functional
- On Habanero, Slot3 should have been "Slot 3".
- We now completely flush the console buffer before power down and reboot
- For chips with ibm,occ-functional-state set to false, we don't wait for the OCC to start. This caused needless delay in booting on simulators which did not simulate OCCs.
- Change OCC reset order to always reset slave OCCs first.
- slw: Remove overwrites for EX_PM_CORE_ECO_VRET and EX_PM_CORE_PFET_VRET (these were already initialized in hostboot)
- p8-i2c: send stop bit on timeouts. Some devices can otherwise leave the bus in a held state.

Other improvements

- many fixes of compiler and static analysis warnings
- increased unit test coverage
- Unit test of "boot debian jessie installer"
- ability to plug in other simulators to run existing tests (e.g. simulator for non pegasus p8)
- Support using (patched) Qemu with PowerNV platform support for running unit tests.
- increased support for running with sparse
- We now build with -fstack-protector-strong if supported by the compiler
- We now build with -Werror for -Wformat
- pflash is now built as part of travis-ci and for Coverity Scan.
- There is now a RPM SPEC file that can be used as the basis for packaging skiboot and associated utilities.

Contributors

We have had a number of improvements in workflow over skiboot-5.1.0. Looking back, we have roughly the same number of changesets (372 for 5.1.0, 334 for 5.2.0-rc1 - even closer for 5.1.0-beta1) which indicates a relatively stable rate of development.

Complete statistics are included below (generated by gitdm), but I'd like to draw attention to a couple of stats:

Release	csets	Ack	Reviews	Tested	Reported
5.0	329	15	20	1	0
5.1	372	13	38	1	4
5.2-rc1	334	20	34	6	11

Overall, it looks like we're on the right trajectory for increasing the number of eyeballs looking at code before it heads in tree, especially around testing. Largely, this increase in Tested-by can be attributed to encouraging the existing test teams to start commenting on the patches themselves.

Anyway, here's the full stats from skiboot 5.1.0 to 5.2.0-rc1:

Processed 334 csets from 27 developers 2 employers found A total of 46172 lines added, 23274 removed (delta 22898)

Developers with the most changesets

Stewart Smith	146 (43.7%)
Cyril Bur	52 (15.6%)
Benjamin Herrenschmidt	15 (4.5%)
Joel Stanley	12 (3.6%)
Gavin Shan	12 (3.6%)
Alistair Popple	10 (3.0%)
Vasant Hegde	10 (3.0%)
Michael Neuling	10 (3.0%)
Russell Currey	9 (2.7%)
Cédric Le Goater	8 (2.4%)
Jeremy Kerr	8 (2.4%)
Samuel Mendoza-Jonas	6 (1.8%)
Neelesh Gupta	6 (1.8%)
Shilpasri G Bhat	4 (1.2%)
Oliver O'Halloran	4 (1.2%)
Mahesh Salgaonkar	4 (1.2%)
Vipin K Parashar	3 (0.9%)
Daniel Axtens	3 (0.9%)
Andrew Donnellan	2 (0.6%)
Philippe Bergheaud	2 (0.6%)
Ananth N Mavinakayanahalli	2 (0.6%)
Vaibhav Jain	1 (0.3%)
Sam Mendoza-Jonas	1 (0.3%)
Adriana Kobylak	1 (0.3%)
Shreyas B. Prabhu	1 (0.3%)
Vaidyanathan Srinivasan	1 (0.3%)
Ian Munsie	1 (0.3%)

Developers with the most changed lines

Stewart Smith	19533 (39.4%)
Oliver O'Halloran	17920 (36.1%)
Alistair Popple	3285 (6.6%)
Daniel Axtens	2154 (4.3%)
Cyril Bur	2028 (4.1%)
Benjamin Herrenschmidt	941 (1.9%)
Neelesh Gupta	434 (0.9%)
Gavin Shan	294 (0.6%)
Russell Currey	261 (0.5%)
Vasant Hegde	245 (0.5%)
Cédric Le Goater	209 (0.4%)
Vipin K Parashar	155 (0.3%)
Shilpasri G Bhat	153 (0.3%)
Joel Stanley	140 (0.3%)
Vaidyanathan Srinivasan	135 (0.3%)
Michael Neuling	111 (0.2%)
Samuel Mendoza-Jonas	81 (0.2%)
Jeremy Kerr	60 (0.1%)
Mahesh Salgaonkar	58 (0.1%)
Vaibhav Jain	50 (0.1%)
Ananth N Mavinakayanahalli	43 (0.1%)
Shreyas B. Prabhu	17 (0.0%)
Sam Mendoza-Jonas	12 (0.0%)
Andrew Donnellan	10 (0.0%)
Ian Munsie	8 (0.0%)
Philippe Bergheaud	6 (0.0%)
Adriana Kobylak	6 (0.0%)

Developers with the most lines removed

Daniel Axtens	2149 (9.2%)
Shreyas B. Prabhu	17 (0.1%)
Andrew Donnellan	9 (0.0%)
Vipin K Parashar	2 (0.0%)

Developers with the most signoffs (total 190)

Stewart Smith	188 (98.9%)
Gavin Shan	1 (0.5%)
Neelesh Gupta	1 (0.5%)

Developers with the most reviews (total 34)

Patrick Williams	5 (14.7%)
Joel Stanley	5 (14.7%)
Cédric Le Goater	5 (14.7%)
Vasant Hegde	4 (11.8%)
Alistair Popple	4 (11.8%)
Sam Mendoza-Jonas	3 (8.8%)
Samuel Mendoza-Jonas	3 (8.8%)
Andrew Donnellan	2 (5.9%)
Cyril Bur	2 (5.9%)
Vaibhav Jain	1 (2.9%)

Developers with the most test credits (total 6)

Vipin K Parashar	3 (50.0%)
Vaibhav Jain	2 (33.3%)
Gajendra B Bandhu1	1 (16.7%)

Developers who gave the most tested-by credits (total 6)

Gavin Shan	2 (33.3%)
Ananth N Mavinakayanahalli	2 (33.3%)
Alistair Popple	1 (16.7%)
Stewart Smith	1 (16.7%)

Developers with the most report credits (total 11)

Vaibhav Jain	2 (18.2%)
Paul Nguyen	2 (18.2%)
Alistair Popple	1 (9.1%)
Cédric Le Goater	1 (9.1%)
Aneesh Kumar K.V	1 (9.1%)
Dionysius d. Bell	1 (9.1%)
Pradeep Ramanna	1 (9.1%)
John Walthour	1 (9.1%)
Benjamin Herrenschmidt	1 (9.1%)

Developers who gave the most report credits (total 11)

Gavin Shan	6 (54.5%)
Stewart Smith	3 (27.3%)
Samuel Mendoza-Jonas	1 (9.1%)
Shilpasri G Bhat	1 (9.1%)

5.1.44 skiboot-5.2.0-rc2

skiboot-5.2.0-rc2 was released on Wednesday March 9th, 2016.

skiboot-5.2.0-rc2 is the second release candidate of skiboot 5.2, which will become the new stable release of skiboot following the 5.1 release, first released August 17th, 2015.

skiboot-5.2.0-rc2 contains all bug fixes as of skiboot-5.1.14.

This is the second release that will follow the (now documented) Skiboot stable rules - see *Skiboot stable tree rules and releases*.

The current plan is to release skiboot-5.2.0 mid-March 2016, with a focus on bug fixing for future 5.2.0-rc releases (if any - I hope this will be the last)

Over skiboot-5.2.0-rc1, we have the following changes:

New platform!

- Add Barreleye platform

Generic

- hw/p8-i2c: Speed up SMBUS_WRITE
- Fix early backtraces

FSP Platforms

- fsp-sensor: rework device tree for sensors
- platforms/firenze: Fix I2C clock source frequency

Simics simulator

- Enable Simics UART console

Mambo simulator

- platforms/mambo: Add terminate callback
  - fix hang in multi-threaded mambo
  - add multithreaded mambo tests

IPMI

- hw/ipmi: fix event data 1 for System Firmware Progress sensor
- ipmi: Log exact NetFn value in OPAL logs

AST BMC based platforms

- hw/bt: allow BT driver to use different buffer size

opal-prd utility

- **opal-prd: Add debug output for firmware-driven OCC events** We indicate when we have a user-driven event, so add corresponding outputs for firmware-driven ones too.

getscom utility

- Add Naples chip support

5.1.45 skiboot-5.2.1

skiboot-5.2.1 was released on Wednesday April 27th, 2016.

skiboot-5.2.1 is the second stable release of skiboot 5.2, the new stable release of skiboot, which will take over from the 5.1.x series which was first released August 17th, 2015.

skiboot-5.2.1 contains all bug fixes as of skiboot-5.1.15.

This is the second release that will follow the (now documented) Skiboot stable rules - see *Skiboot stable tree rules and releases*.

Changes

Over skiboot-5.2.0, the following fixes are included:

pflash

- Allow building under yocto. Makefile fixes to enable building as part of an OpenBMC build.

Garrison platform

- Add PCIe and NPU slot location names
- hw/npu.c: Add ibm, npu-index property to npu device tree
- hmi: Add handling for NPU checkstops

PHB3 (all POWER8 platforms)

- hw/phb3: Ensure PQ bits are cleared in the IVC when masking IRQ When we mask an interrupt, we may race with another interrupt coming in from the hardware. If this occurs, the P and/or Q bit may end up being set but we never EOI/clear them. This could result in a lost interrupt or the next interrupt that comes in after re-enabling never being presented.

  This fixes a bug seen with some CAPI workloads which have lots of interrupt masking at the same time as high interrupt load. The fix is not specific to CAPI though.
- hw/phb3: Fix potential race in EOI When we EOI we need to clear the present (P) bit in the Interrupt Vector Cache (IVC). We must clear P ensuring that any additional interrupts that come in aren't lost while also maintaining coherency with the Interrupt Vector Table (IVT).

  To do this, the hardware provides a conditional update bit in the IVC. This bit ensures that generation counts between the IVT and the IVC updates are synchronised.

Unfortunately we never set this the bit to conditionally update the P bit in the IVC based on the generation count. Also, we didn't set what we wanted the new generation count to be if the update was successful.

FSP platforms

- OPAL:Handle mbox response with bad status:0x24 during FSP termination OPAL committed a predictive log with SRC BB822411 in some situations.

Generic

- hmi: Fix a bug where partial hmi event was reported to host. This bug fix ensures the CPU PIR is reported correctly:

```
  [  305.628283] Fatal Hypervisor Maintenance interrupt [Not recovered]
  [  305.628341]  Error detail: Malfunction Alert
  [  305.628388]    HMER: 8040000000000000
- [  305.628423]    CPU PIR: 00000000
+ [  200.123021]    CPU PIR: 000008e8
  [  305.628458]    [Unit: VSU] Logic core check stop
```

- xscom: Return OPAL_WRONG_STATE on XSCOM ops if CPU is asleep

Contributors

Processed 15 csets from 7 developers A total of 436 lines added, 59 removed (delta 377)

Developers with the most changesets

Russell Currey	7 (46.7%)
Alistair Popple	2 (13.3%)
Michael Neuling	2 (13.3%)
Patrick Williams	1 (6.7%)
Stewart Smith	1 (6.7%)
Mamatha	1 (6.7%)
Mahesh Salgaonkar	1 (6.7%)

Developers with the most changed lines

Alistair Popple	215 (48.3%)
Russell Currey	140 (31.5%)
Michael Neuling	55 (12.4%)
Mamatha	15 (3.4%)
Patrick Williams	9 (2.0%)
Mahesh Salgaonkar	8 (1.8%)
Stewart Smith	3 (0.7%)

Developers with the most lines removed

Patrick Williams	5 (8.5%)

Developers with the most signoffs (total 30)

Stewart Smith	15 (50.0%)
Russell Currey	7 (23.3%)
Michael Neuling	2 (6.7%)
Alistair Popple	2 (6.7%)
Patrick Williams	1 (3.3%)
Oliver O'Halloran	1 (3.3%)
Mahesh Salgaonkar	1 (3.3%)
Mamatha	1 (3.3%)

Developers with the most reviews (total 11)

Alistair Popple	5 (45.5%)
Andrew Donnellan	3 (27.3%)
Mahesh Salgaonkar	2 (18.2%)
Joel Stanley	1 (9.1%)

Developers with the most Acked-by (total 1)

Alistair Popple	1 (100.0%)

Developers with the most test credits (total 3)

Andrew Donnellan	2 (66.7%)
Vaibhav Jain	1 (33.3%)

Developers who received the most tested-by credits (total 3)

Michael Neuling	3 (100.0%)

5.1.46 skiboot-5.2.2

skiboot-5.2.2 was released on Thursday May 5th, 2016.

skiboot-5.2.2 is the third stable release of skiboot 5.2, the new stable release of skiboot, which will take over from the 5.1.x series which was first released August 17th, 2015.

Skiboot 5.2.2 replaces skiboot-5.2.1 as the current stable version, which was released on April 27th, 2016. Over skiboot-5.2.1, skiboot 5.2.2 contains one bug fix targeted at P8NVL systems, notably the Garrison platform.

skiboot-5.2.2 contains all bug fixes as of skiboot-5.1.16.

This is the second release that will follow the (now documented) Skiboot stable rules - see *Skiboot stable tree rules and releases*.

Over skiboot-5.2.1, the following fixes are included:

P8NVL/Garrison

- **PHB3: Fix corruption of pref window register** On P8+ Garrison platform, the root port's pref window register might be not writable and we have to emulate the window because of hardware defect. In order to detect that, we read the register content, write inversed value and read the register content again. The register is regarded as read-only if the values from the two continuous read are same. However, the original register content isn't written back and it causes corruption on pref window register if it's writable.

  This fixes the above issue by writing the original content back to the register at the end.

5.1.47 skiboot-5.2.3

skiboot-5.2.3 was released on Thursday June 30th, 2016.

skiboot-5.2.3 is the 4th stable release of skiboot 5.2, the new stable release of skiboot, which takes over from the 5.1.x series which was first released August 17th, 2015.

Skiboot 5.2.3 replaces skiboot-5.2.2 as the current stable version, which was released on May 5th, 2016. Over skiboot-5.2.2, skiboot 5.2.3 contains one important bug fix regarding parsing data from the OCC regarding CPU frequency tables, which could lead to no CPU frequency scaling.

skiboot-5.2.3 contains all bug fixes as of skiboot-5.1.16.

This is the second release that will follow the (now documented) Skiboot stable rules - see *Skiboot stable tree rules and releases*.

Over skiboot-5.2.2, the following fixes are included:

OpenPOWER platforms

- occ: Filter out entries from Pmin to Pmax in pstate table (cherry picked from commit eca02ee2e62cee115d921a01cea061782ce47cc7) Without this fix, with newer OCC firmware on some OpenPOWER machines, we would fail to parse the table from the OCC, which meant the host OS would not get a table of supported CPU frequencies.

General

- pci: Do a dummy config write to devices to establish bus number (cherry picked from commit f46c1e506d199332b0f9741278c8ec35b3e39135)

  On PCI Express, devices need to know their own bus number in order to provide the correct source identification (aka RID) in upstream packets they might send, such as error messages or DMAs.

  However while devices know (and hard wire) their own device and function number, they know nothing about bus numbers by default, those are decoded by bridges for routing. All they know is that if their parent bridge sends a "type 0" configuration access, they should decode it provided the device and function numbers match.

  The PCIe spec thus defines that when a device receive such a configuration access and it's a write, it should "capture" the bus number in the source field of the packet, and re-use as the originator bus number of all subsequent outgoing requests.

In order to ensure that a device has this bus number firmly established before it's likely to send error packets upstream, we should thus do a dummy configuration write to it as soon as possible after probing.

- Fix GCC 6 warning in backtrace code (cherry picked from commit 793f6f5b32c96f2774bd955b6062c74a672317ca)
- Backport of user visible typo fixes partial cherry picked from 4c95b5e04e3c4f72e4005574f67cd6e365d3276f

Utilities

- Fix ARM build failure with parallel make

5.1.48 skiboot-5.2.4

skiboot-5.2.4 was released on Tuesday July 12th, 2016.

This is the 5th stable release of skiboot 5.2, the new stable release of skiboot (first release with 5.2.0 on March 16th 2016).

Skiboot 5.2.4 replaces skiboot-5.2.3 as the current stable version, which was released on June 30th 2016. Over skiboot-5.2.3, skiboot 5.2.4 contains bug fixes to make skiboot more resilient to errors in the XSCOM engine and some build improvements for the pflash utility.

skiboot-5.2.4 contains all bug fixes as of skiboot-5.1.16.

This is the second release that will follow the (now documented) Skiboot stable rules - see *Skiboot stable tree rules and releases*.

Over skiboot-5.2.3, the following fixes are included:

All platforms

- Make the XSCOM engine code more resilient to errors:
  - hw/xscom: Reset XSCOM engine after querying sleeping core FIR
  - hw/xscom: Reset XSCOM engine after finite number of retries when busy

Userspace utilities

- pflash build improvements

5.1.49 skiboot-5.2.5

skiboot-5.2.5 was released on Thursday July 28th, 2016.

skiboot-5.2.5 contains all bug fixes as of skiboot-5.1.17.

This is the second release that will follow the (now documented) Skiboot stable rules - see *Skiboot stable tree rules and releases*.

Over skiboot-5.2.4, the following fixes are included:

- pflash: Fix the makefile (cherry picked from commit fd599965f723330da5ec55519c20cdb6aa2b3a2d)

- pflash: Clean up makefiles and resolve build race (cherry picked from commit c327eddd9b291a0e6e54001fa3b1e547bad3fca2)
- FSP/ELOG: Fix OPAL generated elog resend logic (cherry picked from commit a6d4a7884e95cb9c918b8a217c11e46b01218358)
- FSP/ELOG: Fix possible event notifier hangs (cherry picked from commit e7c8cba4ad773055f390632c2996d3242b633bf4)
- FSP/ELOG: Disable event notification if list is not consistent (cherry picked from commit 1fb10de164d3ca034193df81c1f5d007aec37781)
- FSP/ELOG: Improve elog event states (cherry picked from commit cec5750a4a86ff3f69e1d8817eda023f4d40c492)
- FSP/ELOG: Fix OPAL generated elog event notification (cherry picked from commit ec366ad4e2e871096fa4c614ad7e89f5bb6f884f)
- FSP/ELOG: Disable event notification during kexec (cherry picked from commit d2ae07fd97bb9408456279cec799f72cb78680a6)
- hw/xscom: Reset XSCOM engine after querying sleeping core FIR (cherry picked from commit 15cec493804ff14e6246eb1b65e9d0c7cb469a81)
- hw/xscom: Reset XSCOM engine after finite number of retries when busy (cherry picked from commit e761222593a1ae932cddbc81239b6a7cd98ddb70)
- xscom: Return OPAL_WRONG_STATE on XSCOM ops if CPU is asleep (cherry picked from commit 9c2d82394fd2303847cac4a665dee62556ca528a)
- fsp/console: Ignore data on unresponsive consoles (cherry picked from commit fd6b71fcc6912611ce81f455b4805f0531699d5e)
- SEL: Fix eSEL ID while logging eSEL event

5.1.50 skiboot-5.3.0

skiboot-5.3.0 was released on Tuesday August 2nd, 2016.

skiboot-5.3.0 is the first stable release of skiboot 5.3, the new stable release of skiboot, which will take over from the 5.2.x series which was first released Wednesday March 16th, 2016.

skiboot-5.3.0 contains all bug fixes as of skiboot-5.1.17 and skiboot-5.2.5.

Changes over skiboot-5.3.0-rc2: - Adopt libtool rules for soname versioning for libflash

See skiboot-5.3.0-rc2 and skiboot-5.3.0-rc1 release notes for a complete list of changes from skiboot-5.3.0.

5.1.51 skiboot-5.3.0-rc1

skiboot-5.3.0-rc1 was released on Monday July 25th, 2016

skiboot-5.3.0-rc1 is the first release candidate of skiboot 5.3, which will become the new stable release of skiboot following the 5.2 release, first released March 16th 2016.

skiboot-5.3.0-rc1 contains all bug fixes as of skiboot-5.1.16 and skiboot-5.2.4 (the existing stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases*.

The current plan is to release skiboot-5.3.0 August 1st 2016.

Over skiboot-5.2, we have the following changes:

OPAL API/Device Tree

- **Reserve OPAL API numbers for XICS emulation for XIVE** Additionally, we put in some skeleton docs for what's coming, key points being that this is for P9 and above, relies on a device being present in the device tree and is modelled on the PAPR calls.
- interrupts: Remove #interrupt-cells from ICP nodes
- Stop adding legacy linux, phandle to device tree, just add phandle No Linux kernel has ever existed for powernv that only knows linux,phandle.

POWER9

- Add base POWER9 support In *NO WAY* is this geared towards real POWER9 hardware. Suitable for use in simulators *only*, and even then, only if you intensely know what you're doing.
- Document changes in OPAL API for POWER9 Some things are going to change, we start documenting them.
- cpu: supply ibm,dec-bits via devicetree
- power9: Add example device tree for phb4
- device-tree: Only advertise ibm, opal-v3 (not v2) on POWER9 and above

CAPI

- phb3: Test CAPI mode on both CAPP units on Naples
- hmi: Recover both CAPP units on Naples after malfunction alert
- chiptod: Sync timebase in both CAPP units on Naples
- phb3: Set CAPI mode for both CAPP units on Naples
- phb3: Load CAPP ucode to both CAPP units on Naples
- **phb3: Add support for CAPP DMA mode** The XSL used in the Mellanox CX4 card uses a DMA mode of CAPI, which requires a few registers configured specially. This adds a new mode to the OPAL_PCI_SET_PHB_CAPI_MODE API to enable CAPI in DMA mode.

PCI

- pci: Do a dummy config write to devices to establish bus number
- phb: Work around XSL bug sending PTE updates with wrong scope
- Support for PCI hotplug (if a platform supports it)

Garrison

- NVLink/NPU support
- Full garrison platform support.

BMC based platforms

- bt: use the maximum retry count returned by the BMC
- **SEL: Fix eSEL ID while logging eSEL event** Commit 127a7dac added eSEL ID to SEL event in reverse order (0700 instead of 0007). This code fixes this issue by adding ID in proper order.

Tests/Simulation

- test/hello_world: always use shutdown type zero
- make check: make test runs less noisy
- boot-tests: force booting from primary (non-golden) side
- mambo: Enable multicore configurations
- mambo: Flatten device tree at the end
- mambo: Increase memory to 4GB and change memory map
- Timebase quirk for slow simulators like AWAN and SIMICS
- chip: Add simics specific quirks
- mambo: Flash driver using bogus disk
- platform/mambo: Add a heartbeat time, making console more responsive
- mambo: Fix bt command and add little endian support

FSP platforms

- beginnings of support for SPIRA-S structure
- Handle mbox response with bad status:0x24 during FSP termination
- FSP: Validate fsp_msg response memory allocation
- FSP/ELOG: Fix OPAL generated elog event notification
- FSP/ELOG: Disable event notification during kexec Possible crash if error log timing around kexec is unfortunate
- fsp/console: Ignore data on unresponsive consoles

  Linux kernels from v4.1 onwards will try to request an irq for each hvc console using OPAL_EVENT_CONSOLE_INPUT, however because the IRQF_SHARED flag is not set any console after the first will fail. If there is data on one of these failed consoles OPAL will set OPAL_EVENT_CONSOLE_INPUT every time fsp_console_read is called, leading to RCU stalls in the kernel.

  As a workaround for unpatched kernels, cease setting OPAL_EVENT_CONSOLE_INPUT for consoles that we have noticed are not being read.

HMI

- hmi: Fix a bug where partial hmi event was reported to host.
- hmi: Add handling for NPU checkstops
- hmi: Only raise a catchall HMI if no other components have

- hmi: Rework HMI event handling of FIR read failure

Tools

- **external: Add a getsram command** The getsram command reads the OCC SRAM. This is useful for debug.
- bug fixes in flash utilities (pflash/gard)
- pflash: Allow building under yocto.
- external/opal-prd: Ensure that struct host_interfaces matches the thunk
- external/pflash: Handle incorrect cmd-line options better
- libflash: fix bug on reading truncated flash file
- pflash: add support for manipulating file rather than flash
- gard: fix compile error on ARM
- libflash: Add sanity checks to ffs init code.
- external: Add dynamically linked pflash

Mambo

- **Test device tree for kernel location** This can reduce the boot time since the kernel no longer needs to relocate itself when loaded directly at 0.

Generic

- hw/lpc: Log LPC SYNC errors as OPAL_PLATFORM_ERR_EVT errors
- Explicitly disable the attn instruction on all CPUs on boot.
- hw/xscom: Reset XSCOM engine after finite number of retries when busy
- hw/xscom: Reset XSCOM engine after querying sleeping core FIR
- core/timer: Add support for platform specific heartbeat
- Fix GCOV_COUNTERS ifdef logic for GCC 6.0
- core: Fix backtrace for gcc 6 fixes a compiler warning on GCC 6 and above
- **cpu: Don't call time_wait with lock held** Also make the locking around re-init safer, properly block the OS from restarting a thread that was caught for re-init.
- flash: Increase the maximum number of flash devices

Contributors

Extending the analysis done for the last few releases, we can see our trends in code review across versions:

Release	csets	Ack	Reviews	Tested	Reported
5.0	329	15	20	1	0
5.1	372	13	38	1	4
5.2-rc1	334	20	34	6	11
5.3-rc1	302	36	53	4	5

An increase in reviews this cycle is great!

Detailed statistics for 5.3.0-rc1 are below:

Processed 302 csets from 31 developers A total of 20887 lines added, 4540 removed (delta 16347)

Developers with the most changesets

Stewart Smith	82 (27.2%)
Gavin Shan	36 (11.9%)
Benjamin Herrenschmidt	28 (9.3%)
Michael Neuling	25 (8.3%)
Vasant Hegde	24 (7.9%)
Russell Currey	14 (4.6%)
Brad Bishop	12 (4.0%)
Vipin K Parashar	10 (3.3%)
Cédric Le Goater	9 (3.0%)
Shreyas B. Prabhu	8 (2.6%)
Jeremy Kerr	7 (2.3%)
Philippe Bergheaud	6 (2.0%)
Cyril Bur	5 (1.7%)
Mukesh Ojha	4 (1.3%)
Alistair Popple	4 (1.3%)
Ian Munsie	4 (1.3%)
Oliver O'Halloran	3 (1.0%)
Chris Smart	3 (1.0%)
Sam Mendoza-Jonas	2 (0.7%)
Joel Stanley	2 (0.7%)
Dinar Valeev	2 (0.7%)
Shilpasri G Bhat	2 (0.7%)
Patrick Williams	2 (0.7%)
Deb McLemore	1 (0.3%)
Balbir Singh	1 (0.3%)
Andrew Donnellan	1 (0.3%)
Suraj Jitindar Singh	1 (0.3%)
Frederic Bonnard	1 (0.3%)
Kamalesh Babulal	1 (0.3%)
Mamatha	1 (0.3%)
Mahesh Salgaonkar	1 (0.3%)

Developers with the most changed lines

Benjamin Herrenschmidt	7491 (34.4%)
Gavin Shan	4821 (22.1%)
Vasant Hegde	4740 (21.7%)
Stewart Smith	1294 (5.9%)
Michael Neuling	620 (2.8%)
Cédric Le Goater	470 (2.2%)
Jeremy Kerr	338 (1.6%)
Shreyas B. Prabhu	330 (1.5%)

Continued on next page

Table 4 – continued from previous page

Vipin K Parashar	305 (1.4%)
Russell Currey	295 (1.4%)
Alistair Popple	229 (1.1%)
Philippe Bergheaud	170 (0.8%)
Ian Munsie	133 (0.6%)
Dinar Valeev	126 (0.6%)
Brad Bishop	80 (0.4%)
Oliver O'Halloran	80 (0.4%)
Cyril Bur	62 (0.3%)
Frederic Bonnard	61 (0.3%)
Sam Mendoza-Jonas	32 (0.1%)
Chris Smart	27 (0.1%)
Shilpasri G Bhat	20 (0.1%)
Patrick Williams	18 (0.1%)
Suraj Jitindar Singh	17 (0.1%)
Mamatha	15 (0.1%)
Mukesh Ojha	8 (0.0%)
Mahesh Salgaonkar	8 (0.0%)
Joel Stanley	4 (0.0%)
Balbir Singh	4 (0.0%)
Kamalesh Babulal	2 (0.0%)
Deb McLemore	1 (0.0%)
Andrew Donnellan	1 (0.0%)

Developers with the most lines removed

Dinar Valeev	68 (1.5%)
Patrick Williams	10 (0.2%)
Mukesh Ojha	4 (0.1%)
Kamalesh Babulal	1 (0.0%)

Developers with the most signoffs (total 249)

Stewart Smith	236 (94.8%)
Vaidyanathan Srinivasan	6 (2.4%)
Benjamin Herrenschmidt	3 (1.2%)
Michael Neuling	2 (0.8%)
Oliver O'Halloran	1 (0.4%)
Vipin K Parashar	1 (0.4%)

Developers with the most reviews (total 53)

Andrew Donnellan	11 (20.8%)
Russell Currey	9 (17.0%)
Joel Stanley	7 (13.2%)
Alistair Popple	7 (13.2%)
Mukesh Ojha	5 (9.4%)
Cyril Bur	3 (5.7%)
Mahesh Salgaonkar	2 (3.8%)
Gavin Shan	2 (3.8%)
Vasant Hegde	2 (3.8%)
Stewart Smith	1 (1.9%)
Vaidyanathan Srinivasan	1 (1.9%)
Vipin K Parashar	1 (1.9%)
Frederic Barrat	1 (1.9%)
Cédric Le Goater	1 (1.9%)

Developers with the most test credits (total 4)

Andrew Donnellan	2 (50.0%)
Russell Currey	1 (25.0%)
Vaibhav Jain	1 (25.0%)

Developers who gave the most tested-by credits (total 4)

Michael Neuling	3 (75.0%)
Gavin Shan	1 (25.0%)

Developers with the most report credits (total 5)

Mukesh Ojha	2 (40.0%)
Russell Currey	1 (20.0%)
Pridhiviraj Paidipeddi	1 (20.0%)
Balbir Singh	1 (20.0%)

Developers who gave the most report credits (total 5)

Gavin Shan	2 (40.0%)
Stewart Smith	2 (40.0%)
Vasant Hegde	1 (20.0%)

5.1.52 skiboot-5.3.0-rc2

skiboot-5.3.0-rc2 was released on Thursday July 28th, 2016.

The current plan is to release skiboot-5.3.0 August 1st 2016.

Over skiboot-5.3.0-rc1, we have the following changes:

pflash

- pflash: Clean up makefiles and resolve build race
- pflash: use atexit for musl compatibility

General

- core/flash: Fix passing pointer instead of value

POWER9

- **mambo: Update Radix Tree Size as per ISA 3.0** In Linux we recently changed to this encoding, so we no longer boot. The associated Linux commit is b23d9c5b9c83c05e013aa52460f12a8365062cf4

FSP Platforms

- platforms/ibm-fsp: Fix incorrect struct member access and comparison
- FSP/MDST: Fix TCE alignment issue In some corner cases (like source memory size = 4097) we may endup doing wrong mapping and corrupting part of SYSDUMP.
- hdat/vpd: Add chip-id property to processor chip node under vpd

CAPI

- hw/phb3: Increase AIB TX command credit for DMA read in CAPP DMA mode

5.1.53 skiboot-5.3.1

skiboot-5.3.1 was released on Wednesday August 10th, 2016.

This is the 2nd stable release of skiboot 5.3, the new stable release of skiboot (first released with 5.3.0 on August 2nd, 2016).

Skiboot 5.3.1 replaces skiboot-5.3.0 as the current stable version. It contains a few minor bug fixes.

This release follows the Skiboot stable rules, see *Skiboot stable tree rules and releases*.

Over skiboot-5.3.0, the following fixes are included:

FSP systems:

- **FSP/ELOG: elog_enable flag should be false by default** This issue is one of the corner case, which is related to recent change went upstream and only observed in the petitboot prompt, where we see only one error log instead of getting all error log in /sys/firmware/opal/elog.

NVLink systems (i.e. Garrison):

- **npu: reword "error" to indicate it's actually a warning** Without this patch, you get spurious FirmWare Test Suite (FWTS) warnings about NVLink not working on machines that aren't fully populated with GPUs.
- **hmi: Clean up NPU FIR debug messages** With the skiboot log set to debug, the FIR (and related registers) were logged all in the same message. It was too much for one line, didn't clarify if the numbers were in hex, and didn't show leading zeroes.

General:

- asm: Fix backtrace for unexpected exception
- correct the log level from PR_ERROR down to PR_INFO for some skiboot log messages.

5.1.54 skiboot-5.3.2

skiboot-5.3.2 was released on Friday August 26th, 2016.

This is the 3rd stable release of skiboot 5.3, the new stable release of skiboot (first released with 5.3.0 on August 2nd, 2016).

Skiboot 5.3.2 replaces skiboot-5.3.1 as the current stable version. It contains a few minor bug fixes.

Over skiboot-5.3.1, the following fixes are included:

- opal/hmi: Fix a TOD HMI failure during a race condition. Rare race condition which meant we wouldn't recover from TOD error
- lpc: Log LPC SYNC errors as unrecoverable ones for manufacturing Only affects systems in manufacturing mode. No behaviour change when not in manufacturing mode.
- hw/phb3: Update capi initialization sequence The capi initialization sequence was revised in a circumvention document when a 'link down' error was converted from fatal to Endpoint Recoverable. Other, non-capi, register setup was corrected even before the initial open-source release of skiboot, but a few capi-related registers were not updated then, so this patch fixes it. The point is that a link-down error detected by the UTL logic will lead to an AIB fence, so that the CAPP unit can detect the error.

5.1.55 skiboot-5.3.3

skiboot-5.3.3 was released on Friday September 2nd, 2016.

This is the 4th stable release of skiboot 5.3, the new stable release of skiboot (first released with 5.3.0 on August 2nd, 2016).

Skiboot 5.3.3 replaces skiboot-5.3.2 as the current stable version. It contains two bug fixes for machines utilizing the NPU (i.e. Garrison)

Over skiboot-5.3.2, the following fixes are included:

- hw/npu: assert the NPU irq min is aligned.
- hw/npu: program NPU BUID reg properly The NPU BUID register was incorrectly programmed resulting in npu interrupt level 0 causing a PB_CENT_CRESP_ADDR_ERROR checkstop, and irqs from npus in odd chips being aliased to and processed as the interrupts from the corresponding npu on the even chips.

5.1.56 skiboot-5.3.4

skiboot-5.3.4 was released on Tuesday September 13th, 2016.

This is the 5th stable release of skiboot 5.3, the new stable release of skiboot (first released with 5.3.0 on August 2nd, 2016).

Skiboot 5.3.4 replaces skiboot-5.3.3 as the current stable version. It contains a couple of bug fixes, specifically around failing XSCOMs.

Over skiboot-5.3.3, the following fixes are included:

- xscom: Initialize the data to a known value in xscom_read In case of error, don't leave the data random. It helps debugging when the user fails to check the error code. This happens due to a bug in the PRD wrapper app.

- xscom: Map all HMER status codes to OPAL errors
- centaur: Mark centaur offline after 10 consecutive access errors This avoids spamming the logs when the centaur is dead and PRD constantly tries to access it
- nvlink: Fix bad PE number check in error inject code path (<= rather than <)

5.1.57 skiboot-5.3.5

skiboot-5.3.5 was released on Wednesday September 14th, 2016.

This is the 6th stable release of skiboot 5.3, the new stable release of skiboot (first released with 5.3.0 on August 2nd, 2016).

Skiboot 5.3.5 replaces skiboot-5.3.4 as the current stable version. It contains a couple of minor bug fixes: simply clarifying two error messages.

Over skiboot-5.3.4, the following fixes are included:

- centaur: print message on disabling xscoms to centaur due to many errors
- slw: improve error message for SLW timer stuck We still register dump, but only to in memory console buffer by default.

5.1.58 skiboot-5.3.6

skiboot-5.3.6 was released on Saturday September 17th, 2016.

This is the 7th stable release of skiboot 5.3, the new stable release of skiboot (first released with 5.3.0 on August 2nd, 2016).

Skiboot 5.3.6 replaces skiboot-5.3.5 as the current stable version. It contains one minor bug fix.

Over skiboot-5.3.5, the following fixes are included:

- SLW: Actually print the register dump only to memory A fix in 5.3.5 was only partially correct, we still had the log priority incorrect for dumping of the SLW registers.

5.1.59 skiboot-5.3.7

skiboot-5.3.7 was released on Wednesday October 12th, 2016.

This is the 8th stable release of skiboot 5.3, the new stable release of skiboot (first released with 5.3.0 on August 2nd, 2016).

Skiboot 5.3.7 replaces skiboot-5.3.6 as the current stable version. It contains a few bugfixes, including an important PCI bug fix that could cause some adapters to not be detected.

Over skiboot-5.3.6, the following fixes are included:

PCI:

- **pci: Avoid hot resets at boot time** In the PCI post-fundamental reset code, a hot reset is performed at the end. This is causing issues at boot time as a reset signal is being sent downstream before the links are up, which is causing issues on adapters behind switches. No errors result in skiboot, but the adapters are not usable in Linux as a result.

  This patch fixes some adapters not being configurable in Linux on some systems. The issue was not present in skiboot 5.2.x.

- **core/pci: Fix the power-off timeout in pci_slot_power_off()** The timeout should be 1000ms instead of 1000 ticks while powering off PCI slot in pci_slot_power_off(). Otherwise, it's likely to hit timeout powering off the PCI slot as below skiboot logs reveal:

  [47912590456,5] SkiBoot skiboot-5.3.6 starting… (snip) [5399532365,7] PHB#0005:02:11.0 Bus 0f..ff scanning… [5399540804,7] PHB#0005:02:11.0 No card in slot [5399576870,5] PHB#0005:02:11.0 Timeout powering off slot [5401431782,3] FIRENZE-PCI: Wrong state 00000000 on slot 8000000002880005

PRD:

- **occ/prd/opal-prd: Queue OCC_RESET event message to host in OpenPOWER** During an OCC reset cycle the system is forced to Psafe pstate. When OCC becomes active, the system has to be restored to its last pstate as requested by host. So host needs to be notified of OCC_RESET event or else system will continue to remian in Psafe state until host requests a new pstate after the OCC reset cycle.
- **opal-prd: Fix error code from scom_read & scom_write** Currently, we always return a zero value from scom_read & scom_write, so the HBRT implementation has no way of detecting errors during scom operations. This change uses the actual return value from the scom operation from the kernel instead.
- **opal-prd: Add get_interface_capabilities to host interfaces** We need a way to indicate behaviour changes & fixes in the prd interface, without requiring a major version bump.

  This change introduces the get_interface_capabilities callback, returning a bitmask of capability flags, pertaining to 'sets' of capabilities. We currently return 0 for all.

IBM FSP Platforms:

- platforms/firenze: Fix clock frequency dt property
- platforms/firence: HDAT: Fix typo in nest-frequency property

NVLink:

- **hw/npu.c: Fix reserved PE#** Currently the reserved PE is set to NPU_NUM_OF_PES, which is one greater than the maximum PE resulting in the following kernel errors at boot:

  [0.000000] pnv_ioda_reserve_pe: Invalid PE 4 on PHB#4 [0.000000] pnv_ioda_reserve_pe: Invalid PE 4 on PHB#5

  Due to a HW errata PE#0 is already reserved in the kernel, so update the opal-reserved-pe device-tree property to match this.

5.1.60 skiboot-5.4.0

skiboot-5.4.0 was released on Friday November 11th 2016. It is the new stable skiboot release, taking over from the 5.3.x series (first released August 2nd, 2016). It comes after four release candidates, which have helped to shake out a few issues.

skiboot-5.4.0 contains all bug fixes as of *skiboot-5.3.7* and *skiboot-5.1.18* (the currently maintained stable releases).

Skiboot 5.4.x becomes the new stable release. For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

Over *skiboot-5.4.0-rc4*, we have a few changes:

- libstb: bump up the byte timeout for tpm i2c requests

  This bumps up the byte timeout for tpm i2c requests from 10ms to 30ms. Some p8dtu systems are getting i2c request timeout.
- external/pflash: Perform the correct cleanup when -F is used to operate on a file.

- Add SuperMicro p8dtu1u and p8dtu2u platforms
- Revert "core/ipmi: Set interrupt-parent property". This reverts commit d997e482705d9fdff8e25fcbe07fb56008f96ae1 (introduced in 5.4.0-rc1)

  A problem was found with pre 4.2 linux kernels where a spurious WARNING would be emitted. This change doesn't matter enough to scare users so we can just revert it.

  ```
  Warning was:
  [    0.947741] irq: irq-62==>hwirq-0x3e mapping failed: -22
  [    0.947793] ------------[ cut here ]------------
  [    0.947838] WARNING: at kernel/irq/irqdomain.c:485
  ```

- libflash/libffs: Fix possible NULL dereference

Previous Release Candidates

There were four release candidates for skiboot 5.4.0:

- *skiboot-5.4.0-rc4*
- *skiboot-5.4.0-rc3*
- *skiboot-5.4.0-rc2*
- *skiboot-5.4.0-rc1*

Changes since skiboot 5.3

Over skiboot-5.3, we have the following changes:

New Features

- Add SuperMicro p8dtu1u and p8dtu2u platforms
- Initial Trusted Boot support (see *Secure and Trusted Boot Library (LibSTB) Documentation*). There are several limitations with this initial release:
  - Only Nuvoton TPM 2.0 is supported
  - Requires hardware rework on late revision Habanero or Firestone boards in order to install TPM.
  - Add i2c Nuvoton TPM 2.0 Driver
  - romcode driver for POWER8 secure ROM
  - See Device tree docs: *Trusted Platform Module (TPM)* and *ibm,secureboot*
  - See *Secure and Trusted Boot Library (LibSTB) Documentation*
- Support `ibm,skiboot` NVRAM partition with skiboot configuration options.
  - These should generally only be used if you either completely know what you are doing or need to work around a skiboot bug. They are **not** intended for end users and are *explicitly* **NOT ABI**.
  - Add support for supplying the kernel boot arguments from the `bootargs` configuration string in the `ibm,skiboot` NVRAM partition.
  - Enabling the experimental fast reset feature is done via this method.
- Add support for nap mode on P8 while in skiboot

  - While nap has been exposed to the Operating System since day 1, we have not utilized low power states when in skiboot itself, leading to higher power consumption during boot. We only enable the functionality after the 0x100 vector has been patched, and we disable it before transferring control to Linux.

- libflash: add 128MB MX66L1G45G part
- Pointer validation of OPAL API call arguments.
  - If the kernel called an OPAL API with vmalloc'd address or any other address range in real mode, we would hit a problem with aliasing. Since the top 4 bits are ignored in real mode, pointers from 0xc.. and 0xd.. (and other ranges) could collide and lead to hard to solve bugs. This patch adds the infrastructure for pointer validation and a simple test case for testing the API
  - The checks validate pointers sent in using `opal_addr_valid()`
- Fast reboot for P8

  This makes reboot take an *awful* lot less time, somewhere between four and ten times faster than a full IPL. It is currently experimental and not enabled by default. You can enable the experimental support via nvram option:

  ```
  # nvram -p ibm,skiboot --update-config experimental-fast-reset=feeling-lucky
  ```

  WARNING: While we *think* we've managed to work out or around most of the kinks with fast-reset, we are *not* enabling it by default in 5.4.

  Notably, fast reset will *not* happen in the following scenarios:

  - platform error

    Most of the time, if we're rebooting due to a platform error, we should trigger a checkstop. However, if we haven't been told what we should do to trigger a checkstop (e.g. on an FSP machine), then we should still fail to fast-reboot.

    So, fast-reboot is disabled in the OPAL_CEC_REBOOT2 code path for the OPAL_REBOOT_PLATFORM_ERROR reboot type.
  - FSP code update
  - Unrecoverable HMI
  - A PHB is in CAPI mode

    If a PHB is in CAPI mode, we cannot safely fast reboot - the PHB will be fenced during the reboot resulting in major problems when we load the new kernel.

    In order to handle this safely, we need to disable CAPI mode before resetting PHBs during the fast reboot. However, we don't currently support this.

    In the meantime, when fast rebooting, check if there are any PHBs with a CAPP attached, and if so, abort the fast reboot and revert to a normal reboot instead.

Documentation

There have been a number of documentation fixes this release. Most prominent is the switch to Sphinx (from the Python project) and ReStructured Text (RST) as the documentation format. RST and Sphinx enable both production of pretty documentation in HTML and PDF formats while remaining readable in their raw form to those with no knowledge of RST.

You can build a HTML site by doing the following:

```
cd doc/
make html
```

As always, documentation patches are very, *very* welcome as we attempt to document the OPAL API, the device tree bindings and important parts of OPAL internals.

We would like the Device Tree documentation to follow the style that can be included in the Device Tree Specification.

General

- Make console-log time more readable: seconds rather than timebase Log format is now `[SECONDS.(tb%512000000),LEVEL]`
- Flash (PNOR) code improvements
  - flash: Make size 64 bit safe This makes the size of flash 64 bit safe so that we can have flash devices greater than 4GB. This is especially useful for mambo disks passed through to Linux.
  - core/flash.c: load actual partition size We are downloading 0x20000 bytes from PNOR for CAPP, but currently the CAPP lid is only 40K.
  - flash: Rework error paths and messages for multiple flash controllers Now that we have mambo bogusdisk flash, we can have many flash chips. This is resulting in some confusing output messages.
- core/init: Fix "failure of getting node in the free list" warning on boot.
- slw: improve error message for SLW timer stuck
- Centaur / XSCOM error handling
  - print message on disabling xscoms to centaur due to many errors
  - Mark centaur offline after 10 consecutive access errors
- XSCOM improvements
  - xscom: Map all HMER status codes to OPAL errors
  - xscom: Initialize the data to a known value in `xscom_read` In case of error, don't leave the data random. It helps debugging when the user fails to check the error code. This happens due to a bug in the PRD wrapper app.
  - chip: Add a quirk for when core direct control XSCOMs are missing
- p8-i2c: Don't crash if a centaur errored out
- cpu: Make endian switch message more informative
- cpu: Display number of started CPUs during boot
- core/init: ensure that HRMOR is zero at boot
- asm: Fix backtrace for unexpected exception
- cpu: Remove pollers calling heuristics from `cpu_wait_job` This will be handled by `time_wait_ms()`. Also remove a useless `smt_medium()`. Note that this introduce a difference in behaviour: time_wait will only call the pollers on the boot CPU while `cpu_wait_job()` could call them on any. However, I can't think of a case where this is a problem.
- cpu: Remove global job queue Instead, target a specific CPU for a global job at queuing time. This will allow us to wake up the target using an interrupt when implementing nap mode. The algorithm used is to look for idle primary threads first, then idle secondaries, and finally the less loaded thread. If nothing can be found, we fallback to a synchronous call.
- lpc: Log LPC SYNC errors as unrecoverable ones for manufacturing
- lpc: Optimize SerIRQ dispatch based on which PSI IRQ fired

- **interrupts: Add new source `->attributes()` callback** This allows a given source to provide per-interrupt attributes such as whether it targets OPAL or Linux and it's estimated frequency.

  The former allows to get rid of the double set of ops used to decide which interrupts go where on some modules like the PHBs and the latter will be eventually used to implement smart caching of the source lookups.
- opal/hmi: Fix a TOD HMI failure during a race condition.
- platform: Add BT to Generic platform

NVRAM

- Support `ibm,skiboot` partition for skiboot specific configuration options
- **flash: Size NVRAM based on ECC for OpenPOWER platforms** If NVRAM has ECC (as per the ffs header) then the actual size of the partition is less than reported by the ffs header in the PNOR then the actual size of the partition is less than reported by the ffs header.

NVLink/NPU

- Fix reserved PE#
- NPU bdfn allocation bugfix
- **Fix bad PE number check** NPUs have 4 PEs which are zero indexed, so {0, 1, 2, 3}. A bad PE number check in npu_err_inject checks if the PE number is greater than 4 as a fail case, so it would wrongly perform operations on a non-existant PE 4.
- Use PCI virtual device
- assert the NPU irq min is aligned.
- program NPU BUID reg properly
- **npu: reword "error" to indicate it's actually a warning** Incorrect FWTS annotation. Without this patch, you get spurious FirmWare Test Suite (FWTS) warnings about NVLink not working on machines that aren't fully populated with GPUs.
- **external: NPU hardware procedure script** Performing NPU hardware procedures requires some config space magic. Put all that magic into a script, so you can just specify the target device and the procedure number.

PCI

- Generic fixes
  - Claim surprise hotplug capability
  - Reserve PCI buses for RC's slot
  - Update PCI topology after power change
  - Return slot cached power state
  - Cache power state on slot without power control
  - Avoid hot resets at boot time
  - Fix initial PCIe slot power state

  - Print CRS retry times It's useful to know the CRS retry times before the PCI device is detected successfully. In PCI hot add case, it usually indicates time consumed for the adapter's firmware to be partially ready (responsive PCI config space).
  - core/pci: Fix the power-off timeout in `pci_slot_power_off()` The timeout should be 1000ms instead of 1000 ticks while powering off PCI slot in `pci_slot_power_off()`. Otherwise, it's likely to hit timeout powering off the PCI slot as below skiboot logs reveal:

```
[5399576870,5] PHB#0005:02:11.0 Timeout powering off slot
```

  - pci: Check power state before powering off slot. Prevents the erroneous "Error -1 powering off slot" error message.

- PHB3
  - Override root slot's `prepare_link_change()` with PHB's
  - Disable surprise link down event on PCI slots
  - Disable ECRC on Broadcom adapter behind PMC switch
- astbmc platforms
  - Support dynamic PCI slot. We might insert a PCIe switch to PHB direct slot and the downstream ports of the PCIe switch supports PCI hotplug.

CAPI

- **hw/phb3: Update capi initialization sequence** The capi initialization sequence was revised in a circumvention document when a 'link down' error was converted from fatal to Endpoint Recoverable. Other, non-capi, register setup was corrected even before the initial open-source release of skiboot, but a few capi-related registers were not updated then, so this patch fixes it.

Mambo Simulator

- Helpers for POWER9 Mambo.
- mambo: Advertise available RADIX page sizes
- mambo: Add section for kernel command line boot args Users can set kernel command line boot arguments for Mambo in a tcl script.
- mambo: add exception and qtrace helpers
- external/mambo: Update skiboot.tcl to add page-sizes nodes to device tree

Simics Simulator

- chiptod: Enable ChipTOD in SIMICS

Utilities

- pflash
  - fix harmless buffer overflow: `fl_total_size` was `uint32_t` not `uint64_t`.
  - Don't try to write protect when writing to flash file

  - Misc small improvements to code and code style
  - makefile bug fixes
  - external/pflash: Make MTD accesses the default

    Now that BMC and host kernel mtd drivers exist and have matured we should use them by default.

    This is especially important since we seem to be telling everyone to use pflash (pflash world domination plans are continuing on schedule).
  - external/pflash: Catch incompatible combination of flags
  - external/common: arm: Don't error trying to wrprotect with MTD access
  - libflash/libffs: Use blocklevel_smart_write() when updating partitions
- external/boot_tests
  - remove lid from the BMC after flashing
  - add the nobooting option -N
  - add arbitrary lid option -F
- **`getscom` / `getsram` / `putscom`: Parse chip-id as hex** We print the chip-id in hex (without a leading 0x), but we fail to parse that same value correctly in `getscom` / `getsram` / `putscom`

```
# getscom -l
...
80000000 | DD2.0 | Centaur memory buffer
# getscom -c 80000000 201140a
Error -19 reading XSCOM
```

  Fix this by assuming base 16 when parsing chip-id.

PRD

- opal-prd: Fix error code from `scom_read` and `scom_write`
- opal-prd: Add get_interface_capabilities to host interfaces
- opal-prd: fix for 64-bit pnor sizes
- **occ/prd/opal-prd: Queue OCC_RESET event message to host in OpenPOWER** During an OCC reset cycle the system is forced to Psafe pstate. When OCC becomes active, the system has to be restored to its last pstate as requested by host. So host needs to be notified of OCC_RESET event or else system will continue to remian in Psafe state until host requests a new pstate after the OCC reset cycle.

IBM FSP Based Platforms

- **fsp/console: Allocate irq for each hvc console** Allocate an irq number for each hvc console and set its interrupt-parent property so that Linux can use the opal irqchip instead of the OPAL_EVENT_CONSOLE_INPUT interface.
- platforms/firenze: Fix clock frequency dt property:

```
[ 1.212366090,3] DT: Unexpected property length /xscom@3fc0000000000/i2cm@a0020/
↪clock-frequency
```

- **HDAT: Fix typo in nest-frequency property** nest-frquency -> nest-frequency

- **platforms/ibm-fsp: Use power_ctl bit when determining slot reset method** The power_ctl bit is used to represent if power management is available. If power_ctl is set to true, then the I2C based external power management functionality will be populated on the PCI slot. Otherwise we will try to use the inband PERST as the fundamental reset, as before.
- **FSP/ELOG: Fix elog timeout issue** Presently we set timeout value as soon as we add elog to queue. If we have multiple elogs to write, it doesn't consider queue wait time. Instead set timeout value when we are actually sending elog to FSP.
- **FSP/ELOG: elog_enable flag should be false by default** This issue is one of the corner case, which is related to recent change went upstream and only observed in the petitboot prompt, where we see only one error log instead of getting all error log in `/sys/firmware/opal/elog`.

POWER9

Skiboot 5.4 contains only *preliminary* support for POWER9. It's suitable only for use in simulators. If working on hardware, use more recent skiboot or development branches. We will not be backporting POWER9 fixes to 5.4.x.

- mambo: Make POWER9 look like DD2
- core/cpu.c: Add OPAL call to setup Nest MMU
- psi: On p9, create an interrupt-map for routing PSI interrupts
- lpc: Add P9 LPC interrupts support
- chiptod: Basic P9 support
- psi: Add P9 support

Testing and Debugging

- test/qemu: bump qemu version used in CI, adds IPMI support
- platform/qemu: add BT and IPMI support Enables testing BT and IPMI functionality in the Qemu simulator
- init: In debug builds, enable debug output to console
- **mem_region: Be a bit smarter about poisoning** Don't poison chunks that are already free and poison regions on first allocation. This speeds things up dramatically.
- **libc: Use 8-bytes stores for non-0 memset too** Memory poisoning hammers this, so let's be a bit smart about it and avoid falling back to byte stores when the data is not 0
- fwts: add annotation for manufacturing mode
- check: Fix bugs in mem region tests
- **Don't set -fstack-protector-all unconditionally** We set it already in DEBUG builds and we use -fstack-protector-strong in release builds which provides most of the benefits and is more efficient.
- **Build host programs (and checks) with debug enabled** This enables memory poisoning in allocations and list checking among other things.
- Add global DEBUG make flag

Command line arguments to BOOTKERNEL

- core/init.c: Fix bootargs parsing

  Currently the bootargs are unconditionally deleted, which causes a bug where the bootargs passed in by the device tree are lost.

  This patch deletes bootargs only if it needs to be replaced by the NVRAM entry.

  This patch also removes KERNEL_COMMAND_LINE config option in favour of using the NVRAM or a device tree.

Other changes

- extract-gcov: build with -m64 if compiler supports it.

  Fixes build break on 32bit ppc64 (e.g. PowerMac G5, where user space is mostly 32bit).

Flash on OpenPOWER platforms

- flash: rework flash_load_resource to correctly read FFS/STB

  This fixes the previous reverts of loading the CAPP partition with STB headers (which broke CAPP partitions without STB headers).

  The new logic fixes both CAPP partition loading with STB headers *and* addresses a long standing bug due to differing interpretations of FFS.

  The f_part utility that *constructs* PNOR files just sets actualSize=totalSize no matter on what the size of the partition is. Prior to this patch, skiboot would always load actualSize, leading to longer than needed IPL.

  The pflash utility updates actualSize, so no developer has really ever noticed this, apart from maybe an inkling that it's odd that a freshly baked PNOR from op-build takes ever so slightly longer to boot than one that has had individual partitions pflashed in.

  With this patch, we now compute actualSize. For partitions with a STB header, we take the payload size from the STB header. For partitions that don't have a STB header, we compute the size either by parsing the ELF header or by looking at the subpartition header and computing it.

  We now need to read the entire partition for partitions with subpartitions so that we pass consistent values to be measured as part of Trusted Boot.

  As of this patch, the actualSize field in FFS is *not* relied on for partition size, we determine it from the content of the partition.

  However, this patch *will* break loading of partitions that are not ELF and do not contain subpartitions. Luckily, nothing in-tree makes use of that.

Contributors

Extending the analysis done for the last few releases, we can see our trends in code review across versions:

Release	csets	Ack	Reviews	Tested	Reported
5.0	329	15	20	1	0
5.1	372	13	38	1	4
5.2-rc1	334	20	34	6	11
5.3-rc1	302	36	53	4	5
5.4-rc1	278	8	19	0	4
5.4.0	361	16	28	1	9

Interesting is the stats of 5.4.0-rc1 versus the final 5.4.0, there's been a doubling of Acks, an increase in reviewed-by and reported-by. There's nothing like an impending release to get people to look closer.

Processed 361 csets from 34 developers A total of 20206 lines added, 5843 removed (delta 14363)

Developers with the most changesets:

Developer	#	%
Stewart Smith	105	(29.1%)
Benjamin Herrenschmidt	50	(13.9%)
Claudio Carvalho	47	(13.0%)
Gavin Shan	24	(6.6%)
Cyril Bur	20	(5.5%)
Oliver O'Halloran	18	(5.0%)
Michael Neuling	12	(3.3%)
Mukesh Ojha	12	(3.3%)
Pridhiviraj Paidipeddi	7	(1.9%)
Vasant Hegde	7	(1.9%)
Russell Currey	7	(1.9%)
Joel Stanley	4	(1.1%)
Alistair Popple	4	(1.1%)
Mahesh Salgaonkar	4	(1.1%)
Nageswara R Sastry	4	(1.1%)
Chris Smart	3	(0.8%)
Sam Mendoza-Jonas	3	(0.8%)
Vipin K Parashar	3	(0.8%)
Balbir Singh	3	(0.8%)
Frederic Barrat	3	(0.8%)
leoluo	2	(0.6%)
Rafael Fonseca	2	(0.6%)
Jack Miller	2	(0.6%)
Patrick Williams	2	(0.6%)
Jeremy Kerr	2	(0.6%)
Suraj Jitindar Singh	2	(0.6%)
Milton Miller	2	(0.6%)
Andrew Donnellan	1	(0.3%)
Shilpasri G Bhat	1	(0.3%)
Frederic Bonnard	1	(0.3%)
Breno Leitao	1	(0.3%)
Anton Blanchard	1	(0.3%)
Nicholas Piggin	1	(0.3%)
Cédric Le Goater	1	(0.3%)

Developers with the most changed lines:

Developer	#	%
Claudio Carvalho	6947	(32.9%)
Stewart Smith	6667	(31.6%)
Benjamin Herrenschmidt	2586	(12.3%)
Gavin Shan	1185	(5.6%)
Cyril Bur	692	(3.3%)
Mukesh Ojha	565	(2.7%)
Oliver O'Halloran	343	(1.6%)
Russell Currey	343	(1.6%)
leoluo	269	(1.3%)
Pridhiviraj Paidipeddi	236	(1.1%)
Balbir Singh	227	(1.1%)
Michael Neuling	211	(1.0%)
Nageswara R Sastry	132	(0.6%)
Cédric Le Goater	115	(0.5%)
Vipin K Parashar	68	(0.3%)
Alistair Popple	66	(0.3%)
Vasant Hegde	65	(0.3%)
Mahesh Salgaonkar	50	(0.2%)
Shilpasri G Bhat	45	(0.2%)
Suraj Jitindar Singh	41	(0.2%)
Nicholas Piggin	34	(0.2%)
Sam Mendoza-Jonas	33	(0.2%)
Jack Miller	32	(0.2%)
Chris Smart	28	(0.1%)
Jeremy Kerr	23	(0.1%)
Milton Miller	19	(0.1%)
Joel Stanley	13	(0.1%)
Andrew Donnellan	13	(0.1%)
Rafael Fonseca	12	(0.1%)
Patrick Williams	11	(0.1%)
Frederic Barrat	6	(0.0%)
Anton Blanchard	3	(0.0%)
Frederic Bonnard	2	(0.0%)
Breno Leitao	2	(0.0%)

Developers with the most lines removed:

Developer	#	%
Cyril Bur	206	(3.5%)
Rafael Fonseca	8	(0.1%)

Developers with the most signoffs (total 278):

Developer	#	%
Stewart Smith	268	(96.4%)
Alistair Popple	4	(1.4%)
Jim Yuan	2	(0.7%)
Cyril Bur	1	(0.4%)
Michael Neuling	1	(0.4%)
Jeremy Kerr	1	(0.4%)
Benjamin Herrenschmidt	1	(0.4%)

Developers with the most reviews (total 28):

Developer	#	%
Andrew Donnellan	6	(21.4%)
Vasant Hegde	5	(17.9%)
Mukesh Ojha	5	(17.9%)
Joel Stanley	3	(10.7%)
Russell Currey	3	(10.7%)
Cyril Bur	2	(7.1%)
Balbir Singh	2	(7.1%)
Alistair Popple	1	(3.6%)
Vaidyanathan Srinivasan	1	(3.6%)

Developers with the most test credits (total 1):

Developer	#	%
Pridhiviraj Paidipeddi	1	(100.0%)

Developers who gave the most tested-by credits (total 1):

Developer	#	%
Gavin Shan	1	(100.0%)

Developers with the most report credits (total 9):

Developer	#	%
Pridhiviraj Paidipeddi	3	(33.3%)
Gavin Shan	1	(11.1%)
Vasant Hegde	1	(11.1%)
Michael Neuling	1	(11.1%)
Benjamin Herrenschmidt	1	(11.1%)
Andrei Warkenti	1	(11.1%)
Li Meng	1	(11.1%)

5.1.61 skiboot-5.4.0-rc1

skiboot-5.4.0-rc1 was released on Monday October 17th 2016. It is the first release candidate of skiboot 5.4, which will become the new stable release of skiboot following the 5.3 release, first released August 2nd 2016.

skiboot-5.4.0-rc1 contains all bug fixes as of *skiboot-5.3.7* and *skiboot-5.1.18* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to release a new release candidate every week until we feel good about it. The aim is for skiboot-5.4.x to be in op-build v1.13, which is due by November 23rd 2016.

Over skiboot-5.3, we have the following changes:

New Features

- Initial Trusted Boot support (see *Secure and Trusted Boot Library (LibSTB) Documentation*). There are several limitations with this initial release:
  - CAPP partition is not measured correctly
  - Only Nuvoton TPM 2.0 is supported
  - Requires hardware rework on late revision Habanero or Firestone boards in order to install TPM.
  - Add i2c Nuvoton TPM 2.0 Driver
  - romcode driver for POWER8 secure ROM
  - See Device tree docs for tpm and ibm,secureboot nodes
  - See main secure and trusted boot documentation.
- Fast reboot for P8

  This makes reboot take an *awful* lot less time, somewhere between four and ten times faster than a full IPL. It is currently experimental and not enabled by default. You can enable the experimental support via nvram option:

  ```
  # nvram -p ibm,skiboot --update-config experimental-fast-reset=feeling-lucky
  ```

  WARNING: This has *known* bugs. For example, if you have used a device in CAPI mode, we will currently *NOT* reset it back to plain PCI. There are also some known issues in most simulators.
- Support `ibm,skiboot` NVRAM partition with skiboot configuration options.
  - These should generally only be used if you either completely know what you are doing or need to work around a skiboot bug. They are **not** intended for end users.
  - Add support for supplying the kernel boot arguments from the `bootargs` configuration string in the `ibm,skiboot` NVRAM partition.
  - Enabling the experimental fast reset feature is done via this method.
- Add support for nap mode on P8 while in skiboot
  - While nap has been exposed to the Operating System since day 1, we have not utilized low power states when in skiboot itself, leading to higher power consumption during boot. We only enable the functionality after the 0x100 vector has been patched, and we disable it before transferring control to Linux.
- libflash: add 128MB MX66L1G45G part
- Pointer validation of OPAL API call arguments.
  - If the kernel called an OPAL API with vmalloc'd address or any other address range in real mode, we would hit a problem with aliasing. Since the top 4 bits are ignored in real mode, pointers from 0xc.. and 0xd.. (and other ranges) could collide and lead to hard to solve bugs. This patch adds the infrastructure for pointer validation and a simple test case for testing the API
  - The checks validate pointers sent in using `opal_addr_valid()`

Documentation

There have been a number of documentation fixes this release. Most prominent is the switch to Sphinx (from the Python project) and ReStructured Text (RST) as the documentation format. RST and Sphinx enable both production of pretty documentation in HTML and PDF formats while remaining readable in their raw form to those with no knowledge of RST.

You can build a HTML site by doing the following:

```
cd doc/
make html
```

As always, documentation patches are very, *very* welcome as we attempt to document the OPAL API, the device tree bindings and important parts of OPAL internals.

We would like the Device Tree documentation to follow the style that can be included in the Device Tree Specification.

General

- Make console-log time more readable: seconds rather than timebase Log format is now `[SECONDS.(tb%512000000),LEVEL]`
- Flash (PNOR) code improvements
  - flash: Make size 64 bit safe This makes the size of flash 64 bit safe so that we can have flash devices greater than 4GB. This is especially useful for mambo disks passed through to Linux.
  - core/flash.c: load actual partition size We are downloading 0x20000 bytes from PNOR for CAPP, but currently the CAPP lid is only 40K.
  - flash: Rework error paths and messages for multiple flash controllers Now that we have mambo bogusdisk flash, we can have many flash chips. This is resulting in some confusing output messages.
- core/init: Fix "failure of getting node in the free list" warning on boot.
- slw: improve error message for SLW timer stuck
- Centaur / XSCOM error handling
  - print message on disabling xscoms to centaur due to many errors
  - Mark centaur offline after 10 consecutive access errors
- XSCOM improvements
  - xscom: Map all HMER status codes to OPAL errors
  - xscom: Initialize the data to a known value in `xscom_read` In case of error, don't leave the data random. It helps debugging when the user fails to check the error code. This happens due to a bug in the PRD wrapper app.
  - chip: Add a quirk for when core direct control XSCOMs are missing
- p8-i2c: Don't crash if a centaur errored out
- cpu: Make endian switch message more informative
- cpu: Display number of started CPUs during boot
- core/init: ensure that HRMOR is zero at boot
- asm: Fix backtrace for unexpected exception

- cpu: Remove pollers calling heuristics from `cpu_wait_job` This will be handled by `time_wait_ms()`. Also remove a useless `smt_medium()`. Note that this introduce a difference in behaviour: time_wait will only call the pollers on the boot CPU while `cpu_wait_job()` could call them on any. However, I can't think of a case where this is a problem.
- cpu: Remove global job queue Instead, target a specific CPU for a global job at queuing time. This will allow us to wake up the target using an interrupt when implementing nap mode. The algorithm used is to look for idle primary threads first, then idle secondaries, and finally the less loaded thread. If nothing can be found, we fallback to a synchronous call.
- lpc: Log LPC SYNC errors as unrecoverable ones for manufacturing
- lpc: Optimize SerIRQ dispatch based on which PSI IRQ fired
- **interrupts: Add new source `->attributes()` callback** This allows a given source to provide per-interrupt attributes such as whether it targets OPAL or Linux and it's estimated frequency.

  The former allows to get rid of the double set of ops used to decide which interrupts go where on some modules like the PHBs and the latter will be eventually used to implement smart caching of the source lookups.
- opal/hmi: Fix a TOD HMI failure during a race condition.
- platform: Add BT to Generic platform

NVRAM

- Support `ibm,skiboot` partition for skiboot specific configuration options
- **flash: Size NVRAM based on ECC for OpenPOWER platforms** If NVRAM has ECC (as per the ffs header) then the actual size of the partition is less than reported by the ffs header in the PNOR then the actual size of the partition is less than reported by the ffs header.

NVLink/NPU

- Fix reserved PE#
- NPU bdfn allocation bugfix
- **Fix bad PE number check** NPUs have 4 PEs which are zero indexed, so {0, 1, 2, 3}. A bad PE number check in npu_err_inject checks if the PE number is greater than 4 as a fail case, so it would wrongly perform operations on a non-existant PE 4.
- Use PCI virtual device
- assert the NPU irq min is aligned.
- program NPU BUID reg properly
- **npu: reword "error" to indicate it's actually a warning** Incorrect FWTS annotation. Without this patch, you get spurious FirmWare Test Suite (FWTS) warnings about NVLink not working on machines that aren't fully populated with GPUs.
- **external: NPU hardware procedure script** Performing NPU hardware procedures requires some config space magic. Put all that magic into a script, so you can just specify the target device and the procedure number.

PCI

- Generic fixes
  - Claim surprise hotplug capability
  - Reserve PCI buses for RC's slot
  - Update PCI topology after power change
  - Return slot cached power state
  - Cache power state on slot without power control
  - Avoid hot resets at boot time
  - Fix initial PCIe slot power state
  - Print CRS retry times It's useful to know the CRS retry times before the PCI device is detected successfully. In PCI hot add case, it usually indicates time consumed for the adapter's firmware to be partially ready (responsive PCI config space).
  - core/pci: Fix the power-off timeout in `pci_slot_power_off()` The timeout should be 1000ms instead of 1000 ticks while powering off PCI slot in `pci_slot_power_off()`. Otherwise, it's likely to hit timeout powering off the PCI slot as below skiboot logs reveal:

    ```
    [5399576870,5] PHB#0005:02:11.0 Timeout powering off slot
    ```
- PHB3
  - Override root slot's `prepare_link_change()` with PHB's
  - Disable surprise link down event on PCI slots
  - Disable ECRC on Broadcom adapter behind PMC switch
- astbmc platforms
  - Support dynamic PCI slot. We might insert a PCIe switch to PHB direct slot and the downstream ports of the PCIe switch supports PCI hotplug.

CAPI

- **hw/phb3: Update capi initialization sequence** The capi initialization sequence was revised in a circumvention document when a 'link down' error was converted from fatal to Endpoint Recoverable. Other, non-capi, register setup was corrected even before the initial open-source release of skiboot, but a few capi-related registers were not updated then, so this patch fixes it.

IPMI

- **core/ipmi: Set interrupt-parent property** This allows ipmi-opal to properly use the OPAL irqchip rather than falling back to the event interface in Linux.

Mambo Simulator

- Helpers for POWER9 Mambo.
- mambo: Advertise available RADIX page sizes

- mambo: Add section for kernel command line boot args Users can set kernel command line boot arguments for Mambo in a tcl script.
- mambo: add exception and qtrace helpers
- external/mambo: Update skiboot.tcl to add page-sizes nodes to device tree

Simics Simulator

- chiptod: Enable ChipTOD in SIMICS

Utilities

- pflash
  - fix harmless buffer overflow: `fl_total_size` was `uint32_t` not `uint64_t`.
  - Don't try to write protect when writing to flash file
  - Misc small improvements to code and code style
  - makefile bug fixes
- external/boot_tests
  - remove lid from the BMC after flashing
  - add the nobooting option -N
  - add arbitrary lid option -F
- **`getscom` / `getsram` / `putscom`: Parse chip-id as hex** We print the chip-id in hex (without a leading 0x), but we fail to parse that same value correctly in `getscom` / `getsram` / `putscom`

  ```
  # getscom -l
  ...
  80000000 | DD2.0 | Centaur memory buffer
  # getscom -c 80000000 201140a
  Error -19 reading XSCOM
  ```

  Fix this by assuming base 16 when parsing chip-id.

PRD

- opal-prd: Fix error code from `scom_read` and `scom_write`
- opal-prd: Add get_interface_capabilities to host interfaces
- opal-prd: fix for 64-bit pnor sizes
- **occ/prd/opal-prd: Queue OCC_RESET event message to host in OpenPOWER** During an OCC reset cycle the system is forced to Psafe pstate. When OCC becomes active, the system has to be restored to its last pstate as requested by host. So host needs to be notified of OCC_RESET event or else system will continue to remian in Psafe state until host requests a new pstate after the OCC reset cycle.

IBM FSP Based Platforms

- **fsp/console: Allocate irq for each hvc console** Allocate an irq number for each hvc console and set its interrupt-parent property so that Linux can use the opal irqchip instead of the OPAL_EVENT_CONSOLE_INPUT interface.
- platforms/firenze: Fix clock frequency dt property:

```
[ 1.212366090,3] DT: Unexpected property length /xscom@3fc0000000000/i2cm@a0020/
↪clock-frequency
```

- **HDAT: Fix typo in nest-frequency property** nest-frquency -> nest-frequency
- **platforms/ibm-fsp: Use power_ctl bit when determining slot reset method** The power_ctl bit is used to represent if power management is available. If power_ctl is set to true, then the I2C based external power management functionality will be populated on the PCI slot. Otherwise we will try to use the inband PERST as the fundamental reset, as before.
- **FSP/ELOG: Fix elog timeout issue** Presently we set timeout value as soon as we add elog to queue. If we have multiple elogs to write, it doesn't consider queue wait time. Instead set timeout value when we are actually sending elog to FSP.
- **FSP/ELOG: elog_enable flag should be false by default** This issue is one of the corner case, which is related to recent change went upstream and only observed in the petitboot prompt, where we see only one error log instead of getting all error log in `/sys/firmware/opal/elog`.

POWER9

- mambo: Make POWER9 look like DD2
- **flash: Move flash node under `ibm,opal/flash/`** This changes the boot ABI, so it's only active for P9 and later systems, even though it's unrelated to hardware changes. There is an associated Linux change to properly search for this node as well.
- core/cpu.c: Add OPAL call to setup Nest MMU
- psi: On p9, create an interrupt-map for routing PSI interrupts
- lpc: Add P9 LPC interrupts support
- chiptod: Basic P9 support
- psi: Add P9 support

Testing and Debugging

- test/qemu: bump qemu version used in CI, adds IPMI support
- platform/qemu: add BT and IPMI support Enables testing BT and IPMI functionality in the Qemu simulator
- init: In debug builds, enable debug output to console
- **mem_region: Be a bit smarter about poisoning** Don't poison chunks that are already free and poison regions on first allocation. This speeds things up dramatically.
- **libc: Use 8-bytes stores for non-0 memset too** Memory poisoning hammers this, so let's be a bit smart about it and avoid falling back to byte stores when the data is not 0
- fwts: add annotation for manufacturing mode
- check: Fix bugs in mem region tests

- **Don't set -fstack-protector-all unconditionally** We set it already in DEBUG builds and we use -fstack-protector-strong in release builds which provides most of the benefits and is more efficient.
- **Build host programs (and checks) with debug enabled** This enables memory poisoning in allocations and list checking among other things.
- Add global DEBUG make flag

Contributors

Extending the analysis done for the last few releases, we can see our trends in code review across versions:

Release	csets	Ack	Reviews	Tested	Reported
5.0	329	15	20	1	0
5.1	372	13	38	1	4
5.2-rc1	334	20	34	6	11
5.3-rc1	302	36	53	4	5
5.4-rc1	278	8	19	0	4

This release has fewer changesets over previous 5.x first release candidates, but that is not indicative of the size or complexity of these changes.

Processed 278 csets from 31 developers A total of 17052 lines added, 4745 removed (delta 12307)

Developers with the most changesets

Stewart Smith	71	(25.5%)
Benjamin Herrenschmidt	50	(18.0%)
Claudio Carvalho	38	(13.7%)
Gavin Shan	20	(7.2%)
Oliver O'Halloran	18	(6.5%)
Mukesh Ojha	9	(3.2%)
Cyril Bur	7	(2.5%)
Russell Currey	7	(2.5%)
Vasant Hegde	7	(2.5%)
Pridhiviraj Paidipeddi	6	(2.2%)
Michael Neuling	6	(2.2%)
Alistair Popple	4	(1.4%)
Sam Mendoza-Jonas	3	(1.1%)
Vipin K Parashar	3	(1.1%)
Balbir Singh	3	(1.1%)
Mahesh Salgaonkar	3	(1.1%)
Frederic Barrat	3	(1.1%)
Chris Smart	2	(0.7%)
Jack Miller	2	(0.7%)
Patrick Williams	2	(0.7%)
Jeremy Kerr	2	(0.7%)
Suraj Jitindar Singh	2	(0.7%)
Milton Miller	2	(0.7%)
Shilpasri G Bhat	1	(0.4%)
Frederic Bonnard	1	(0.4%)

Continued on next page

Table 7 – continued from previous page

Joel Stanley	1	(0.4%)
Breno Leitao	1	(0.4%)
Anton Blanchard	1	(0.4%)
Nicholas Piggin	1	(0.4%)
Nageswara R Sastry	1	(0.4%)
Cédric Le Goater	1	(0.4%)

Developers with the most changed lines

Claudio Carvalho	6817	(38.2%)
Stewart Smith	4677	(26.2%)
Benjamin Herrenschmidt	2586	(14.5%)
Gavin Shan	1005	(5.6%)
Cyril Bur	509	(2.9%)
Mukesh Ojha	361	(2.0%)
Oliver O'Halloran	343	(1.9%)
Russell Currey	343	(1.9%)
Balbir Singh	227	(1.3%)
Pridhiviraj Paidipeddi	194	(1.1%)
Michael Neuling	121	(0.7%)
Cédric Le Goater	115	(0.6%)
Vipin K Parashar	68	(0.4%)
Alistair Popple	66	(0.4%)
Vasant Hegde	65	(0.4%)
Shilpasri G Bhat	45	(0.3%)
Suraj Jitindar Singh	41	(0.2%)
Nicholas Piggin	34	(0.2%)
Sam Mendoza-Jonas	33	(0.2%)
Jack Miller	32	(0.2%)
Nageswara R Sastry	32	(0.2%)
Jeremy Kerr	23	(0.1%)
Mahesh Salgaonkar	21	(0.1%)
Chris Smart	20	(0.1%)
Milton Miller	19	(0.1%)
Patrick Williams	11	(0.1%)
Frederic Barrat	6	(0.0%)
Anton Blanchard	3	(0.0%)
Frederic Bonnard	2	(0.0%)
Joel Stanley	2	(0.0%)
Breno Leitao	2	(0.0%)

Developers with the most lines removed

Cyril Bur	299	(6.3%)

Developers with the most signoffs (total 226)

Stewart Smith	219	(96.9%)
Alistair Popple	4	(1.8%)
Cyril Bur	1	(0.4%)
Jeremy Kerr	1	(0.4%)
Benjamin Herrenschmidt	1	(0.4%)

Developers with the most reviews (total 19)

Mukesh Ojha	5	(26.3%)
Andrew Donnellan	4	(21.1%)
Vasant Hegde	3	(15.8%)
Russell Currey	3	(15.8%)
Balbir Singh	2	(10.5%)
Cyril Bur	1	(5.3%)
Vaidyanathan Srinivasan	1	(5.3%)

Developers with the most test credits (total 0)

Developers who gave the most tested-by credits (total 0)

Developers with the most report credits (total 4)

Benjamin Herrenschmidt	1	(25.0%)
Li Meng	1	(25.0%)
Pridhiviraj Paidipeddi	1	(25.0%)
Gavin Shan	1	(25.0%)

Developers who gave the most report credits (total 4)

Gavin Shan	1	(25.0%)
Vasant Hegde	1	(25.0%)
Russell Currey	1	(25.0%)
Stewart Smith	1	(25.0%)

5.1.62 skiboot-5.4.0-rc2

skiboot-5.4.0-rc2 was released on Wednesday October 26th 2016. It is the second release candidate of skiboot 5.4, which will become the new stable release of skiboot following the 5.3 release, first released August 2nd 2016.

skiboot-5.4.0-rc2 contains all bug fixes as of *skiboot-5.3.7* and *skiboot-5.1.18* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

Since this is a release candidate, it should *NOT* be put into production.

The current plan is to release a new release candidate every week until we feel good about it. The aim is for skiboot-5.4.x to be in op-build v1.13, which is due by November 23rd 2016.

Over *skiboot-5.4.0-rc1*, we have a few changes:

Secure and Trusted Boot

skiboot 5.4.0-rc2 improves upon the progress towards Secure and Trusted Boot in rc1. It is important to note that this is *not* a complete, end-to-end secure/trusted boot implementation.

With the current code, it is now possible to verify and measure resources loaded from PNOR by skiboot (namely the CAPP and BOOTKERNEL partitions).

Note that this functionality is currently *only* available on systems that use the libflash backend. It is *NOT* enabled on IBM FSP based systems. There is some support for some simulators though.

- libstb/stb.c: ignore the secure mode flag unless forced in NVRAM

  For this stage in Trusted Boot development, we are wishing to not force Secure Mode through the whole firmware boot process, but we are wanting to be able to test it (classic chicken and egg problem with build infrastructure).

  We disabled secure mode if the secure-enabled devtree property is read from the device tree *IF* we aren't overriding it through NVRAM. Seeing as we can only increase (not decrease) what we're checking through the NVRAM variable, it is safe.

  The NVRAM setting is force-secure-mode=true in the ibm,skiboot partition.

  However, if you want to force secure mode even if Hostboot has *not* set the secure-enabled proprety in the device tree, set force-secure-mode to "always".

  There is also a force-trusted-mode NVRAM setting to force trusted mode even if Hostboot has not enabled it int the device tree.

  To indicate to Linux that we haven't gone through the whole firmware process in secure mode, we replace the 'secure-enabled' property with 'partial-secure-enabled', to indicate that only part of the firmware boot process has gone through secure mode.

Command line arguments to BOOTKERNEL

- core/init.c: Fix bootargs parsing

  Currently the bootargs are unconditionally deleted, which causes a bug where the bootargs passed in by the device tree are lost.

  This patch deletes bootargs only if it needs to be replaced by the NVRAM entry.

  This patch also removes KERNEL_COMMAND_LINE config option in favour of using the NVRAM or a device tree.

pflash utility

- external/pflash: Make MTD accesses the default

  Now that BMC and host kernel mtd drivers exist and have matured we should use them by default.

  This is especially important since we seem to be telling everyone to use pflash (pflash world domination plans are continuing on schedule).

- external/pflash: Catch incompatible combination of flags
- external/common: arm: Don't error trying to wrprotect with MTD access
- libflash/libffs: Use blocklevel_smart_write() when updating partitions

Other changes

- extract-gcov: build with -m64 if compiler supports it.

  Fixes build break on 32bit ppc64 (e.g. PowerMac G5, where user space is mostly 32bit).

Fast Reset

- fast-reset: disable fast reboot in event of platform error

  Most of the time, if we're rebooting due to a platform error, we should trigger a checkstop. However, if we haven't been told what we should do to trigger a checkstop (e.g. on an FSP machine), then we should still fail to fast-reboot.

  So, disable fast-reboot in the OPAL_CEC_REBOOT2 code path for OPAL_REBOOT_PLATFORM_ERROR reboot type.
- fast-reboot: disable on FSP code update or unrecoverable HMI
- fast-reboot: abort fast reboot if CAPP attached

  If a PHB is in CAPI mode, we cannot safely fast reboot - the PHB will be fenced during the reboot resulting in major problems when we load the new kernel.

  In order to handle this safely, we need to disable CAPI mode before resetting PHBs during the fast reboot. However, we don't currently support this.

  In the meantime, when fast rebooting, check if there are any PHBs with a CAPP attached, and if so, abort the fast reboot and revert to a normal reboot instead.

OpenPOWER Platforms

For all hardware platforms that aren't IBM FSP machines:

- Revert "flash: Move flash node under ibm,opal/flash/"

  This reverts commit e1e6d009860d0ef60f9daf7a0fbe15f869516bd0.

  Breaks DT enough that it makes people cranky, reverting for now. This could break access to flash with existing kernels in POWER9 simulators
- flash: rework flash_load_resource to correctly read FFS/STB

  This fixes the previous reverts of loading the CAPP partition with STB headers (which broke CAPP partitions without STB headers).

  The new logic fixes both CAPP partition loading with STB headers *and* addresses a long standing bug due to differing interpretations of FFS.

  The f_part utility that *constructs* PNOR files just sets actualSize=totalSize no matter on what the size of the partition is. Prior to this patch, skiboot would always load actualSize, leading to longer than needed IPL.

  The pflash utility updates actualSize, so no developer has really ever noticed this, apart from maybe an inkling that it's odd that a freshly baked PNOR from op-build takes ever so slightly longer to boot than one that has had individual partitions pflashed in.

  With this patch, we now compute actualSize. For partitions with a STB header, we take the payload size from the STB header. For partitions that don't have a STB header, we compute the size either by parsing the ELF header or by looking at the subpartition header and computing it.

We now need to read the entire partition for partitions with subpartitions so that we pass consistent values to be measured as part of Trusted Boot.

As of this patch, the actualSize field in FFS is *not* relied on for partition size, we determine it from the content of the partition.

However, this patch *will* break loading of partitions that are not ELF and do not contain subpartitions. Luckily, nothing in-tree makes use of that.

PCI

- pci: Check power state before powering off slot

  Prevents the erroneous "Error -1 powering off slot" error message.

Contributors

Since *skiboot-5.4.0-rc1*, we have 23 csets from 8 developers.

A total of 876 lines added, 621 removed (delta 255)

Developers with the most changesets

Developer	#	%
Stewart Smith	7	(30.4%)
Cyril Bur	5	(21.7%)
Mukesh Ojha	3	(13.0%)
Gavin Shan	3	(13.0%)
Claudio Carvalho	2	(8.7%)
Chris Smart	1	(4.3%)
Andrew Donnellan	1	(4.3%)
Nageswara R Sastry	1	(4.3%)

Developers with the most changed lines

Developer	#	%
Stewart Smith	424	(45.7%)
Mukesh Ojha	204	(22.0%)
Gavin Shan	173	(18.6%)
Cyril Bur	69	(7.4%)
Claudio Carvalho	35	(3.8%)
Andrew Donnellan	13	(1.4%)
Chris Smart	8	(0.9%)
Nageswara R Sastry	2	(0.2%)

Developers with the most lines removed

Developer	#	%
Gavin Shan	9	(1.4%)
Chris Smart	4	(0.6%)

Developers with the most signoffs (total 16)

Developer	#	%
Stewart Smith	16	(100.0%)

Developers with the most reviews (total 4)

Developer	#	%
Vasant Hegde	2	(50.0%)
Andrew Donnellan	2	(50.0%)

Developers with the most test credits (total 1)

Developer	#	%
Pridhiviraj Paidipeddi	1	(100.0%)

Developers who gave the most tested-by credits (total 1)

Developer	#	%
Gavin Shan	1	(100.0%)

Developers with the most report credits (total 3)

Developer	#	%
Pridhiviraj Paidipeddi	1	(33.3%)
Andrei Warkenti	1	(33.3%)
Michael Neuling	1	(33.3%)

Developers who gave the most report credits (total 3)

Developer	#	%
Stewart Smith	2	(66.7%)
Gavin Shan	1	(33.3%)

5.1.63 skiboot-5.4.0-rc3

skiboot-5.4.0-rc3 was released on Wednesday November 2nd 2016. It is the third release candidate of skiboot 5.4, which will become the new stable release of skiboot following the 5.3 release, first released August 2nd 2016.

skiboot-5.4.0-rc3 contains all bug fixes as of *skiboot-5.3.7* and *skiboot-5.1.18* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

Since this is a release candidate, it should *NOT* be put into production.

The current plan is to release a new release candidate every week until we feel good about it. The aim is for skiboot-5.4.x to be in op-build v1.13, which is due by November 23rd 2016.

Over *skiboot-5.4.0-rc2*, we have a few changes:

- pflash: Fail when file is larger than partition You can still shoot yourself in the foot by passing –force.
- core/flash: Don’t do anything clever for OPAL_FLASH_{READ, WRITE, ERASE} This fixes a bug where opal-prd and opal-gard could fail. Fixes: https://github.com/open-power/skiboot/issues/44

- boot-tests: force BMC to boot from non-golden side
- fast-reset: Send special reset sequence to operational CPUs only. Fixes fast-reset for cases where there are garded CPUs
- Secure/Trusted boot: be much clearer about what is being measured where.
- Secure/Trusted boot: be more resilient to disabled TPM(s).
- Secure/Trusted boot: The `force-secure-mode` NVRAM setting introduced temporarily in *skiboot-5.4.0-rc2* has changed behaviour. Now, by default, the `secure-mode` flag in the device tree is obeyed. As always, any skiboot NVRAM options are in no way ABI, API or supported and may cause unfinished verbose analogies to appear in release notes relating to the dangers of using developer only options.
- gard: Fix compiler warning on modern GCC targetting ARM 32-bit
- opal-prd: systemd scripts improvements, only run on supported systems

5.1.64 skiboot-5.4.0-rc4

skiboot-5.4.0-rc4 was released on Tuesday November 8th 2016. It is the fourth (and hopefully final) release candidate of skiboot 5.4, which will become the new stable release of skiboot following the 5.3 release, first released August 2nd 2016.

skiboot-5.4.0-rc4 contains all bug fixes as of *skiboot-5.3.7* and *skiboot-5.1.18* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

Since this is a release candidate, it should *NOT* be put into production.

With this release candidate, I'm hoping that it's the last one, and that within the week we're able to tag a final 5.4.0 release. There is one bit of code I'm hoping to merge in before the final 5.4.0, and that's the p8dtu platform definition. The aim is for skiboot-5.4.x to be in op-build v1.13, which is due by November 23rd 2016.

Over *skiboot-5.4.0-rc3*, we have a few changes:

- Add BMC platform to enable correct OEM IPMI commands

  An out of tree platform (p8dtu) uses a different IPMI OEM command for IPMI_PARTIAL_ADD_ESEL. This exposed some assumptions about the BMC implementation in our core code.

  Now, with platform.bmc, each platform can dictate (or detect) the BMC that is present. We allow it to be set at runtime rather than purely statically in struct platform as it's possible to have differing BMC implementations on the one machine (e.g. AMI BMC or OpenBMC).
- hw/ipmi-sensor: Fix setting of firmware progress sensor properly.

  On FSP systems, OPAL was incorrectly setting firmware status on a sensor id "00" which doesn't exist.
- pflash: remove stray d in from info message
- libflash/pflash: support whole chip erase on mtd access
- boot_test: fix typo in console message
- core/pci: Fix criteria in pci_cfg_reg_filter(), i.e. NVLink didn't work.
- Remove KERNEL_COMMAND_LINE mention from config.h

  We removed the functionality but not the define.

5.1.65 skiboot-5.4.1

skiboot-5.4.1 was released on Tuesday November 29th 2016. It replaces *skiboot-5.4.0* as the current stable release.

Over *skiboot-5.4.0*, we have a few changes:

- Nuvoton i2c TPM driver: bug fixes and improvements, especially around timeouts and error handling.
- Limit number of "Poller recursion detected" errors to display. In some error conditions, we could spiral out of control on this and spend all of our time printing the exact same backtrace.
- slw: do SLW timer testing while holding xscom lock. In some situations without this, it could take long enough to get the xscom lock that the 1ms timeout would expire and we'd falsely think the SLW timer didn't work when in fact it did.
- p8i2c: Use calculated poll_interval when booting OPAL. Otherwise we'd default to 2seconds (TIMER_POLL) during boot on chips with a functional i2c interrupt, leading to slow i2c during boot (or hitting timeouts instead).
- i2c: More efficiently run TPM I2C operations during boot, avoiding hitting timeouts
- fsp: Don't recurse pollers in ibm_fsp_terminate

5.1.66 skiboot-5.4.10

skiboot-5.4.10 was released on Monday May 28th, 2018. It replaces *skiboot-5.4.9* as the current stable release in the 5.4.x series.

Over *skiboot-5.4.9*, we have a few bug fixes:

- opal-prd: Do not error out on first failure for soft/hard offline.

  The memory errors (CEs and UEs) that are detected as part of background memory scrubbing are reported by PRD asynchronously to opal-prd along with affected memory ranges. hservice_memory_error() converts these ranges into page granularity before hooking up them to soft/hard offline-ing infrastructure.

  But the current implementation of hservice_memory_error() does not hookup all the pages to soft/hard offline-ing if any of the page offline action fails. e.g hard offline can fail for:

  – Pages that are not part of buddy managed pool.
  – Pages that are reserved by kernel using memblock_reserved()
  – Pages that are in use by kernel.

  But for the pages that are in use by user space application, the hard offline marks the page as hwpoison, sends SIGBUS signal to kill the affected application as recovery action and returns success.

  Hence, It is possible that some of the pages in that memory range are in use by application or free. By stopping on first error we loose the opportunity to hwpoison the subsequent pages which may be free or in use by application. This patch fixes this issue.

- OPAL_PCI_SET_POWER_STATE: fix locking in error paths

  Otherwise we could exit OPAL holding locks, potentially leading to all sorts of problems later on.

- p8-i2c: Limit number of retry attempts

  Current we will attempt to start an I2C transaction until it succeeds. In the event that the OCC does not release the lock on an I2C bus this results in an async token being held forever and the kernel thread that started the transaction will block forever while waiting for an async completion message. Fix this by limiting the number of attempts to start the transaction.

- FSP/CONSOLE: Disable notification on unresponsive consoles

  Commit fd6b71fc fixed the situation where ipmi console was open (hvc0) but got data on different console (hvc1).

  During FSP R/R OPAL closes all consoles. After R/R complete FSP requests to open hvc1 and sends data on this. If hvc1 registration failed or not opened in host kernel then it will not read data and results in RCU stalls.

  Note that this is workaround for older kernel where we don't have separate irq for each console. Latest kernel works fine without this patch.

5.1.67 skiboot-5.4.2

skiboot-5.4.2 was released on Friday December 2nd 2016. It replaces *skiboot-5.4.1* as the current stable release.

Over *skiboot-5.4.1*, we have two bug fixes exclusively aimed at machines with TPMs:

- i2c: Add nuvoton TPM quirk, disallowing i2cdetect as it can hard lock the TPM
- p8-i2c improve I2C reset code path, solves getting stuck resetting i2c engine

5.1.68 skiboot-5.4.3

skiboot-5.4.3 was released on Monday January 16th, 2017. It replaces *skiboot-5.4.2* as the current stable release.

Over *skiboot-5.4.2*, we have a small number of bug fixes:

- Makefile: Disable stack protector due to gcc problems
- Makefile: Use -ffixed-r13. We use r13 for our own stuff, make sure it's properly fixed
- phb3: Lock the PHB on set_xive callbacks
- arch_flash_arm: Don't assume mtd labels are short
- Stop using 3-operand cmp[l][i] for latest binutils
- hw/phb3: fix error handling in complete reset

5.1.69 skiboot-5.4.4

skiboot-5.4.4 was released on Wednesday May 3rd, 2017. It replaces *skiboot-5.4.3* as the current stable release in the 5.4.x series.

Over *skiboot-5.4.3*, we have a small number of bug fixes:

- hw/fsp: Do not queue SP and SPCN class messages during reset/reload In certain cases of communicating with the FSP (e.g. sensors), the OPAL FSP driver returns a default code (async completion) even though there is no known bound from the time of this error return to the actual data being available. The kernel driver keeps waiting leading to soft-lockup on the host side.

  Mitigate both these (known) cases by returning OPAL_BUSY so the host driver knows to retry later.

- core/pci: Fix PCIe slot's presence According to PCIe spec, the presence bit is hardcoded to 1 if PCIe switch downstream port doesn't support slot capability. The register used for the check in pcie_slot_get_presence_state() is wrong. It should be PCIe capability register instead of PCIe slot capability register. Otherwise, we always have present bit on the PCI topology.

  The issue is found on Supermicro's p8dtu2u machine:

```
 # lspci -t
 -+-[0022:00]---00.0-[01-08]----00.0-[02-08]--+-01.0-[03]----00.0
  |                                           \-02.0-[04-08]--
 # cat /sys/bus/pci/slots/S002204/adapter
 1
 # lspci -vvs 0022:02:02.0
 # lspci -vvs 0022:02:02.0
 0022:02:02.0 PCI bridge: PLX Technology, Inc. PEX 8718 16-Lane, \
 5-Port PCI Express Gen 3 (8.0 GT/s) Switch (rev ab) (prog-if 00 [Normal decode])
    :
 Capabilities: [68] Express (v2) Downstream Port (Slot+), MSI 00
    :
    SltSta:    Status: AttnBtn- PowerFlt- MRL- CmdCplt- PresDet- Interlock-
               Changed: MRL- PresDet- LinkState-

This fixes the issue by checking the correct register (PCIe capability).
Also, the register's value is cached in advance as we did for slot and
link capability.
```

- core/pci: More reliable way to update PCI slot power state

  The power control bit (SLOT_CTL, offset: PCIe cap + 0x18) isn't reliable enough to reflect the PCI slot's power state. Instead, the power indication bits are more reliable comparatively. This leads to mismatch between the cached power state and PCI slot's presence state, resulting in the hotplug driver in kernel refuses to unplug the devices properly on the request. The issue was found on below NVMe card on "supermicro,p8dtu2u" machine. We don't have this issue on the integrated PLX 8718 switch.

```
# lspci
0022:01:00.0 PCI bridge: PLX Technology, Inc. PEX 9733 33-lane, \
             9-port PCI Express Gen 3 (8.0 GT/s) Switch (rev aa)
0022:02:01.0 PCI bridge: PLX Technology, Inc. PEX 9733 33-lane, \
             9-port PCI Express Gen 3 (8.0 GT/s) Switch (rev aa)
0022:02:04.0 PCI bridge: PLX Technology, Inc. PEX 9733 33-lane, \
             9-port PCI Express Gen 3 (8.0 GT/s) Switch (rev aa)
0022:02:05.0 PCI bridge: PLX Technology, Inc. PEX 9733 33-lane, \
             9-port PCI Express Gen 3 (8.0 GT/s) Switch (rev aa)
0022:02:06.0 PCI bridge: PLX Technology, Inc. PEX 9733 33-lane, \
             9-port PCI Express Gen 3 (8.0 GT/s) Switch (rev aa)
0022:02:07.0 PCI bridge: PLX Technology, Inc. PEX 9733 33-lane, \
             9-port PCI Express Gen 3 (8.0 GT/s) Switch (rev aa)
0022:17:00.0 Non-Volatile memory controller: Device 19e5:0123 (rev 45)
```

  This updates the cached PCI slot's power state using the power indication bits instead of power control bit, to fix above issue.

- core/pci: Avoid hreset after freset

5.1.70 skiboot-5.4.5

skiboot-5.4.5 was released on Friday June 9th, 2017. It replaces *skiboot-5.4.4* as the current stable release in the 5.4.x series.

Over *skiboot-5.4.4*, we have a small number of bug fixes:

- On FSP platforms: notify FSP of Platform Log ID after Host Initiated Reset Reload Trigging a Host Initiated Reset (when the host detects the FSP has gone out to lunch and should be rebooted), would cause "Unknown Command" messages to appear in the OPAL log.

This patch implements those messages.

Log showing unknown command:

```
/ # cat /sys/firmware/opal/msglog | grep -i ,3
[  110.232114723,3] FSP: fsp_trigger_reset() entry
[  188.431793837,3] FSP #0: Link down, starting R&R
[  464.109239162,3] FSP #0: Got XUP with no pending message !
[  466.340598554,3] FSP-DPO: Unknown command 0xce0900
[  466.340600126,3] FSP: Unhandled message ce0900
```

- hw/i2c: Fix early lock drop

  When interacting with an I2C master the p8-i2c driver (common to p9) aquires a per-master lock which it holds for the duration of it's interaction with the master. Unfortunately, when p8_i2c_check_initial_status() detects that the master is busy with another transaction it drops the lock and returns OPAL_BUSY. This is contrary to the driver's locking strategy which requires that the caller aquire and drop the lock. This leads to a crash due to the double unlock(), which skiboot treats as fatal.

- head.S: store all of LR and CTR

  When saving the CTR and LR registers the skiboot exception handlers use the 'stw' instruction which only saves the lower 32 bits of the register. Given these are both 64 bit registers this leads to some strange register dumps, for example:

```
***********************************************
Unexpected exception 200 !
SRR0 : 0000000030016968 SRR1 : 9000000000201000
HSRR0: 0000000000000180 HSRR1: 9000000000001000
LR   : 3003438830823f50 CTR  : 3003438800000018
CFAR : 00000000300168fc
CR   : 40004208  XER: 00000000
```

  In this dump the upper 32 bits of LR and CTR are actually stack gunk which obscures the underlying issue.

5.1.71 skiboot-5.4.6

skiboot-5.4.6 was released on Wednesday June 14th, 2017. It replaces *skiboot-5.4.5* as the current stable release in the 5.4.x series.

Over *skiboot-5.4.5*, we have a small number of bug fixes for FSP based platforms:

- FSP/CONSOLE: Workaround for unresponsive ipmi daemon

  In some corner cases, where FSP is active but not responding to console MBOX message (due to buggy IPMI) and we have heavy console write happening from kernel, then eventually our console buffer becomes full. At this point OPAL starts sending OPAL_BUSY_EVENT to kernel. Kernel will keep on retrying. This is creating kernel soft lockups. In some extreme case when every CPU is trying to write to console, user will not be able to ssh and thinks system is hang.

  If we reset FSP or restart IPMI daemon on FSP, system recovers and everything becomes normal.

  This patch adds workaround to above issue by returning OPAL_HARDWARE when cosole is full. Side effect of this patch is, we may endup dropping latest console data. But better to drop console data than system hang.

  Alternative approach is to drop old data from console buffer, make space for new data. But in normal condition only FSP can update 'next_out' pointer and if we touch that pointer, it may introduce some other race conditions. Hence we decided to just new console write request.

- FSP: Set status field in response message for timed out message

  For timed out FSP messages, we set message status as "fsp_msg_timeout". But most FSP driver users (like surviellance) are ignoring this field. They always look for FSP returned status value in callback function (second byte in word1). So we endup treating timed out message as success response from FSP.

  Sample output:

  ```
  [69902.432509048,7] SURV: Sending the heartbeat command to FSP
  [70023.226860117,4] FSP: Response from FSP timed out, word0 = d66a00d7, word1 = 0␣
  ↪state: 3
  ....
  [70023.226901445,7] SURV: Received heartbeat acknowledge from FSP
  [70023.226903251,3] FSP: fsp_trigger_reset() entry
  ```

  Here SURV code thought it got valid response from FSP. But actually we didn't receive response from FSP.

- FSP: Improve timeout message

  Presently we print word0 and word1 in error log. word0 contains sequence number and command class. One has to understand word0 format to identify command class.

  Lets explicitly print command class, sub command etc.

- FSP/RTC: Remove local fsp_in_reset variable

  Now that we are using fsp_in_rr() to detect FSP reset/reload, fsp_in_reset become redundant. Lets remove this local variable.

- FSP/RTC: Fix possible FSP R/R issue in rtc write path

  fsp_opal_rtc_write() checks FSP status before queueing message to FSP. But if FSP R/R starts before getting response to queued message then we will continue to return OPAL_BUSY_EVENT to host. In some extreme condition host may experience hang. Once FSP is back we will repost message, get response from FSP and return OPAL_SUCCESS to host.

  This patch caches new values and returns OPAL_SUCCESS if FSP R/R is happening. And once FSP is back we will send cached value to FSP.

- hw/fsp/rtc: read/write cached rtc tod on fsp hir.

  Currently fsp-rtc reads/writes the cached RTC TOD on an fsp reset. Use latest fsp_in_rr() function to properly read the cached rtc value when fsp reset initiated by the hir.

  Below is the kernel trace when we set hw clock, when hir process starts.

  ```
  [ 1727.775824] NMI watchdog: BUG: soft lockup - CPU#57 stuck for 23s!␣
  ↪[hwclock:7688]
  [ 1727.775856] Modules linked in: vmx_crypto ibmpowernv ipmi_powernv uio_pdrv_
  ↪genirq ipmi_devintf powernv_op_panel uio ipmi_msghandler powernv_rng leds_
  ↪powernv ip_tables x_tables autofs4 ses enclosure scsi_transport_sas crc32c_
  ↪vpmsum lpfc ipr tg3 scsi_transport_fc
  [ 1727.775883] CPU: 57 PID: 7688 Comm: hwclock Not tainted 4.10.0-14-generic #16-
  ↪Ubuntu
  [ 1727.775883] task: c000000fdfdc8400 task.stack: c000000fdfef4000
  [ 1727.775884] NIP: c00000000090540c LR: c0000000000846f4 CTR: 000000003006dd70
  [ 1727.775885] REGS: c000000fdfef79a0 TRAP: 0901   Not tainted  (4.10.0-14-
  ↪generic)
  [ 1727.775886] MSR: 9000000000009033 <SF,HV,EE,ME,IR,DR,RI,LE>
  [ 1727.775889]   CR: 28024442  XER: 20000000
  [ 1727.775890] CFAR: c00000000008472c SOFTE: 1
                 GPR00: 0000000030005128 c000000fdfef7c20 c00000000144c900␣
  ↪fffffffffffffff4
  ```

(continues on next page)

(continued from previous page)

```
                GPR04: 0000000028024442 c00000000090540c 9000000000009033␣
↪0000000000000000
                GPR08: 0000000000000000 0000000031fc4000 c000000000084710␣
↪9000000000001003
                GPR12: c0000000000846e8 c00000000fba0100
[ 1727.775897] NIP [c00000000090540c] opal_set_rtc_time+0x4c/0xb0
[ 1727.775899] LR [c0000000000846f4] opal_return+0xc/0x48
[ 1727.775899] Call Trace:
[ 1727.775900] [c000000fdfef7c20] [c00000000090540c] opal_set_rtc_time+0x4c/0xb0␣
↪(unreliable)
[ 1727.775901] [c000000fdfef7c60] [c000000000900828] rtc_set_time+0xb8/0x1b0
[ 1727.775903] [c000000fdfef7ca0] [c000000000902364] rtc_dev_ioctl+0x454/0x630
[ 1727.775904] [c000000fdfef7d40] [c00000000035b1f4] do_vfs_ioctl+0xd4/0x8c0
[ 1727.775906] [c000000fdfef7de0] [c00000000035bab4] SyS_ioctl+0xd4/0xf0
[ 1727.775907] [c000000fdfef7e30] [c00000000000b184] system_call+0x38/0xe0
[ 1727.775908] Instruction dump:
[ 1727.775909] f821ffc1 39200000 7c832378 91210028 38a10020 39200000 38810028␣
↪f9210020
[ 1727.775911] 4bfffe6d e8810020 80610028 4b77f61d <60000000> 7c7f1b78 3860000a␣
↪2fbffff4
```

This is found when executing the op-test-framework fspresetReload testcase

With this fix ran fsp hir torture testcase in the above test which is working fine.

- FSP/CHIPTOD: Return false in error path

5.1.72 skiboot-5.4.7

skiboot-5.4.7 was released on Tuesday September 19th, 2017. It replaces *skiboot-5.4.6* as the current stable release in the 5.4.x series.

Over *skiboot-5.4.6*, we have two backported bug fixes for FSP platforms:

- FSP: Add check to detect FSP Reset/Reload inside fsp_sync_msg()

  During FSP Reset/Reload we move outstanding MBOX messages from msgq to rr_queue including inflight message (fsp_reset_cmdclass()). But we are not resetting inflight message state.

  In extreme corner case where we sent message to FSP via fsp_sync_msg() path and FSP Reset/Reload happens before getting respose from FSP, then we will endup waiting in fsp_sync_msg() until everything becomes normal.

  This patch adds fsp_in_rr() check to fsp_sync_msg() and return error to caller if FSP is in R/R.

- platforms/ibm-fsp/firenze: Fix PCI slot power-off pattern

  When powering off the PCI slot, the corresponding bits should be set to 0bxx00xx00 instead of 0bxx11xx11. Otherwise, the specified PCI slot can't be put into power-off state. Fortunately, it didn't introduce any side-effects so far.

5.1.73 skiboot-5.4.8

skiboot-5.4.8 was released on Wednesday October 11th, 2017. It replaces *skiboot-5.4.7* as the current stable release in the 5.4.x series.

Over *skiboot-5.4.7*, we have a few bug fixes for FSP platforms:

- libflash/file: Handle short read()s and write()s correctly

  Currently we don't move the buffer along for a short read() or write() and nor do we request only the remaining amount.

- FSP/NVRAM: Handle "get vNVRAM statistics" command

  FSP sends MBOX command (cmd : 0xEB, subcmd : 0x05, mod : 0x00) to get vNVRAM statistics. OPAL doesn't maintain any such statistics. Hence return FSP_STATUS_INVALID_SUBCMD.

  Sample OPAL log:

  ```
  [16944.384670488,3] FSP: Unhandled message eb0500
  [16944.474110465,3] FSP: Unhandled message eb0500
  [16945.111280784,3] FSP: Unhandled message eb0500
  [16945.293393485,3] FSP: Unhandled message eb0500
  ```

- FSP/CONSOLE: Limit number of error logging

  Commit c8a7535f (FSP/CONSOLE: Workaround for unresponsive ipmi daemon, added in skiboot 5.4.6 and 5.7-rc1) added error logging when buffer is full. In some corner cases kernel may call this function multiple time and we may endup logging error again and again.

  This patch fixes it by generating error log only once.

- FSP/CONSOLE: Fix fsp_console_write_buffer_space() call

  Kernel calls fsp_console_write_buffer_space() to check console buffer space availability. If there is enough buffer space to write data, then kernel will call fsp_console_write() to write actual data.

  In some extreme corner cases (like one explained in commit c8a7535f) console becomes full and this function returns 0 to kernel (or space available in console buffer < next incoming data size). Kernel will continue retrying until it gets enough space. So we will start seeing RCU stalls.

  This patch keeps track of previous available space. If previous space is same as current means not enough space in console buffer to write incoming data. It may be due to very high console write operation and slow response from FSP -OR- FSP has stopped processing data (ex: because of ipmi daemon died). At this point we will start timer with timeout of SER_BUFFER_OUT_TIMEOUT (10 secs). If situation is not improved within 10 seconds means something went bad. Lets return OPAL_RESOURCE so that kernel can drop console write and continue.

- FSP/CONSOLE: Close SOL session during R/R

  Presently we are not closing SOL and FW console sessions during R/R. Host will continue to write to SOL buffer during FSP R/R. If there is heavy console write operation happening during FSP R/R (like running *top* command inside console), then at some point console buffer becomes full. fsp_console_write_buffer_space() returns 0 (or less than required space to write data) to host. While one thread is busy writing to console, if some other threads tries to write data to console we may see RCU stalls (like below) in kernel.

  kernel call trace:

  ```
  [ 2082.828363] INFO: rcu_sched detected stalls on CPUs/tasks: { 32} (detected by␣
  ↪16, t=6002 jiffies, g=23154, c=23153, q=254769)
  [ 2082.828365] Task dump for CPU 32:
  [ 2082.828368] kworker/32:3    R  running task        0  4637      2 0x00000884
  [ 2082.828375] Workqueue: events dump_work_fn
  [ 2082.828376] Call Trace:
  [ 2082.828382] [c000000f1633fa00] [c00000000013b6b0] console_unlock+0x570/0x600␣
  ↪(unreliable)
  [ 2082.828384] [c000000f1633fae0] [c00000000013ba34] vprintk_emit+0x2f4/0x5c0
  [ 2082.828389] [c000000f1633fb60] [c00000000099e644] printk+0x84/0x98
  ```

(continues on next page)

(continued from previous page)

```
[ 2082.828391] [c000000f1633fb90] [c0000000000851a8] dump_work_fn+0x238/0x250
[ 2082.828394] [c000000f1633fc60] [c0000000000ecb98] process_one_work+0x198/0x4b0
[ 2082.828396] [c000000f1633fcf0] [c0000000000ed3dc] worker_thread+0x18c/0x5a0
[ 2082.828399] [c000000f1633fd80] [c0000000000f4650] kthread+0x110/0x130
[ 2082.828403] [c000000f1633fe30] [c000000000009674] ret_from_kernel_thread+0x5c/
↪0x68
```

Hence lets close SOL (and FW console) during FSP R/R.

- FSP/CONSOLE: Do not associate unavailable console

  Presently OPAL sends associate/unassociate MBOX command for all FSP serial console (like below OPAL message). We have to check console is available or not before sending this message.

  OPAL log:

```
[ 5013.227994012,7] FSP: Reassociating HVSI console 1
[ 5013.227997540,7] FSP: Reassociating HVSI console 2
```

- FSP: Disable PSI link whenever FSP tells OPAL about impending Reset/Reload

  Commit 42d5d047 fixed scenario where DPO has been initiated, but FSP went into reset before the CEC power down came in. But this is generic issue that can happen in normal shutdown path as well.

  Hence disable PSI link as soon as we detect FSP impending R/R.

- fsp: return OPAL_BUSY_EVENT on failure sending FSP_CMD_POWERDOWN_NORM Also, return OPAL_BUSY_EVENT on failure sending FSP_CMD_REBOOT / DEEP_REBOOT.

  We had a race condition between FSP Reset/Reload and powering down the system from the host:

  Roughly:

#	FSP	Host
1	Power on	
2		Power on
3	(inject EPOW)	
4	(trigger FSP R/R)	
5		Processes EPOW event, starts shutting down
6		calls OPAL_CEC_POWER_DOWN
7	(is still in R/R)	
8		gets OPAL_INTERNAL_ERROR, spins in opal_poll_events
9	(FSP comes back)	
10		spinning in opal_poll_events
11	(thinks host is running)	

  The call to OPAL_CEC_POWER_DOWN is only made once as the reset/reload error path for fsp_sync_msg() is to return -1, which means we give the OS OPAL_INTERNAL_ERROR, which is fine, except that our own API docs give us the opportunity to return OPAL_BUSY when trying again later may be successful, and we're ambiguous as to if you should retry on OPAL_INTERNAL_ERROR.

  For reference, the linux code looks like this:

```
static void __noreturn pnv_power_off(void)
{
        long rc = OPAL_BUSY;

```

(continues on next page)

(continued from previous page)

```
        pnv_prepare_going_down();

        while (rc == OPAL_BUSY || rc == OPAL_BUSY_EVENT) {
                rc = opal_cec_power_down(0);
                if (rc == OPAL_BUSY_EVENT)
                        opal_poll_events(NULL);
                else
                        mdelay(10);
        }
        for (;;)
                opal_poll_events(NULL);
}
```

Which means that *practically* our only option is to return OPAL_BUSY or OPAL_BUSY_EVENT.

We choose OPAL_BUSY_EVENT for FSP systems as we do want to ensure we're running pollers to communicate with the FSP and do the final bits of Reset/Reload handling before we power off the system.

5.1.74 skiboot-5.4.9

skiboot-5.4.9 was released on Friday January 5th, 2018. It replaces *skiboot-5.4.8* as the current stable release in the 5.4.x series.

Over *skiboot-5.4.8*, we have one new feature:

- Parse IPL FW feature settings

  Add parsing for the firmware feature flags in the HDAT. This indicates the settings of various parameters which are set at IPL time by firmware.

5.1.75 skiboot-5.5.0

skiboot-5.5.0 was released on Friday April 7th 2017. It is the new stable release of skiboot, taking over from the 5.4 release, first released on November 11th 2016.

skiboot-5.5.0 contains all bug fixes as of *skiboot-5.4.3* and *skiboot-5.1.19* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

This release is a good level set of POWER9 support for bringup activities. If you are doing bringup, it is strongly suggested you continue to follow skiboot master.

After skiboot 5.5.0, we move to a regular six week release cycle, similar to op-build, but slightly offset to allow for a short stabilisation period. Expected release dates and contents are tracked using GitHub milestone and issues: https://github.com/open-power/skiboot/milestones

Changes in skiboot-5.5.0

See changes in the release candidates:

- *skiboot-5.5.0-rc1*
- *skiboot-5.5.0-rc2*
- *skiboot-5.5.0-rc3*

Changes since skiboot-5.5.0-rc3

- hdat: parse processor attached i2c devices

  Adds basic parsing for i2c devices that are attached to the processor I2C interfaces. This is mainly VPD SEEP-ROMs.

- libflash/blocklevel: Add blocklevel_smart_erase()

  With recent changes to flash drivers in linux not all erase blocks are 4K anymore. While most level of the pflash/gard tool stacks were written to not mind, it turns out there are bugs which means not 4K erase block backing stores aren't handled all that well. Part of the problem is the FFS layout that is 4K aligned and with larger block sizes pflash and the gard tool don't check if their erase commands are erase block aligned - which they are usually not with 64K erase blocks.

  This patch aims to add common functionality to blocklevel so that (at least) pflash and the gard tool don't need to worry about the problem anymore.

- external/pflash: Use blocklevel_smart_erase()
- external/gard: Use blocklevel_smart_erase()
- libstb/create-container: Add full container build and sign with imprint keys

  This adds support for writing all the public key and signature fields to the container header, and for dumping the prefix and software headers so they may may be signed, and for signing those headers with the imprint keys.

- asm: do not set SDR1 on POWER9. This register does not exist in ISAv3.

Testing:

- mambo: Allow setting the Linux command line from the environment

  For automated testing it's helpful to be able to set the Linux command line via an environment variable.

- mambo: Add util function for breaking on console output

Contributors

Processed 408 csets from 31 developers

3 employers found

A total of 24073 lines added, 16759 removed (delta 7314)

Extending the analysis done for the last few releases, we can see our trends in code review across versions:

Release	csets	Ack	Reviews	Tested	Reported
5.0	329	15	20	1	0
5.1	372	13	38	1	4
5.2-rc1	334	20	34	6	11
5.3-rc1	302	36	53	4	5
5.4.0	361	16	28	1	9
5.5.0	408	11	48	14	10

I am absolutely *thrilled* as to the uptick of reviews and tested-by occuring over our 5.4.0 release. Although we are not yet back up to 5.3 era levels for review, we're much closer. For tested-by, we've set a new record, which is excellent!

Developers with the most changesets

Developer	#	%
Benjamin Herrenschmidt	139	(34.1%)
Stewart Smith	60	(14.7%)
Oliver O'Halloran	54	(13.2%)
Gavin Shan	23	(5.6%)
Michael Neuling	20	(4.9%)
Vasant Hegde	15	(3.7%)
Cyril Bur	15	(3.7%)
Claudio Carvalho	14	(3.4%)
Andrew Donnellan	11	(2.7%)
Ananth N Mavinakayanahalli	9	(2.2%)
Alistair Popple	6	(1.5%)
Nicholas Piggin	5	(1.2%)
Cédric Le Goater	5	(1.2%)
Pridhiviraj Paidipeddi	5	(1.2%)
Michael Ellerman	4	(1.0%)
Shilpasri G Bhat	4	(1.0%)
Russell Currey	3	(0.7%)
Jack Miller	2	(0.5%)
Chris Smart	2	(0.5%)
Dave Heller	1	(0.2%)
Akshay Adiga	1	(0.2%)
Reza Arbab	1	(0.2%)
Matt Brown	1	(0.2%)
Frederic Barrat	1	(0.2%)
Hank Chang	1	(0.2%)
Willie Liauw	1	(0.2%)
Werner Fischer	1	(0.2%)
Jeremy Kerr	1	(0.2%)
Patrick Williams	1	(0.2%)
Joel Stanley	1	(0.2%)
Alexey Kardashevskiy	1	(0.2%)

Developers with the most changed lines

Developer	#	%
Oliver O'Halloran	18278	(48.5%)
Benjamin Herrenschmidt	5512	(14.6%)
Cyril Bur	3184	(8.4%)
Alistair Popple	3102	(8.2%)
Stewart Smith	2757	(7.3%)
Gavin Shan	802	(2.1%)
Ananth N Mavinakayanahalli	544	(1.4%)
Claudio Carvalho	489	(1.3%)
Dave Heller	425	(1.1%)
Willie Liauw	361	(1.0%)
Andrew Donnellan	315	(0.8%)

Continued on next page

Table 10 – continued from previous page

Developer	#	%
Michael Neuling	290	(0.8%)
Vasant Hegde	253	(0.7%)
Shilpasri G Bhat	228	(0.6%)
Nicholas Piggin	222	(0.6%)
Reza Arbab	198	(0.5%)
Russell Currey	158	(0.4%)
Jack Miller	127	(0.3%)
Cédric Le Goater	126	(0.3%)
Chris Smart	95	(0.3%)
Akshay Adiga	57	(0.2%)
Hank Chang	56	(0.1%)
Pridhiviraj Paidipeddi	47	(0.1%)
Michael Ellerman	29	(0.1%)
Matt Brown	29	(0.1%)
Alexey Kardashevskiy	2	(0.0%)
Frederic Barrat	1	(0.0%)
Werner Fischer	1	(0.0%)
Jeremy Kerr	1	(0.0%)
Patrick Williams	1	(0.0%)
Joel Stanley	1	(0.0%)

Developers with the most lines removed

Developer	#	%
Oliver O'Halloran	8516	(50.8%)
Werner Fischer	1	(0.0%)

Developers with the most signoffs

Total: 364

Developer	#	%
Stewart Smith	348	(95.6%)
Michael Neuling	6	(1.6%)
Oliver O'Halloran	3	(0.8%)
Benjamin Herrenschmidt	2	(0.5%)
Vaidyanathan Srinivasan	1	(0.3%)
Hank Chang	1	(0.3%)
Jack Miller	1	(0.3%)
Gavin Shan	1	(0.3%)
Alistair Popple	1	(0.3%)

Developers with the most reviews

Total 50

Developer	#	%
Vasant Hegde	14	(28.0%)
Andrew Donnellan	9	(18.0%)
Russell Currey	6	(12.0%)
Cédric Le Goater	5	(10.0%)
Oliver O'Halloran	4	(8.0%)
Vaidyanathan Srinivasan	3	(6.0%)
Gavin Shan	3	(6.0%)
Alistair Popple	2	(4.0%)
Frederic Barrat	2	(4.0%)
Mahesh Salgaonkar	1	(2.0%)
Cyril Bur	1	(2.0%)

Developers with the most test credits

Total 14

Developer	#	%
Willie Liauw	4	(28.6%)
Mark E Schreiter	3	(21.4%)
Claudio Carvalho	3	(21.4%)
Gavin Shan	1	(7.1%)
Michael Neuling	1	(7.1%)
Pridhiviraj Paidipeddi	1	(7.1%)
Chris Smart	1	(7.1%)

Developers who gave the most tested-by credits

Total 14

Developer	#	%
Gavin Shan	7	(50.0%)
Stewart Smith	4	(28.6%)
Chris Smart	1	(7.1%)
Oliver O'Halloran	1	(7.1%)
Ananth N Mavinakayanahalli	1	(7.1%)

Developers with the most report credits

Total 10

Developer	#	%
Hank Chang	4	(40.0%)
Mark E Schreiter	3	(30.0%)
Guilherme G. Piccoli	1	(10.0%)
Colin Ian King	1	(10.0%)
Pradipta Ghosh	1	(10.0%)

Developers who gave the most report credits

Total 10

Developer	#	%
Gavin Shan	8	(80.0%)
Andrew Donnellan	1	(10.0%)
Jeremy Kerr	1	(10.0%)

Top changeset contributors by employer

Employer	#	%
IBM	406	(99.5%)
SuperMicro	1	(0.2%)
Thomas-Krenn AG	1	(0.2%)

Top lines changed by employer

Employer	#	%
IBM	37329	(99.0%)
SuperMicro	361	(1.0%)
Thomas-Krenn AG	1	(0.0%)

Employers with the most signoffs

Total 364

Employer	#	%
IBM	363	(99.7%)
(Unknown)	1	(0.3%)

Employers with the most hackers

Total 31

Employer	#	%
IBM	29	(93.5%)
Thomas-Krenn AG	1	(3.2%)
SuperMicro	1	(3.2%)

5.1.76 skiboot-5.5.0-rc1

skiboot-5.5.0-rc1 was released on Tuesday March 28th 2017. It is the first release candidate of skiboot 5.5, which will become the new stable release of skiboot following the 5.4 release, first released November 11th 2016.

skiboot-5.5.0-rc1 contains all bug fixes as of *skiboot-5.4.3* and *skiboot-5.1.19* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.5.0 by April 8th, with skiboot 5.5.0 being for all POWER8 and POWER9 platforms in op-build v1.16 (Due April 12th). This is a short cycle as this release is mainly targetted towards POWER9 bringup efforts.

Following skiboot-5.5.0, we will move to a regular six week release cycle, similar to op-build, but slightly offset to allow for a short stabilisation period. Expected release dates and contents are tracked using GitHub milestone and issues: https://github.com/open-power/skiboot/milestones

Over skiboot-5.4, we have the following changes:

New Platforms

- SuperMicro's (SMC) P8DNU: An astbmc based POWER8 platform
- Add a generic platform to help with bringup of new systems.
- Four POWER9 based systems (NOTE: All POWER9 systems should be considered for bringup use only at this point):
  - Romulus
  - Witherspoon (a POWER9 system with NVLink2 attached GPUs)
  - Zaius (OpenCompute platform, also known as "Barreleye 2")
  - ZZ (FSP based system)

New features

- System reset IPI facility and Mambo implementation Add an opal call *OPAL_SIGNAL_SYSTEM_RESET* which allows system reset exceptions to be raised on other CPUs and act as an NMI IPI. There is an initial simple Mambo implementation, but allowances are made for a more complex hardware implementation.

  The Mambo implementation is based on the RFC implementation for POWER8 hardware (see https://patchwork.ozlabs.org/patch/694794/) which we hope makes it into a future release.

  This implements an in-band NMI equivalent.
- add CONTRIBUTING.md, ensuring that people new to the project have a one-stop place to find out how to get started.
- interrupts: Add optional name for OPAL interrupts

  This adds the infrastructure for an interrupt source to provide a name for an interrupt directed toward OPAL. Those names will be put into an "opal-interrupts-names" property which is a standard DT string list corresponding 1:1 with the "opal-interrupts" property. PSI interrupts get names, and this is visible in Linux through /proc/interrupts
- platform: add OPAL_REBOOT_FULL_IPL reboot type

  There may be circumstances in which a user wants to force a full IPL reboot rather than using fast reboot. Add a new reboot type, OPAL_REBOOT_FULL_IPL, that disables fast reboot. On platforms which don't support fast reboot, this will be equivalent to a normal reboot.

- phb3: Trick to allow control of the PCIe link width and speed

  This implements a hook inside OPAL that catches 16 and 32 bit writes to the link status register of the PHB.

  It allows you to write a new speed or a new width, and OPAL will then cause the PHB to renegociate.

  Example:

  First read the link status on PHB4:

  ```
  setpci -s 0004:00:00.0 0x5a.w
  a103
  ```

  It's at x16 Gen3 speed (8GT/s)

  bits 0x0ff0 are the width and 0x000f the speed. The width can be 1 to 16 and the speed 1 to 3 (2.5, 5 and 8GT/s)

  Then try to bring it down to 1x Gen1 :

  ```
  setpci -s 0004:00:00.0 0x5a.w=0xa011
  ```

  Observe the result in the PHB:

  ```
  / # lspci -s 0004:00:00.0 -vv
  0004:00:00.0 PCI bridge: IBM Device 03dc (prog-if 00 [Normal decode])
  .../...
  LnkSta: Speed 2.5GT/s, Width x1, TrErr- Train- SlotClk- DLActive+ BWMgmt-␣
  ↪ABWMgmt+
  ```

  And in the device:

  ```
  / # lspci -s 0004:01:00.0 -vv
  .../...
  LnkSta: Speed 2.5GT/s, Width x1, TrErr- Train- SlotClk+ DLActive- BWMgmt-␣
  ↪ABWMgmt-
  ```

- core/init: Add hdat-map property to OPAL node.

  Exports the HDAT heap to the OS. This allows the OS to view the HDAT heap directly. This allows us to view the HDAT area without having to use getmemproc.

- Add a generic platform: If /bmc in device tree, attempt to init one For the most part, this gets us somewhere on some OpenPOWER systems before there's a platform file for that machine.

  Useful in bringup only, and marked as such with scary looking log messages.

Core

- asm: Don't try to set LPCR:LPES1 on P8 and P9, the bit doesn't exist.
- pci: Add a framework for quirks

  In future we may want to be able to do fixups for specific PCI devices in skiboot, so add a small framework for doing this.

  This is not intended for the same purposes as quirks in the Linux kernel, as the PCI devices that quirks can match for in skiboot are not properly configured. This is intended to enable having a custom path to make changes that don't directly interact with the PCI device, for example adding device tree entries.

- hw/slw: fix possible NULL dereference

- slw: Print enabled stop states on boot
- uart: Fix Linux pass-through policy, provide NVRAM override option
- libc/stdio/vsnprintf.c: add explicit fallthrough, this silences a recent (GCC 7.x) warning
- init: print the FDT blob size in decimal
- init: Print some more info before booting linux

  The kernel command line from nvram and the stdout-path are useful to know when debugging console related problems.
- Makefile: Disable stack protector due to gcc problems

  Depending on how it was built, gcc will use the canary from a global (works for us) or from the TLS (doesn't work for us and accesses random stuff instead).

  Fixing that would be tricky. There are talks of adding a gcc option to force use of globals, but in the meantime, disable the stack protector.
- Stop using 3-operand cmp[l][i] for latest binutils Since a5721ba270, binutils does not support 3-operand cmp[l][i]. This adds (previously optional) parameter L.
- buddy: Add a simple generic buddy allocator
- stack: Don't recurse into __stack_chk_fail
- Makefile: Use -ffixed-r13 We use r13 for our own stuff, make sure it's properly fixed
- Always set ibm,occ-functional-state correctly
- psi: fix the xive registers initialization on P8, which seems to be fine for real HW but causes a lof of pain under qemu
- slw: Set PSSCR value for idle states
- Limit number of "Poller recursion detected" errors to display

  In some error conditions, we could spiral out of control on this and spend all of our time printing the exact same backtrace.

  Limit it to 16 times, because 16 is a nice number.
- slw: do SLW timer testing while holding xscom lock

  We add some routines that let a caller get the xscom lock once and then do a bunch of xscoms while holding it. In some situations without this, it could take long enough to get the xscom lock that the 1ms timeout would expire and we'd falsely think the SLW timer didn't work when in fact it did.
- wait_for_resource_loaded: don't needlessly sleep for 5ms
- run pollers in cpu_process_local_jobs() if running job synchonously
- fsp: Don't recurse pollers in ibm_fsp_terminate
- chiptod: More hardening against -1 chip ID
- interrupts: Rewrite/correct doc for opal_set/get_xive
- cpu: Don't enable nap mode/PM mode on non-P8
- platform: Call generic platform probe and init UART there
- psi: Don't register more interrupts than the HW supports
- psi: Add DT option to disable LPC interrupts

I2C and TPM

- p8i2c: Use calculated poll_interval when booting OPAL Otherwise we'd default to 2seconds (TIMER_POLL) during boot on chips with a functional i2c interrupt, leading to slow i2c during boot (or hitting timeouts instead).
- i2c: Add i2c_run_req() to crank the state machine for a request
- tpm_i2c_nuvoton: work out the polling time using mftb()
- tpm_i2c_nuvoton: handle errors after reading the tpm fifo
- tpm_i2c_nuvoton: cleanup variables in tpm_read_fifo()
- tpm_i2c_nuvoton: handle errors after writting the tpm fifo
- tpm_i2c_nuvoton: cleanup variables in tpm_write_fifo()
- tpm_i2c_nuvoton: handle errors after writing sts.commandReady in step 5
- tpm_i2c_nuvoton: handle errors after writing sts.go
- tpm_i2c_nuvoton: handle errors after checking the tpm fifo status
- tpm_i2c_nuvoton: return burst_count in tpm_read_burst_count()
- tpm_i2c_nuvoton: isolate the code that handles the TPM_TIMEOUT_D timeout
- tpm_i2c_nuvoton: handle errors after reading sts.commandReady
- tpm_i2c_nuvoton: add tpm_status_read_byte()
- tpm_i2c_nuvoton: add tpm_check_status()
- tpm_i2c_nuvoton: rename defines to shorter names
- tpm_i2c_interface: decouple rc from being done with i2c request
- tpm_i2c_interface: set timeout before each request
- i2c: Add nuvoton quirk, disallowing i2cdetect as it locks TPM

  p8-i2c reset things manually in some error conditions
- stb: create-container and wrap skiboot in Secure/Trusted Boot container

  We produce **UNSIGNED** skiboot.lid.stb and skiboot.lid.xz.stb as build artifacts.

  These are suitable blobs for flashing onto Trusted Boot enabled op-build builds *WITH* the secure boot jumpers *ON* (i.e. *NOT* in secure mode). It's just enough of the Secure and Trusted Boot container format to make Hostboot behave.

PCI

- core/pci: Support SRIOV VFs

  Currently, skiboot can't see SRIOV VFs. It introduces some troubles as I can see: The device initialization logic (phb->ops->device_init()) isn't applied to VFs, meaning we have to maintain same and duplicated mechanism in kernel for VFs only. It introduces difficulty to code maintaining and prone to lose sychronization.

  This was motivated by bug reported by Carol: The VF's Max Payload Size (MPS) isn't matched with PF's on Mellanox's adapter even kernel tried to make them same. It's caused by readonly PCIECAP_EXP_DEVCTL register on VFs. The skiboot would be best place to emulate this bits to eliminate the gap as I can see.

  This supports SRIOV VFs. When the PF's SRIOV capability is populated, the number of maximal VFs (struct pci_device) are instanciated, but but not usable yet. In the mean while, PCI config register filter is registered against PCIECAP_SRIOV_CTRL_VFE to capture the event of enabling or disabling VFs. The VFs are initialized, put into the PF's children list (pd->children), populate its PCI capabilities, and register PCI config register filter against PCICAP_EXP_DEVCTL. The filter's handler caches what is written to MPS field and returns the cached value on read, to eliminate the gap mentioned as above.

- core/pci: Avoid hreset after freset

  Commit 5ac71c9 ("pci: Avoid hot resets at boot time") missed to avoid hot reset after fundamental reset for PCIe common slots.

  This fixes it.

- core/pci: Enforce polling PCIe link in hot-add path

  In surprise hot-add path, the power state isn't changed on hardware. Instead, we set the cached power state (@slot->power_state) and return OPAL_SUCCESS. The upper layer starts the PCI probing immediately when receiving OPAL_SUCCESS. However, the PCIe link behind the PCI slot is likely down. Nothing will be probed from the PCI slot even we do have PCI adpater connected to the slot.

  This fixes the issue by returning OPAL_ASYNC_COMPLETION to force upper layer to poll the PCIe link before probing the PCI devices behind the slot in surprise and managed hot-add paths.

- **hw/phb3: fix error handling in complete reset** During a complete reset, when we get a timeout waiting for pending transaction in state PHB3_STATE_CRESET_WAIT_CQ, we mark the PHB as permanently broken.

  Set the state to PHB3_STATE_FENCED so that the kernel can retry the complete reset.

- phb3: Lock the PHB on set_xive callbacks

p8dnu platform

- astbmc/p8dnu: Enable PCI slot's power supply on PEX9733 in hot-add path
- astbmc/p8dnu: Enable PCI slot's power supply on PEX8718 in hot-add path
- core/pci: Mark broken PDC on slots without surprise hotplug capability

  We has to support surprise hotplug on PCI slots that don't support it on hardware. So we're fully utilizing the PCIe link state change event to detect the events (hot-remove and hot-add). The PDC (Presence Detection Change) event isn't reliable for the purpose. For example, PEX8718 on superMicro's machines.

  This adds another PCI slot property "ibm,slot-broken-pdc" in the device-tree, to indicate the PDC isn't reliable on those (software claimed) surprise pluggable slots.

- core/pci: Fix PCIe slot's presence

  According to PCIe spec, the presence bit is hardcoded to 1 if PCIe switch downstream port doesn't support slot capability. The register used for the check in pcie_slot_get_presence_state() is wrong. It should be PCIe capability register instead of PCIe slot capability register. Otherwise, we always have present bit on the PCI topology. The issue is found on Supermicro's p8dtu2u machine:

```
# lspci -t
-+-[0022:00]---00.0-[01-08]----00.0-[02-08]--+-01.0-[03]----00.0
 |                                            \-02.0-[04-08]--
# cat /sys/bus/pci/slots/S002204/adapter
1
# lspci -vvs 0022:02:02.0
# lspci -vvs 0022:02:02.0
0022:02:02.0 PCI bridge: PLX Technology, Inc. PEX 8718 16-Lane, \
5-Port PCI Express Gen 3 (8.0 GT/s) Switch (rev ab) (prog-if 00 [Normal decode])
```

(continues on next page)

(continued from previous page)

```
    :
 Capabilities: [68] Express (v2) Downstream Port (Slot+), MSI 00
    :
    SltSta:    Status: AttnBtn- PowerFlt- MRL- CmdCplt- PresDet- Interlock-
               Changed: MRL- PresDet- LinkState-

This fixes the issue by checking the correct register (PCIe capability).
Also, the register's value is cached in advance as we did for slot and
link capability.
```

- core/pci: More reliable way to update PCI slot power state

  The power control bit (SLOT_CTL, offset: PCIe cap + 0x18) isn't reliable enough to reflect the PCI slot's power state. Instead, the power indication bits are more reliable comparatively. This leads to mismatch between the cached power state and PCI slot's presence state, resulting in the hotplug driver in kernel refuses to unplug the devices properly on the request. The issue was found on below NVMe card on "supermicro,p8dtu2u" machine. We don't have this issue on the integrated PLX 8718 switch.

```
 # lspci
 0022:01:00.0 PCI bridge: PLX Technology, Inc. PEX 9733 33-lane, \
              9-port PCI Express Gen 3 (8.0 GT/s) Switch (rev aa)
 0022:02:01.0 PCI bridge: PLX Technology, Inc. PEX 9733 33-lane, \
              9-port PCI Express Gen 3 (8.0 GT/s) Switch (rev aa)
 0022:02:04.0 PCI bridge: PLX Technology, Inc. PEX 9733 33-lane, \
              9-port PCI Express Gen 3 (8.0 GT/s) Switch (rev aa)
 0022:02:05.0 PCI bridge: PLX Technology, Inc. PEX 9733 33-lane, \
              9-port PCI Express Gen 3 (8.0 GT/s) Switch (rev aa)
 0022:02:06.0 PCI bridge: PLX Technology, Inc. PEX 9733 33-lane, \
              9-port PCI Express Gen 3 (8.0 GT/s) Switch (rev aa)
 0022:02:07.0 PCI bridge: PLX Technology, Inc. PEX 9733 33-lane, \
              9-port PCI Express Gen 3 (8.0 GT/s) Switch (rev aa)
 0022:17:00.0 Non-Volatile memory controller: Device 19e5:0123 (rev 45)

This updates the cached PCI slot's power state using the power
indication bits instead of power control bit, to fix above issue.
```

Utilities

- opal-prd: Direct systemd to always restart opal-prd Always restart the opal-prd daemon, irrespective of why it stopped.
- external/ffspart: Simple C program to be able to make an FFS partition
- getscom: Add chip info for P9.
- gard: Fix make dist target
- pflash/libflash: arch_flash_arm: Don't assume mtd labels are short

libffs

- libffs: Understand how to create FFS partition TOCs and entries.

BMC Based systems

- platforms/astbmc: Support PCI slots for palmetto
- habanero/slottable: Remove Network Mezz(2, 0) from PHB1.
- BMC/PCI: Check slot tables against detected devices On BMC machines, we have slot tables of built in PHBs, slots and devices that are physically present in the system (such as the BMC itself). We can use these tables to check what we *detected* against what *should* be in the system and throw an error if they differ.

  We have seen this occur a couple of times while still booting, giving the user just an empty petitboot screen and not much else to go on. This patch helps in that we get a skiboot error message, and at some point in the future when we pump them up to the OS we could get a big friendly error message telling you you're having a bad day.
- pci/quirk: Populate device tree for AST2400 VGA

  Adding these properties enables the kernel to function in the same way that it would if it could no longer access BMC configuration registers through a backdoor, which may become the default in future.

  The comments describe how isolating the host from the BMC could be achieved in skiboot, assuming all kernels that the system boots support this. Isolating the BMC and the host from each other is important if they are owned by different parties; for example, a cloud provider renting machines "bare metal".
- astbmc/pnor: Use mbox-flash for flash accesses

  If the BMC is MBOX protocol aware, request flash reads/writes over the MBOX regs. This inits the blocklevel for pnor access with mbox-flash.
- ast: Account for differences between 2400 vs 2500
- platform: set default bmc_platform The bmc_platform pointer is set to NULL by default and on non-AMI BMC platforms. As a result a few places in hw/ipmi/ipmi-sel.c will blindly dereference a NULL pointer.

POWER9

- external: Update xscom utils for type 1 indirect accesses
- xscom: Harden indirect writes
- xscom: Add POWER9 scom reset
- homer : Enable HOMER region reservation for POWER9
- slw: Define stop idle states for P9 DD1
- slw: Fix parsing of supported STOP states
- slw: only enable supported STOP states
- dts: add support for p9 cores
- asm: Add POWER9 case to init_shared_sprs

  For now, setup the HID and HMEER. We'll add more as we get good default values from HW.
- xive/psi/lpc: Handle proper clearing of LPC SerIRQ latch on POWER9 DD1
- lpc: Mark the power9 LPC bus as compatible with power8
- Fix typo in PIR mask for POWER9. Fixes booting multi-chip.
- vpd: add vpd_valid() to check keyword VPD blobs

  Adds a function to check whether a blob is a valid IBM ASCII keyword VPD blob. This allows us to recognise when we do and do not have a VPD blob and act accordingly.

- core/cpu.c: Use a device-tree node to detect nest mmu presence The nest mmu address scom was hardcoded which could lead to boot failure on POWER9 systems without a nest mmu. For example Mambo doesn't model the nest mmu which results in failure when calling opal_nmmu_set_ptcr() during kernel load.
- psi: Fix P9 BAR setup on multi-chips

PHB4:

- phb4: Fix TVE encoding for start address
- phb4: Always assign powerbus BARs

  HostBoot configure them with weird values that confuse us, instead let's just own the assignment. This is temporary, I will centralize memory map management next but this gets us going.
- phb4: Fix endian issue with link control2/status2 registers Fixes training at larger than PCIe Gen1 speeds.
- phb4: Add ability to log config space access Useful for debugging
- phb4: Change debug prints Currently we print "PHB4" and mean either "PHB version 4" or "PHB number 4" which can be quite confusing.
- phb4: Fix config space enable bits on DD1
- phb4: Fix location of EEH enable bits
- phb4: Fix setting of max link speed
- phb4: Updated inits as of PHB4 spec 0.52

HDAT fixes:

- hdat: Parse BMC nodes much earlier

  This moves the parsing of the BMC and LPC details to the start of the HDAT parsing. This allows us to enable the Skiboot log console earlier so we can get debug output while parsing the rest of the HDAT.
- astbmc: Don't do P8 PSI or DT fixups on P9

  Previously the HDAT format was only ever used with IBM hardware so it would store vital product data (VPD) blobs in the IBM ASCII Keyword VPD format. With P9 HDAT is used on OpenPower machines which use Industry Standard DIMMs that provide their product data through a "Serial Present Detect" EEPROM mounted on the DIMM.

  The SPD blob has a different format and is exported in the device-tree under the "spd" property rather than the "ibm,vpd" property. This patch adds support for recognising these blobs and placing them in the appropriate DT property.
- hdat: Add __packed to all HDAT structures and workaround HB reserve

  Some HDAT structures aren't properly aligned. We were using __packed on some but not others and got at least one wrong (HB reserve). This adds it everywhere to avoid such problems.

  However this then triggers another problem where HB gives us a crazy range (0.256M) to reserve with no label, which triggers an assertion failure later on in mem_regions.c.

  So also add a test to skip any region starting at 0 until we can undertand that better and have it fixed one way or another.
- hdat: Ignore broken memory reserves

  Ignore HDAT memory reserves > 512MB. These are considered bogus and workaround known HDAT bugs.
- hdat: Add BMC device-tree node for P9 OpenPOWER systems

- hdat: Fix interrupt & device_type of UART node

  The interrupt should use a standard "interrupts" property. The UART node also need a device_type="serial" property for historical reasons otherwise Linux won't pick it up.

- parse and export STOP levels
- add new sppcrd_chip_info fields
- add radix-AP-encodings
- stop using proc_int_line in favor of pir
- rename add_icp() to add_xics_icp()
- Add support for PHB4
- create XIVE nodes under each xscom node
- Add P9 compatible property
- Parse hostboot memory reservations from HDAT
- Add new fields to IPL params structure and update sys family for p9.
- Fix ibm,pa-features for all CPU types
- Fix XSCOM nodes for P9
- Remove deprecated 'ibm, mem-interleave-scope' from DT on POWER9
- Grab system model name from HDAT when available
- Grab vendor information from HDAT when available
- SPIRA-H/S changes for P9
- Add BMC and LPC IOPATH support
- handle ISDIMM SPD blobs
- make HDIF_child() print more useful errors
- Add PSI HB xscom details
- Add new fields to proc_init_data structure
- Add processor version check for hs service ntuple
- add_iplparams_serial - Validate HDIF_get_iarray_size() return value

XIVE:

The list of XIVE fixes and updates is extensive. Below is only a portion of the changes that have gone into skiboot 5.5.0-rc1 for the new XIVE hardware that is present in POWER9:

- xive: Enable backlog on queues
- xive: Use for_each_present_cpu() for setting up XIVE
- xive: Fix logic in opal_xive_get_xirr()
- xive: Properly initialize new VP and EQ structures
- xive: Improve/fix EOI of LSIs
- xive: Add FIXME comments about mask/umask races
- xive: Fix memory barrier in opal_xive_get_xirr()
- xive: Don't try to find a target EQ for prio 0xff

- xive: Bump table sizes in direct mode
- xive: Properly register escalation interrupts
- xive: Split the OPAL irq flags from the internal ones
- xive: Don't touch ESB masks unless masking/unmasking
- xive: Fix xive_get_ir_targetting()
- xive: Cleanup escalation PQ on queue change
- xive: Add *any chip* for allocating interrupts
- xive: Add chip_id to get_vp_info
- xive: Add opal_xive_get/set_vp_info
- xive: Add VP alloc/free OPAL functions
- xive: Workaround for bad DD1 checker
- xive: Add more checks for exploitation mode
- xive: Add support for EOIs via OPAL
- xive/phb4: Work around broken LSI control on P9 DD1
- xive: Forward interrupt names callback
- xive: Export opal_xive_reset() arguments in OPAL API
- xive: Add interrupt allocator
- xive: Implement xive_reset
- xive: Don't assert if xive_get_vp() fails
- xive: Expose exploitation mode DT properties
- xive: Use a constant for max# of chips
- xive: Keep track of which interrupts were ever enabled In order to speed up xive reset
- xive: Implement internal VP allocator
- xive: Add xive_get/set_queue_info
- xive: Add helpers to encode and decode VP numbers
- xive: Add API to donate pages in indirect mode
- xive: Add asynchronous cache updates and update irq targetting
- xive: Split xive_provision_cpu() and use cache watch for VP
- xive: Add cache scrub to push watch updates to memory
- xive: Mark XIVE owned EQs with a specific flag
- xive: Use an allocator for EQDs
- xive: Break assumption that block ID == chip ID
- xive/phb4: Handle bad ESB offsets in PHB4 DD1
- xive: Implement get/set_irq_config APIs
- xive: Rework xive_set_eq_info() to store all info even when masking
- xive: Implement cache watch and use it for EQs

- xive: Add locking to some API calls
- xive: Add opal_xive_get_irq_info()
- xive: Add CPU node "interrupts" properties representing the IPIs
- xive: Add basic opal_xive_reset() call and exploitation mode
- xive: Add support for escalation interrupts
- xive: OPAL API update
- xive: Add some dump facility for debugging
- xive: Document exploitation mode (Pretty much work in progress)
- xive: Indirect table entries must have top bits "type" set
- xive: Remove unused field and clarify comment
- xive: Provide a way to override some IPI sources
- xive: Add helper to retrieve an IPI trigger port
- xive: Fix IPI EOI logic in opal_xive_eoi()
- xive: Don't try to EOI a masked source
- xive: Fix comments in xive_source_set_xive()
- xive: Fix comments in xive_get_ive()
- xive: Configure forwarding ports
- xive: Fix mangling of interrupt server# in opal_get/set_xive()
- xive: Fix interrupt number mangling

Fast-reboot

- fast-reboot: creset PHBs on fast reboot On fast reboot, perform a creset of all PHBs. This ensures that any PHBs that are fenced will be working after the reboot.
- fast-reboot: Enable fast reboot with CAPI adapters in CAPI mode CAPI mode is disabled as part of OPAL_SYNC_HOST_REBOOT.
- opal/fast-reboot: set fw_progress sensor status with IPMI_FW_PCI_INIT.

CAPI

- hmi: Print CAPP FIR information when handling CAPP malfunction alerts

FSP based systems

- hw/fsp: Do not queue SP and SPCN class messages during reset/reload This could cause soft lockups if FSP reset reload was done while in OPAL During FSP R/R, the FSP is inaccessible and will lose state. Messages to the FSP are generally queued for sending later.

Tests

- **core/test/run-trace: Reduce number of samples when running under valgrind** This reduces 'make check' run time by ~10 seconds on my laptop, and just the run-trace test itself takes 15 seconds less (under valgrind).
- test/sreset_world: Kind of like Hello World, but from the SRESET vector. A regression test for the mambo implementation of OPAL_SIGNAL_SYSTEM_RESET.
- **nvram-format: Fix endian issues** NVRAM formats are always BE, so let's use the sparse annotation to catch any issues (and correct said issues).

  On LE platforms, the test was erroneously passing as with building the nvram-format code on LE we were produces an incorrect NVRAM image.
- test/hello_world: use P9MAMBO to differentiate from P8
- hdata_to_dt: Specify PVR on command line
- hdata/test: Add DTS output for the test cases
- hdata/test: strip blobs from the DT output
- mambo: add mprintf()

  mprintf() is printf(), but it goes straight to the mambo console. This allows it to be independent of Skiboot's actual console infrastructure so it can be used for debugging the console drivers and for debugging code that runs before the console is setup.
- generate-fwts-olog: add support for parsing prerror()
- **Add bitmap test** The worst test suite ever
- mambo_utils: add ascii output to hexdump
- mambo_utils: add p_str <addr> [limit]
- mambo_utils: make p return a value
- hello_world: print out full path of missing MAMBO_BINARY
- print-stb-container: Fix build on centos7
- Travis-ci improvements: - install expect on ubuntu 12.04, disable qemu on 16.04/latest - build and test more on centos7 - hello_world: run p9 mambo tests - install systemsim-p8 on centos7 - install systemsim-p8 on centos6 - install systemsim-p9 - enable fedora25 - always pull new docker image - add fedora rawhide
- Add fwts annotation for duplicate DT node entries.

  Reference bug: https://github.com/open-power/op-build/issues/751
- external/fwts: Add 'last-tag' to FWTS olog output This isn't so useful at the moment, but this will make cleaning out crufty old error definitions much easier.
- external/fwts: Add FWTS olog merge script A script to merge olog error definitions from multiple skiboot versions into a single olog JSON file. Will prompt when conflicting patterns are found to update the pattern, or add both.
- mambo: fake NVRAM support
- mambo: Add Fake NVRAM driver
- external/mambo: add shortcut to print all GPRs

Contributors

Processed 363 csets from 28 developers. A total of 18105 lines added, 16499 removed (delta 1606)

Developers with the most changesets

Developer	#	%
Benjamin Herrenschmidt	138	(38.0%)
Stewart Smith	56	(15.4%)
Oliver O'Halloran	47	(12.9%)
Michael Neuling	18	(5.0%)
Gavin Shan	15	(4.1%)
Claudio Carvalho	14	(3.9%)
Vasant Hegde	11	(3.0%)
Cyril Bur	11	(3.0%)
Andrew Donnellan	11	(3.0%)
Ananth N Mavinakayanahalli	5	(1.4%)
Cédric Le Goater	5	(1.4%)
Pridhiviraj Paidipeddi	5	(1.4%)
Shilpasri G Bhat	4	(1.1%)
Nicholas Piggin	4	(1.1%)
Russell Currey	3	(0.8%)
Alistair Popple	2	(0.6%)
Jack Miller	2	(0.6%)
Chris Smart	2	(0.6%)
Matt Brown	1	(0.3%)
Michael Ellerman	1	(0.3%)
Frederic Barrat	1	(0.3%)
Hank Chang	1	(0.3%)
Willie Liauw	1	(0.3%)
Werner Fischer	1	(0.3%)
Jeremy Kerr	1	(0.3%)
Patrick Williams	1	(0.3%)
Joel Stanley	1	(0.3%)
Alexey Kardashevskiy	1	(0.3%)

Developers with the most changed lines

Developer	#	%
Oliver O'Halloran	17961	(56.7%)
Benjamin Herrenschmidt	5509	(17.4%)
Cyril Bur	2801	(8.8%)
Stewart Smith	1649	(5.2%)
Gavin Shan	653	(2.1%)
Claudio Carvalho	489	(1.5%)
Willie Liauw	361	(1.1%)
Ananth N Mavinakayanahalli	340	(1.1%)
Andrew Donnellan	315	(1.0%)
Michael Neuling	240	(0.8%)
Shilpasri G Bhat	228	(0.7%)
Nicholas Piggin	219	(0.7%)
Vasant Hegde	207	(0.7%)
Russell Currey	158	(0.5%)
Jack Miller	127	(0.4%)
Cédric Le Goater	126	(0.4%)
Chris Smart	95	(0.3%)
Hank Chang	56	(0.2%)
Pridhiviraj Paidipeddi	47	(0.1%)
Alistair Popple	39	(0.1%)
Matt Brown	29	(0.1%)
Michael Ellerman	3	(0.0%)
Alexey Kardashevskiy	2	(0.0%)
Frederic Barrat	1	(0.0%)
Werner Fischer	1	(0.0%)
Jeremy Kerr	1	(0.0%)
Patrick Williams	1	(0.0%)
Joel Stanley	1	(0.0%)

Developers with the most lines removed

Developer	#	%
Oliver O'Halloran	8810	(53.4%)
Ananth N Mavinakayanahalli	98	(0.6%)
Alistair Popple	9	(0.1%)
Michael Ellerman	3	(0.0%)
Werner Fischer	1	(0.0%)

Developers with the most signoffs

Total 322

Developer	#	%
Stewart Smith	307	(95.3%)
Michael Neuling	6	(1.9%)
Oliver O'Halloran	3	(0.9%)
Benjamin Herrenschmidt	2	(0.6%)
Vaidyanathan Srinivasan	1	(0.3%)
Hank Chang	1	(0.3%)
Jack Miller	1	(0.3%)
Gavin Shan	1	(0.3%)

Developers with the most reviews

Total: 45

Developer	#	%
Vasant Hegde	10	(22.2%)
Andrew Donnellan	9	(20.0%)
Russell Currey	6	(13.3%)
Cédric Le Goater	5	(11.1%)
Oliver O'Halloran	4	(8.9%)
Gavin Shan	3	(6.7%)
Vaidyanathan Srinivasan	2	(4.4%)
Alistair Popple	2	(4.4%)
Frederic Barrat	2	(4.4%)
Mahesh Salgaonkar	1	(2.2%)
Cyril Bur	1	(2.2%)

Developers with the most test credits

Total 11

Developer	#	%
Willie Liauw	4	(36.4%)
Claudio Carvalho	3	(27.3%)
Gavin Shan	1	(9.1%)
Michael Neuling	1	(9.1%)
Pridhiviraj Paidipeddi	1	(9.1%)
Chris Smart	1	(9.1%)

Developers who gave the most tested-by credits

Total 11

Developer	#	%
Gavin Shan	4	(36.4%)
Stewart Smith	4	(36.4%)
Chris Smart	1	(9.1%)
Oliver O'Halloran	1	(9.1%)
Ananth N Mavinakayanahalli	1	(9.1%)

Developers with the most report credits

Total 7

Developer	#	%
Hank Chang	4	(57.1%)
Guilherme G. Piccoli	1	(14.3%)
Colin Ian King	1	(14.3%)
Pradipta Ghosh	1	(14.3%)

Developers who gave the most report credits

Total 7

Developer	#	%
Gavin Shan	5	(71.4%)
Andrew Donnellan	1	(14.3%)
Jeremy Kerr	1	(14.3%)

5.1.77 skiboot-5.5.0-rc2

skiboot-5.5.0-rc2 was released on Monday April 3rd 2017. It is the second release candidate of skiboot 5.5, which will become the new stable release of skiboot following the 5.4 release, first released November 11th 2016.

skiboot-5.5.0-rc2 contains all bug fixes as of *skiboot-5.4.3* and *skiboot-5.1.19* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.5.0 by April 8th, with skiboot 5.5.0 being for all POWER8 and POWER9 platforms in op-build v1.16 (Due April 12th). This is a short cycle as this release is mainly targetted towards POWER9 bringup efforts.

Following skiboot-5.5.0, we will move to a regular six week release cycle, similar to op-build, but slightly offset to allow for a short stabilisation period. Expected release dates and contents are tracked using GitHub milestone and issues: https://github.com/open-power/skiboot/milestones

Over *skiboot-5.5.0-rc1*, we have the following changes:

NVLINK2

- Introduce NPU2 support

NVLink2 is a new feature introduced on POWER9 systems. It is an evolution of of the NVLink1 feature included in POWER8+ systems but adds several new features including support for GPU address translation using the Nest MMU and cache coherence.

Similar to NVLink1 the functionality is exposed to the OS as a series of virtual PCIe devices. However the actual hardware interfaces are significantly different which limits the amount of common code that can be shared between implementations in the firmware.

This patch adds basic hardware initialisation and exposure of the virtual NVLink2 PCIe devices to the running OS.

- npu2: Add OPAL calls for nvlink2 address translation services (see *OPAL NPU2 calls*)

  Adds three OPAL calls for interacting with NPU2 devices: *OPAL_NPU_INIT_CONTEXT*, *OPAL_NPU_DESTROY_CONTEXT* and *OPAL_NPU_MAP_LPAR*.

  These are used to setup and configure address translation services (ATS) for a process/partition on a given NVLink2 device.

POWER9

- hdata/memory: ignore homer and occ reserved ranges

  We populate these from the HOMER BARs in the PBA directly. There's no need to take the hostboot supplied values so just ignore the corresponding reserved ranges.

- hdata/vpd: Parse the OpenPOWER OPFR record

  Parse the OpenPOWER FRU VPD (OPFR) record on OpenPOWER instead of the VINI records.

- hdata/vpd: Parse additional VINI records

  These records provide hardware version details, CCIN extension information, card type details and hardware characteristics of the FRU

- hdata/cpu: account for p9 shared caches

  On P9 the L2 and L3 caches are shared between pairs of SMT=4 cores. Currently this is not accounted for when creating caches nodes in the device tree. This patch adds additional checking so that a cache node is only created for the first core in the pair and the second core will reference the cache correctly.

- hdata: print backtraces on HDAT errors
- hdat: ignore zero length reserves

  Hostboot can export reserved regions with a length of zero and these should be ignored rather than being turned into reserved range. While we're here fix a memory leak by moving the "too large" region check to before we allocate space for the label.

- SLW: Add init for power9 power management

  This patch adds new function to init core for power9 power management. SPECIAL_WKUP_* SCOM registers, if set, can hold the cores from going into idle states. Hence, clear PPM_SPECIAL_WKUP_HYP_REG scom register for each core during init. (This init are not required for MAMBO)

PCI

- hw/phb3: Adjust ECRC on root port dynamically

  The Samsung NVMe adapter is lost when it's connected to PMC 8546 PCIe switch, until ECRC is disabled on the root port. We found similar issue prevously when Broadcom adapter is connected to same part of PCIe switch and it was fixed by commit 60ce59ccd0e9 ("hw/phb3: Disable ECRC on Broadcom adapter behind PMC switch"). Unfortunately, the commit doesn't fix the Samsung NVMe adapter lost issue.

This fixes the issues by disable ECRC generation/check on root port when PMC 8546 PCIe switch ports are found. This can be extended for other PCIe switches or endpoints in future: Each PHB maintains the count of PCI devices (PMC 8546 PCIe switch ports currently) which require to disable ECRC on root port. The ECRC functionality is enabled when first PMC 8546 switch port is probed and disabled when last PMC 8546 switch port is destroyed (in PCI hot remove scenario). Except PHB's reinitialization after complete reset, the ECRC on root port is untouched.

- core/pci: Fix lost NVMe adapter behind PMC 8546 switch

  The NVMe adapter in below PCI topology is lost. The root cause is the presence bit on its PCI slot is missed, but the PCIe link has been up. The PCI core doesn't probe the adapter behind the slot, leading to lost NVMe adapter in the particular case.

  - PHB3 root port
  - PLX switch 8748 (10b5:8748)
  - PLX swich 9733 (10b5:9733)
  - PMC 8546 swtich (11f8:8546)
  - NVMe adapter (1c58:0023)

  This fixes the issue by overriding the PCI slot presence bit with PCIe link state bit.

- hw/phb4: Locate AER capability position if necessary
- core/pci: Disable surprise hotplug on root port
- core/pci: Ignore PCI slot capability on root port

  We are creating PCI slot on root port, where the PCI slot isn't supported from hardware. For this case, we shouldn't read the PCI slot capability from hardware. When bogus data returned from the hardware, we will attempt to the PCI slot's power state or enable surprise hotplug functionality. All of them can't be accomplished without hardware support.

  This leaves the PCI slot's capability list 0 if PCICAP_EXP_CAP_SLOT isn't set in hardware (pcie_cap + 0x2). Otherwise, the PCI slot's capability list is retrieved from hardware (pcie_cap + 0x14).

- phb4: Default to PCIe GEN2 on DD1

  Default to PCIe GEN2 link speeds on DD1 for stability.

  Can be overridden using nvram pcie-max-link-speed=4 parameter.

- phb3/4: Set max link speed via nvram

  This adds an nvram parameter pcie-max-link-speed to configure the max speed of the pcie link. This can be set from the petitboot prompt using:

  ```
  nvram -p ibm,skiboot --update-config pcie-max-link-speed=4
  ```

  This takes preference over anything set in the device tree and is global to all PHBs.

Tests

- Mambo/Qemu boot tests: expect (and fail) on checkstop

  This allows us to fail a lot faster if we checkstop

5.1.78 skiboot-5.5.0-rc3

skiboot-5.5.0-rc3 was released on Wednesday April 5th 2017. It is the third release candidate of skiboot 5.5, which will become the new stable release of skiboot following the 5.4 release, first released November 11th 2016.

skiboot-5.5.0-rc3 contains all bug fixes as of *skiboot-5.4.3* and *skiboot-5.1.19* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.5.0 by April 8th, with skiboot 5.5.0 being for all POWER8 and POWER9 platforms in op-build v1.16 (Due April 12th). This is a short cycle as this release is mainly targetted towards POWER9 bringup efforts.

Following skiboot-5.5.0, we will move to a regular six week release cycle, similar to op-build, but slightly offset to allow for a short stabilisation period. Expected release dates and contents are tracked using GitHub milestone and issues: https://github.com/open-power/skiboot/milestones

Over *skiboot-5.5.0-rc2*, we have the following changes:

- xive: Fix setting of remote NVT VSD

  This fixes a checkstop when using my XIVE exploitation mode on some multi-chip machines.

- core/init: Use '_' as separator in names of "exports" properties

  The names of the properties under /ibm,opal/firmware/exports are used directly by Linux to create files in sysfs. To remain consistent with the existing naming of OPAL sysfs files, use '_' as the separator.

  In particular for the symbol map which is already exported separately, it's cleaner for the two files to have the same name, eg:

  ```
  /sys/firmware/opal/exports/symbol_map
  /sys/firmware/opal/symbol_map
  ```

- hdata: fix reservation size

  The hostboot reserved ranges are [start, end] pairs rather than [start, end) so we need to stick a +1 in there to calculate the size properly.

- hdat: Add model-name property for OpenPower system
- hdat: Read description from ibm, vpd binary blob
- hdat: Populate model property with 'Unknown' in error path

5.1.79 skiboot-5.6.0

skiboot-5.6.0 was released on Wednesday 24th May 2017. It is the new stable release of skiboot, taking over from the 5.5 release, first released on April 7th 2017. It is the first release done in a regular six week release cycle, mirroring that of op-build.

skiboot-5.6.0 contains all bug fixes as of *skiboot-5.4.4* and *skiboot-5.1.19* (the currently maintained stable releases). We do not currently expect to do any 5.5.x stable releases.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

This release is a good level set of POWER9 support for bringup activities. If you are doing bringup, it is strongly suggested you continue to follow skiboot master.

Changes in skiboot-5.6.0

See changes in the release candidates:

- *skiboot-5.6.0-rc1*
- *skiboot-5.6.0-rc2*

The final 5.6.0 release has no functional changes over the 5.6.0-rc2.

5.1.80 skiboot-5.6.0-rc1

skiboot-5.6.0-rc1 was released on Tuesday May 16th 2017. It is the first release candidate of skiboot 5.6, which will become the new stable release of skiboot following the 5.5 release, first released April 7th 2017.

skiboot-5.6.0-rc1 contains all bug fixes as of *skiboot-5.4.4* and *skiboot-5.1.19* (the currently maintained stable releases). We do not currently expect to do any 5.5.x stable releases.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.6.0 by May 22nd, with skiboot 5.6.0 being for all POWER8 and POWER9 platforms in op-build v1.17 (Due May 24th). This is a short cycle as this release is mainly targetted towards POWER9 bringup efforts.

This is the first release using the new regular six week release cycle, similar to op-build, but slightly offset to allow for a short stabilisation period. Expected release dates and contents are tracked using GitHub milestone and issues: https://github.com/open-power/skiboot/milestones

Over skiboot-5.5, we have the following changes:

New Platforms

Thanks to SuperMicro for submitting support for the p9dsu platform, AKA Boston.

POWER9

XIVE:

- xive: Clear emulation mode queue on reset
- xive: Fixes/improvements to xive reset for multi-chip systems
- xive: Synchronize after disable IRQs in opal_xive_reset()
- xive: Workaround a problem with indirect TM access
- hdata: Make FSPv1 work again One less thing to work around for those crazy enough to try.
- xive: Log more information in opal_xive_dump() for emulation state

  Add a counter of total interrupts taken by a CPU, dump the queue buffer both before and after the current pointer, and also display the HW state of the queue descriptor and the PQ state of the IPI.

- xive: Add a per-cpu logging mechanism to XICS emulation

  This is a small 32-entries rolling buffer that logs a few operations. It's useful to debug odd problems. The output is printed when opal_xive_dump() is called.

- xive: Check queues for duplicates in DEBUG builds.

  There should never be duplicate interrupts in a queue. This adds code to check that when looking at the queue content. Since it can be a performance loss, this is only done for debug builds.

- xive+phb4: Fix exposing trigger page to Linux

HDAT Parsing:

- hdata/spira.c: Add device-tree bindings for nest mmu
- hdata/i2c: Workaround broken i2c devices
- hdata: indicate when booted with elevated risk level

  When the system is IPLed with an elevated risk level Hostboot will set a flag in the IPL parameters structure. Parse and export this in the device tree at: /ipl-params/sys-params/elevated-risk-level

- hdata: Respect OCC and HOMER resevations

  In the past we've ignored these since Hostboot insisted in exporting broken reservations and the OCC was not being used yet. This situation seems to have resolved itself so we should respect the reservations that hostboot provides.

I2C:

- i2c: Add interrupts support on P9

  Some older revisions of hostboot populate the host i2c device fields with all zero entires. Detect and ignore these so we don't crash on boot.

  Without this we get:

```
[  151.251240444,3] DT: dt_attach_root failed, duplicate unknown@0
[  151.251300274,3] ***********************************************
[  151.251339330,3] Unexpected exception 200 !
[  151.251363654,3] SRR0 : 0000000030090c28 SRR1 : 9000000000201000
[  151.251409207,3] HSRR0: 0000000000000010 HSRR1: 9000000000001000
[  151.251444114,3] LR   : 30034018300c5ab0 CTR  : 30034018300a343c
[  151.251478314,3] CFAR : 0000000030024804
[  151.251500346,3] CR   : 40004208  XER: 00000000
    <snip GPRS>
[  151.252083372,0] Aborting!
CPU 0034 Backtrace:
 S: 0000000031cd36a0 R: 000000003001364c   .backtrace+0x2c
 S: 0000000031cd3730 R: 0000000030018db8   ._abort+0x4c
 S: 0000000031cd37b0 R: 0000000030025c6c   .exception_entry+0x114
 S: 0000000031cd3840 R: 0000000000001f00 * +0x1f00
 S: 0000000031cd3a10 R: 0000000031cd3ab0 *
 S: 0000000031cd3aa0 R: 00000000300248b8   .new_property+0x90
 S: 0000000031cd3b30 R: 0000000030024b50   .__dt_add_property_cells+0x30
 S: 0000000031cd3bd0 R: 000000003009abec   .parse_i2c_devs+0x350
 S: 0000000031cd3cf0 R: 0000000030093ffc   .parse_hdat+0x11e4
 S: 0000000031cd3e30 R: 00000000300144c8   .main_cpu_entry+0x138
 S: 0000000031cd3f00 R: 0000000030002648   boot_entry+0x198
```

PHB4:

- phb4: Enforce root complex config space size of 2048

  The root complex config space size on PHB4 is 2048. This patch sets that size and enforces it when trying to read/write the config space in the root complex.

  Without this someone reading the config space via /sysfs in linux will cause an EEH on the PHB.

If too high, reads returns 1s and writes are silently dropped.

- phb4: Add an option for disabling EEH MMIO in nvram

  Having the option to disable EEH for MMIO without rebuilding skiboot could be useful for testing, so check for pci-eeh-mmio=disabled in nvram.

  This is not designed to be a supported option or configuration, just an option that's useful in bringup and development of POWER9 systems.

- phb4: Fix slot presence detect

  This has the nice side effect of improving boot times since we no longer waste time tring to train links that don't have anything present.

- phb4: Enable EEH for MMIO
- phb4: Implement fence check
- phb4: Implement diag data

OCC:

- occ/irq: Fix SCOM address and irq reasons for P9 OCC

  This patch fixes the SCOM address for OCC_MISC register which is used for OCC interupts. In P9, OCC sends an interrupt to notify change in the shared memory like throttle status. This patch handles this interrupt reason.

PRD:

- prd: Fix PRD scoms for P9

NX/DARN:

- nx: Add POWER9 DARN support

NPU2:

- npu2: Do not attempt to initialise non DD1 hardware

  There are significant changes to hardware register addresses and meanings on newer chip revisions making them unlikely to work correctly with the existing code. Better to fail clearly and early.

- npu, npu2: Describe diag data size in device tree

Memory Reservation:

- mem_region: Add reserved regions after memory init

  When a new memory region is added (e.g for memory reserved by firmware) the list of existing memory regions is iterated through and a cut-out is made in any existing region that overlaps with the new one. Prior to the HDAT reservations being made the region init process was always:

  1) Create regions from the memory@<addr> DT nodes. (mostly large)
  2) Create reserved regions from the device-tree. (mostly small)

  When adding new regions we have assumed that the new region will only every intersect with at most one existing region, which it will split. Adding reservations inside the HDAT parser breaks this because when adding the memory@<addr> node regions we can potentially overlap with multiple reserved regions. This patch fixes this by maintaining a seperate list of memory reservations and delaying merging them until after the normal memory init has finished, similar to how DT reservations are handled.

PCI

- pci: Describe PHB diag data size in device tree

  Linux hardcodes the PHB diag data buffer at (as of this commit) 8192 bytes. This has been enough for P7IOC and PHB3, but the 512 PEs of PHB4 pushes the diag data blob over this size. Rather than just increasing the hardcoded size in Linux, provide the size of the diag data blob in the device tree so that the OS can dynamically allocate as much as it needs. This both enables more space for PHB4 and less wasted memory for P7IOC and PHB3.

  P7IOC communicates both hub and PHB data using this buffer, so when setting the size, use whichever struct is largest.

- hdata/i2c: Fix bus and clock frequencies
- ibm-fsp: use opal-prd on p9 and above

  Previously the PRD tooling ran on the FSP, but it was moved into userspace on the host for OpenPower systems. For P9 this system was adopted for FSP systems too.

I2C

- i2c: Remove old hack for bad clock frequency

  This hack dates back to ancient P8 hostboots. The value it would use if it detected the “bad” value was incorrect anyway.

- i2c: Log the engine clock frequency at boot

FSP Systems

These include the Apollo, Firenze and ZZ platforms.

- Remove multiple logging for un-handled fsp sub commands.

  If any new or unknown command need to be handled, just log un-hnadled message from only fsp, not required from fsp-dpo.

  ```
  cat /sys/firmware/opal/msglog | grep -i ,3
  [  110.232114723,3] FSP: fsp_trigger_reset() entry
  [  188.431793837,3] FSP #0: Link down, starting R&R
  [  464.109239162,3] FSP #0: Got XUP with no pending message !
  [  466.340598554,3] FSP-DPO: Unknown command 0xce0900
  [  466.340600126,3] FSP: Unhandled message ce0900
  ```

- FSP: Notify FSP of Platform Log ID after Host Initiated Reset Reload

  Trigging a Host Initiated Reset (when the host detects the FSP has gone out to lunch and should be rebooted), would cause “Unknown Command” messages to appear in the OPAL log.

  This patch implements those messages

  How to trigger FSP RR(HIR):

  ```
  $ putmemproc 300000f8 0x00000000deadbeef
  s1      k0:n0:s0:p00
  ecmd_ppc putmemproc 300000f8 0x00000000deadbeef

  Log showing unknown command:
  ```

(continues on next page)

(continued from previous page)

```
/ # cat /sys/firmware/opal/msglog | grep -i ,3
[  110.232114723,3] FSP: fsp_trigger_reset() entry
[  188.431793837,3] FSP #0: Link down, starting R&R
[  464.109239162,3] FSP #0: Got XUP with no pending message !
[  466.340598554,3] FSP-DPO: Unknown command 0xce0900
[  466.340600126,3] FSP: Unhandled message ce0900
```

The message we need to handle is "Get PLID after host initiated FipS reset/reload". When the FSP comes back from HIR, it asks "hey, so, which error log explains why you rebooted me?". So, we tell it.

Misc

- hdata_to_dt: Misc improvements in the utility and unit test
- GCC7: fixes for -Wimplicit-fallthrough expected regexes

  It turns out GCC7 adds a useful warning and does fancy things like parsing your comments to work out that you intended to do the fallthrough. There's a few places where we don't match the regex. Fix them, as it's harmless to do so.

  Found by building on Fedora Rawhide in Travis.

  While we do not have everything needed to start building successfully with GCC7 (well, at least doing so warning clean), it's a start.
- hdata/i2c: avoid possible int32_t overflow

  We're safe up until engine number 524288. Found by static analysis (of course)
- tpm_i2c_nuvoton: fix use-after-free in tpm_register_chip failure path
- mambo: Fix reserved-ranges node
- external/mambo: add helper for machine checks
- console: Set log level from nvram

  This adds two new nvram options to set the console log level for the driver/uart and in memory. These are called log-level-memory and log-level-driver.

  These are only set once we have nvram inited.

  To set them you do:

  ```
  nvram -p ibm,skiboot --update-config log-level-memory=9
  nvram -p ibm,skiboot --update-config log-level-driver=9
  ```

  You can also use the named versions of emerg, alert, crit, err, warning, notice, printf, info, debug, trace or insane. ie.

  ```
  nvram -p ibm,skiboot --update-config log-level-driver=insane
  ```
- npu: Implement Function Level Reset (FLR)
- mbox: Sanitize interrupts registers
- xive: Fix potential for lost IPIs when manipulating CPPR
- xive: Don't double EOI interrupts that have an EOI override

- libflash/file: Only use 64bit MTD erase ioctl() when needed

  We recently made MTD 64 bit safe in e5720d3fe94 which now requires the 64 bit MTD erase ioctl. Unfortunately this ioctl is not present in older kernels used by some BMC vendors that use pflash.

  This patch addresses this by only using the 64bit version of the erase ioctl() if the parameters exceed 32bit in size.

  If an erase requires the 64bit ioctl() on a kernel which does not support it, the code will still attempt it. There is no way of knowing beforehand if the kernel supports it. The ioctl() will fail and an error will be returned from from the function.

Contributors

This release contains 81 csets from 15 developers, working at 2 employers. A total of 2496 lines added, 641 removed (delta 1855)

Developers with the most changesets

Developer	#	%
Oliver O'Halloran	17	(21.0%)
Benjamin Herrenschmidt	17	(21.0%)
Michael Neuling	16	(19.8%)
Stewart Smith	9	(11.1%)
Russell Currey	8	(9.9%)
Alistair Popple	5	(6.2%)
ppaidipe@linux.vnet.ibm.com	1	(1.2%)
Dave Heller	1	(1.2%)
Jeff Scheel	1	(1.2%)
Nicholas Piggin	1	(1.2%)
Ananth N Mavinakayanahalli	1	(1.2%)
Cyril Bur	1	(1.2%)
Alexey Kardashevskiy	1	(1.2%)
Jim Yuan	1	(1.2%)
Shilpasri G Bhat	1	(1.2%)

Developers with the most changed lines

Developer	#	%
Michael Neuling	748	(28.4%)
Benjamin Herrenschmidt	405	(15.4%)
Russell Currey	360	(13.7%)
Oliver O'Halloran	297	(11.3%)
Nicholas Piggin	187	(7.1%)
Alistair Popple	183	(7.0%)
Stewart Smith	175	(6.6%)
Shilpasri G Bhat	79	(3.0%)
Jim Yuan	56	(2.1%)
Ananth N Mavinakayanahalli	45	(1.7%)
Cyril Bur	38	(1.4%)
Alexey Kardashevskiy	37	(1.4%)
Jeff Scheel	19	(0.7%)
Dave Heller	2	(0.1%)
Pridhiviraj Paidipeddi	1	(0.0%)

Developers with the most lines removed

Developer	#	%
Pridhiviraj Paidipeddi	1	(0.2%)

Developers with the most signoffs

Total of 73.

Developer	#	%
Stewart Smith	56	(76.7%)
Michael Neuling	16	(21.9%)
Oliver O'Halloran	1	(1.4%)

Developers with the most reviews

Total of 6.

Developer	#	%
Oliver O'Halloran	3	(50.0%)
Andrew Donnellan	1	(16.7%)
Gavin Shan	1	(16.7%)
Cyril Bur	1	(16.7%)

Developers with the most test credits

Total of 5.

Developer	#	%
Oliver O'Halloran	2	(40.0%)
Vaidyanathan Srinivasan	1	(20.0%)
Vasant Hegde	1	(20.0%)
Michael Ellerman	1	(20.0%)

Developers who gave the most tested-by credits

Total of 5.

Developer	#	%
Oliver O'Halloran	2	(40.0%)
Benjamin Herrenschmidt	2	(40.0%)
Nicholas Piggin	1	(20.0%)

Developers with the most report credits

Total of 2.

Developer	#	%
Benjamin Herrenschmidt	1	(50.0%)
Pridhiviraj Paidipeddi	1	(50.0%)

Developers who gave the most report credits

Total of 2.

Developer	#	%
Stewart Smith	2	(100.0%)

Top changeset contributors by employer

Total of 2.

Employer	#	%
IBM	80	(98.8%)
SuperMicro	1	(1.2%)

Top lines changed by employer

Employer	#	%
IBM	2576	(97.9%)
SuperMicro	56	(2.1%)

Employers with the most signoffs

Total 73.

Employer	#	%
IBM	73	(100.0%)

Employers with the most hackers

Total 15.

Employer	#	%
IBM	14	(93.3%)
SuperMicro	1	(6.7%)

5.1.81 skiboot-5.6.0-rc2

skiboot-5.6.0-rc2 was released on Friday May 19th 2017. It is the second release candidate of skiboot 5.6, which will become the new stable release of skiboot following the 5.5 release, first released April 7th 2017.

skiboot-5.6.0-rc2 contains all bug fixes as of *skiboot-5.4.4* and *skiboot-5.1.19* (the currently maintained stable releases). We do not currently expect to do any 5.5.x stable releases.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.6.0 by May 22nd, with skiboot 5.6.0 being for all POWER8 and POWER9 platforms in op-build v1.17 (Due May 24th). This is a short cycle as this release is mainly targetted towards POWER9 bringup efforts.

With skiboot 5.6.0, we are moving to a regular six week release cycle, similar to op-build, but slightly offset to allow for a short stabilisation period. Expected release dates and contents are tracked using GitHub milestone and issues: https://github.com/open-power/skiboot/milestones

Over *skiboot-5.6.0-rc1*, we have the following changes:

- hw/i2c: Fix early lock drop

  When interacting with an I2C master the p8-i2c driver (common to p9) aquires a per-master lock which it holds for the duration of it's interaction with the master. Unfortunately, when p8_i2c_check_initial_status() detects that the master is busy with another transaction it drops the lock and returns OPAL_BUSY. This is contrary to the driver's locking strategy which requires that the caller aquire and drop the lock. This leads to a crash due to the double unlock(), which skiboot treats as fatal.

- mambo: Add skiboot/linux symbol lookup

  Adds the skisym and linsym commands which can be used to find the address of a Linux or Skiboot symbol. To function this requires the user to provide the SKIBOOT_MAP and VMLINUX_MAP environmental variables which indicate which skiboot.map and System.map files should be used.

  Examples:

  – Look up a symbol address:

  ```
  systemsim % skisym .load_and_boot_kernel
  0x0000000030013a08
  ```

- Set a breakpoint there:

```
systemsim % b [skisym .load_and_boot_kernel]
breakpoint set at [0:0]: 0x0000000030013a08 (0x0000000030013A08)␣
↪Enc:0x7D800026 : mfcr    r12
```

- libstb: Fix build in OpenSSL 1.1

  The build failure was as follows:

```
[ HOSTCC ] libstb/create-container.c
In file included from /usr/include/openssl/asn1.h:24:0,
                 from /usr/include/openssl/ec.h:30,
                 from libstb/create-container.c:36:
libstb/create-container.c: In function 'getSigRaw':
libstb/create-container.c:104:31: error: dereferencing pointer to incomplete
                                  type 'ECDSA_SIG {aka struct ECDSA_SIG_st}'
  rlen = BN_num_bytes(signature->r);
                               ^
```

5.1.82 skiboot-5.7

skiboot v5.7 was released on Tuesday July 25th 2017. It follows two release candidates of skiboot 5.7, and is now the new stable release of skiboot following the 5.6 release, first released 24th May 2017.

skiboot v5.7 contains all bug fixes as of *skiboot-5.4.6* and *skiboot-5.1.19* (the currently maintained stable releases). We do not currently expect to do any 5.6.x stable releases.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

POWER9 is still in development, and thus all POWER9 users must upgrade to skiboot v5.7.

This is the second release using the new regular six week release cycle, similar to op-build, but slightly offset to allow for a short stabilisation period. Expected release dates and contents are tracked using GitHub milestone and issues: https://github.com/open-power/skiboot/milestones

New Features

Since *skiboot-5.6.0*, we have a few new features:

New features in this release for POWER9 systems:

- In Memory Counters (IMC) (See *OPAL/Skiboot In-Memory Collection (IMC) interface Documentation* for details)
- phb4: Activate shared PCI slot on witherspoon (see *Shared Slot*)
- phb4 capi (i.e. CAPI2): Enable capi mode for PHB4 (see *CAPI on PHB4*)

New feature for IBM FSP based systems:

- fsp/tpo: Provide support for disabling TPO alarm

  This patch adds support for disabling a preconfigured Timed-Power-On(TPO) alarm on FSP based systems. Presently once a TPO alarm is configured from the kernel it will be triggered even if its subsequently disabled.

  With this patch a TPO alarm can be disabled by passing y_m_d==hr_min==0 to fsp_opal_tpo_write(). A branch is added to the function to handle this case by sending FSP_CMD_TPO_DISABLE message to the FSP instead of usual FSP_CMD_TPO_WRITE message. The kernel is expected to call opal_tpo_write() with y_m_d==hr_min==0 to request opal to disable TPO alarm.

POWER9

There are many important changes for POWER9 DD1 and DD2 systems. POWER9 support should be considered in development and skiboot 5.7 is certainly **NOT** suitable for POWER9 production environments.

Since *skiboot-5.7-rc2*:

- platform/witherspoon: Enable eSEL logging

  OpenBMC stack added IPMI OEM extension to log eSEL events. Lets enable eSEL logging from OPAL side.

  See: https://github.com/openbmc/openpower-host-ipmi-oem/blob/d9296050bcece5c2eca5ede0932d944b0ced66c9/oemhandler.cpp#L142 (yes, that is the documentation)

- hdat/i2c: Fix array version check
- mem_region: Check for no-map in reserved nodes

  Regions with the no-map property should be handled seperately to "normal" firmware reservations. When creating mem_region regions from a reserved-memory DT node use the no-map property to select the right reservation type.

- hdata/memory: Add memory reservations to the DT

  Currently we just add these to a list of pre-boot reserved regions which is then converted into a the contents of the /reserved-memory/ node just before Skiboot jumps into the firmware kernel.

  This approach is insufficent because we need to add the ibm,prd-instance labels to the various hostboot reserved regions. To do this we want to create these resevation nodes inside the HDAT parser rather than having the mem_region flattening code handle it. On P8 systems Hostboot placed its memory reservations under the /ibm,hostboot/ node and this patch makes the HDAT parser do the same.

Since Since *skiboot-5.7-rc1*:

- HDAT: Add IPMI sensor data under /bmc node
- numa/associativity: Add a new level of NUMA for GPU's

  Today we have an issue where the NUMA nodes corresponding to GPU's have the same affinity/distance as normal memory nodes. Our reference-points today supports two levels [0x4, 0x4] for normal systems and [0x4, 0x3] for Power8E systems. This patch adds a new level [0x4, X, 0x2] and uses node-id as at all levels for the GPU.

- xive: Enable memory backing of queues

  This dedicates 6x64k pages of memory permanently for the XIVE to use for internal queue overflow. This allows the XIVE to deal with some corner cases where the internal queues might prove insufficient.

- xive: Properly get rid of donated indirect pages during reset

  Otherwise they keep being used accross kexec causing memory corruption in subsequent kernels once KVM has been used.

- cpu: Better handle unknown flags in opal_reinit_cpus()

  At the moment, if we get passed flags we don't know about, we return OPAL_UNSUPPORTED but we still perform whatever actions was requied by the flags we do support. Additionally, on P8, we attempt a SLW re-init which hasn't been supported since Murano DD2.0 and will crash your system.

  It's too late to fix on existing systems so Linux will have to be careful at least on P8, but to avoid future issues let's clean that up, make sure we only use slw_reinit() when HILE isn't supported.

- cpu: Unconditionally cleanup TLBs on P9 in opal_reinit_cpus()

  This can work around problems where Linux fails to properly cleanup part or all of the TLB on kexec.

- Fix scom addresses for power9 nx checkstop hmi handling.

  Scom addresses for NX status, DMA & ENGINE FIR and PBI FIR has changed for Power9. Fixup thoes while handling nx checkstop for Power9.

- Fix scom addresses for power9 core checkstop hmi handling.

  Scom addresses for CORE FIR (Fault Isolation Register) and Malfunction Alert Register has changed for Power9. Fixup those while handling core checkstop for Power9.

  Without this change HMI handler fails to check for correct reason for core checkstop on Power9.

- core/mem_region: check return value of add_region

  The only sensible thing to do if this fails is to abort() as we've likely just failed reserving reserved memory regions, and nothing good comes from that.

Since Since *skiboot-5.6.0*:

- hdata: Reserve Trace Areas

  When hostboot is configured to setup in memory tracing it will reserve some memory for use by the hardware tracing facility. We need to mark these areas as off limits to the operating system and firmware.

- hdata: Make out-of-range idata print at PR_DEBUG

  Some fields just aren't populated on some systems.

- hdata: Ignore unnamed memory reservations.

  Hostboot should name any and all memory reservations that it provides. Currently some hostboots export a broken reservation covering the first 256MB of memory and this causes the system to crash at boot due to an invalid free because this overlaps with the static "ibm,os-reserve" region (which covers the first 768MB of memory).

  According to the hostboot team unnamed reservations are invalid and can be ignored.

- hdata: Check the Host I2C devices array version

  Currently this is not populated on FSP machines which causes some obnoxious errors to appear in the boot log. We also only want to parse version 1 of this structure since future versions will completely change the array item format.

- Ensure P9 DD1 workarounds apply only to Nimbus

  The workarounds for P9 DD1 are only needed for Nimbus. P9 Cumulus will be DD1 but don't need these same workarounds.

  This patch ensures the P9 DD1 workarounds only apply to Nimbus. It also renames some things to make clear what's what.

- cpu: Cleanup AMR and IAMR when re-initializing CPUs

  There's a bug in current Linux kernels leaving crap in those registers accross kexec and not sanitizing them on boot. This breaks kexec under some circumstances (such as booting a hash kernel from a radix one on P9 DD2.0).

  The long term fix is in Linux, but this workaround is a reasonable way of "sanitizing" those SPRs when Linux calls opal_reinit_cpus() and shouldn't have adverse effects.

  We could also use that same mechanism to cleanup other things as well such as restoring some other SPRs to their default value in the future.

- Set POWER9 RPR SPR to 0x00000103070F1F3F. Same value as P8.

  Without this, thread priorities inside a core don't work.

- cpu: Support setting HID[RADIX] and set it by default on P9

  This adds new opal_reinit_cpus() flags to setup radix or hash mode in HID[8] on POWER9.

  By default HID[8] will be set. On P9 DD1.0, Linux will change it as needed. On P9 DD2.0 hash works in radix mode (radix is really "dual" mode) so KVM won't break and existing kernels will work.

  Newer kernels built for hash will call this to clear the HID bit and thus get the full size of the TLB as an optimization.

- Add "cleanup_global_tlb" for P9 and later

  Uses broadcast TLBIE's to cleanup the TLB on all cores and on the nest MMU

- xive: DD2.0 updates

  Add support for StoreEOI, fix StoreEOI MMIO offset in ESB page, and other cleanups

- Update default TSCR value for P9 as recommended by HW folk.
- xive: Fix initialisation of xive_cpu_state struct

  When using XIVE emulation with DEBUG=1, we run into crashes in log_add() due to the xive_cpu_state->log_pos being uninitialised (and thus, with DEBUG enabled, initialised to the poison value of 0x99999999).

PHB4

Since *skiboot-5.7-rc2*:

- phb4: Add link training trace mode

  Add a mode to PHB4 to trace training process closely. This activates as soon as PERST is deasserted and produces human readable output of the process.

  This may increase training times since it duplicates some of the training code. This code has it's own simple checks for fence and timeout but will fall through to the default training code once done.

  Output produced, looks like the "TRACE:" lines below:

```
[    3.410799664,7] PHB#0001[0:1]: FRESET: Starts
[    3.410802000,7] PHB#0001[0:1]: FRESET: Prepare for link down
[    3.410806624,7] PHB#0001[0:1]: FRESET: Assert skipped
[    3.410808848,7] PHB#0001[0:1]: FRESET: Deassert
[    3.410812176,3] PHB#0001[0:1]: TRACE: 0x0000000101000000  0ms
[    3.417170176,3] PHB#0001[0:1]: TRACE: 0x0000100101000000 12ms presence
[    3.436289104,3] PHB#0001[0:1]: TRACE: 0x0000180101000000 49ms training
[    3.436373312,3] PHB#0001[0:1]: TRACE: 0x00001d0811000000 49ms trained
[    3.436420752,3] PHB#0001[0:1]: TRACE: Link trained.
[    3.436967856,7] PHB#0001[0:1]: LINK: Start polling
[    3.437482240,7] PHB#0001[0:1]: LINK: Electrical link detected
[    3.437996864,7] PHB#0001[0:1]: LINK: Link is up
[    4.438000048,7] PHB#0001[0:1]: LINK: Link is stable
```

  Enabled via nvram using:

```
nvram -p ibm,skiboot --update-config pci-tracing=true
```

- phb4: Improve reset and link training timing

  This improves PHB reset and link training timing.

- phb4: Add phb4_check_reg() to sanity check failures

  This adds a function phb4_check_reg() to sanity check when we do MMIO reads from the PHB to make sure it's not fenced.

- phb4: Remove retry on electrical link timeout

  Currently we retry if we don't detect an electrical link. This is pointless as all devices should respond in the given time.

  This patches removes this retry and just returns OPAL_HARDWARE if we don't detect an electrical link.

  This has the additional benefit of improving boot times on machines that have badly wired presence detect (ie. says a device is present when there isn't).

- phb4: Read PERST signal rather than assuming it's asserted

  Currently we assume on boot that PERST is asserted so that we can skip having to assert it ourselves.

  This instead reads the PERST status and determines if we need to assert it based on that.

- phb4: Fix endian of TLP headers print

  Byte swap TLP headers so they are the same as the PCIe spec.

- phb4: Change timeouts prints to error level

  If the link doesn't have a electrical link or the link doesn't train we should make that more obvious to the user.

- phb4: Better logs why the slot didn't work

  Better logs why the slot didn't work and make it a PR_ERR so users see it by default.

- phb4: Force verbose EEH logging

  Force verbose EEH. This is a heavy handed and we should turn if off later as things stabilise, but is useful for now.

- phb4: Initialization sequence updates

  Mostly errata workarounds, some DD1 specific.

  The step Init_5 was moved to Init_16, so the numbering was updated to reflect this.

Since *skiboot-5.7-rc1*:

- phb4: Do more retries on link training failures Currently we only retry once when we have a link training failure. This changes this to be 3 retries as 1 retry is not giving us enough reliablity.

  This will increase the boot time, especially on systems where we incorrectly detect a link presence when there really is nothing present. I'll post a followup patch to optimise our timings to help mitigate this later.

- phb4: Workaround phy lockup by doing full PHB reset on retry

  For PHB4 it's possible that the phy may end up in a bad state where it can no longer recieve data. This can manifest as the link not retraining. A simple PERST will not clear this. The PHB must be completely reset.

  This changes the retry state to CRESET to do this.

  This issue may also manifest itself as the link training in a degraded state (lower speed or narrower width). This patch doesn't attempt to fix that (will come later).

- pci: Add ability to trace timing

  PCI link training is responsible for a huge chunk of the skiboot boot time, so add the ability to trace it waiting in the main state machine.

- pci: Print resetting PHB notice at higher log level

  Currently during boot there a long delay while we wait for the PHBs to be reset and train. During this time, there is no output from skiboot and the last message doesn't give an indication of what's happening.

  This boosts the PHB reset message from info to notice so users can see what's happening during this long period of waiting.

- phb4: Only set one bit in nfir

  The MPIPL procedure says to only set bit 26 when forcing the PEC into freeze mode. Currently we set bits 24-27.

  This changes the code to follow spec and only set bit 26.

- phb4: Fix order of pfir/nfir clearing in CRESET

  According to the workbook, pfir must be cleared before the nfir. The way we have it now causes the nfir to not clear properly in some error circumstances.

  This swaps the order to match the workbook.

- phb4: Remove incorrect state transition

  When waiting in PHB4_SLOT_CRESET_WAIT_CQ for transations to end, we incorrectly move onto the next state. Generally we don't hit this as the transactions have ended already anyway.

  This removes the incorrect state transition.

- phb4: Set default lane equalisation

  Set default lane equalisation if there is nothing in the device-tree.

  Default value taken from hdat and confirmed by hardware team. Neatens the code up a bit too.

- hdata: Fix phb4 lane-eq property generation

  The lane-eq data we get from hdat is all 7s but what we end up in the device tree is:

```
xscom@603fc00000000/pbcq@4010c00/stack@0/ibm,lane-eq
                 00000000 31c339e0 00000000 0000000c
                 00000000 00000000 00000000 00000000
                 00000000 31c30000 77777777 77777777
                 77777777 77777777 77777777 77777777
```

  This fixes grabbing the properties from hdat and fixes the call to put them in the device tree.

- phb4: Fix PHB4 fence recovery.

  We had a few problems:

  - We used the wrong register to trigger the reset (spec bug)
  - We should clear the PFIR and NFIR while the reset is asserted
  - … and in the right order !
  - We should only apply the DD1 workaround after the reset has been lifted.
  - We should ensure we use ASB whenever we are fenced or doing a CRESET
  - Make config ops write with ASB

- phb4: Verbose EEH options

  Enabled via nvram pci-eeh-verbose=true. ie.

```
nvram -p ibm,skiboot --update-config pci-eeh-verbose=true
```

- phb4: Print more info when PHB fences

  For now at PHBERR level. We don't have room in the diags data passed to Linux for these unfortunately.

Since *skiboot-5.6.0*:

- phb4: Fix number of index bits in IODA tables

  On PHB4 the number of index bits in the IODA table address register was bumped to 10 bits to accomodate for 1024 MSIs and 1024 TVEs (DD2).

  However our macro only defined the field to be 9 bits, thus causing "interesting" behaviours on some systems.

- phb4: Harden init with bad PHBs

  Currently if we read all 1's from the EEH or IRQ capabilities, we end up train wrecking on some other random code (eg. an assert() in xive).

  This hardens the PHB4 code to look for these bad reads and more gracefully fails the init for that PHB alone. This allows the rest of the system to boot and ignore those bad PHBs.

- phb4 capi (i.e. CAPI2): Handle HMI events

  Find the CAPP on the chip associated with the HMI event for PHB4. The recovery mode (re-initialization of the capp, resume of functional operations) is only available with P9 DD2. A new patch will be provided to support this feature.

- phb4 capi (i.e. CAPI2): Enable capi mode for PHB4

  Enable the Coherently attached processor interface. The PHB is used as a CAPI interface. CAPI Adapters can be connected to either PEC0 or PEC2. Single port CAPI adapter can be connected to either PEC0 or PEC2, but Dual-Port Adapter can be only connected to PEC2 * CAPP0 attached to PHB0(PEC0 - single port) * CAPP1 attached to PHB3(PEC2 - single or dual port)

- hw/phb4: Rework phb4_get_presence_state()

  There are two issues in current implementation: It should return errcode visibile to Linux, which has prefix OPAL_*. The code isn't very obvious.

  This returns OPAL_HARDWARE when the PHB is broken. Otherwise, OPAL_SUCCESS is always returned. In the mean while, It refactors the code to make it obvious: OPAL_PCI_SLOT_PRESENT is returned when the presence signal (low active) or PCIe link is active. Otherwise, OPAL_PCI_SLOT_EMPTY is returned.

- phb4: Error injection for config space

  Implement CFG (config space) error injection.

  This works the same as PHB3. MMIO and DMA error injection require a rewrite, so they're unsupported for now.

  While it's not feature complete, this at least provides an easy way to inject an error that will trigger EEH.

- phb4: Error clear implementation
- phb4: Mask link down errors during reset

  During a hot reset the PCI link will drop, so we need to mask link down events to prevent unnecessary errors.

- phb4: Implement root port initialization

  phb4_root_port_init() was a NOP before, so fix that.

- phb4: Complete reset implementation

  This implements complete reset (creset) functionality for POWER9 DD1.

  Only partially tested and contends with some DD1 errata, but it's a start.

- phb4: Activate shared PCI slot on witherspoon

  Witherspoon systems come with a 'shared' PCI slot: physically, it looks like a x16 slot, but it's actually two x8 slots connected to two PHBs of two different chips. Taking advantage of it requires some logic on the PCI adapter. Only the Mellanox CX5 adapter is known to support it at the time of this writing.

  This patch enables support for the shared slot on witherspoon if a x16 adapter is detected. Each x8 slot has a presence bit, so both bits need to be set for the activation to take place. Slot sharing is activated through a gpio.

  Note that there's no easy way to be sure that the card is indeed a shared-slot compatible PCI adapter and not a normal x16 card. Plugging a normal x16 adapter on the shared slot should be avoided on witherspoon, as the link won't train on the second slot, resulting in a timeout and a longer boot time. Only the first slot is usable and the x16 adapter will end up using only half the lines.

  If the PCI card plugged on the physical slot is only x8 (or less), then the presence bit of the second slot is not set, so this patch does nothing. The x8 (or less) adapter should work like on any other physical slot.

- phb4: Block D-state power management on direct slots

  As current revisions of PHB4 don't properly handle the resulting L1 link transition.

- phb4: Call pci config filters
- phb4: Mask out write-1-to-clear registers in RC cfg

  The root complex config space only supports 4-byte accesses. Thus, when the client requests a smaller size write, we do a read-modify-write to the register.

  However, some register have bits defined as "write 1 to clear".

  If we do a RMW cycles on such a register and such bits are 1 in the part that the client doesn't intend to modify, we will accidentally write back those 1's and clear the corresponding bit.

  This avoids it by masking out those magic bits from the "old" value read from the register.

- phb4: Properly mask out link down errors during reset
- phb3/4: Silence a useless warning

  PHB's don't have base location codes on non-FSP systems and it's normal.

- phb4: Workaround bug in spec 053

  Wait for DLP PGRESET to clear *after* lifting the PCIe core reset

- phb4: DD2.0 updates

  Support StoreEOI, full complements of PEs (twice as big TVT) and other updates.

  Also renumber init steps to match spec 063

NPU2

Note that currently NPU2 support is limited to POWER9 DD1 hardware.

Since *skiboot-5.6.0*:

- platforms/astbmc/witherspoon.c: Add NPU2 slot mappings

  For NVLink2 to function PCIe devices need to be associated with the right NVLinks. This association is supposed to be passed down to Skiboot via HDAT but those fields are still not correctly filled out. To work around this we add slot tables for the NVLinks similar to what we have for P8+.

- hw/npu2.c: Fix device aperture calculation

  The POWER9 NPU2 implements an address compression scheme to compress 56-bit P9 physical addresses to 47-bit GPU addresses. System software needs to know both addresses, unfortunately the calculation of the compressed address was incorrect. Fix it here.

- hw/npu2.c: Change MCD BAR allocation order

  MCD BARs need to be correctly aligned to the size of the region. As GPU memory is allocated from the top of memory down we should start allocating from the highest GPU memory address to the lowest to ensure correct alignment.

- NPU2: Add flag to nvlink config space indicating DL reset state

  Device drivers need to be able to determine if the DL is out of reset or not so they can safely probe to see if links have already been trained. This patch adds a flag to the vendor specific config space indicating if the DL is out of reset.

- hw/npu2.c: Hardcode MSR_SF when setting up npu XTS contexts

  We don't support anything other than 64-bit mode for address translations so we can safely hardcode it.

- hw/npu2-hw-procedures.c: Add nvram option to override zcal calculations

  In some rare cases the zcal state machine may fail and flag an error. According to hardware designers it is sometimes ok to ignore this failure and use nominal values for the calculations. In this case we add a nvram variable (nv_zcal_override) which will cause skiboot to ignore the failure and use the nominal value specified in nvram.

- npu2: Fix npu2_{read,write}_4b()

  When writing or reading 4-byte values, we need to use the upper half of the 64-bit SCOM register.

  Fix npu2_{read,write}_4b() and their callers to use uint32_t, and appropriately shift the value being written or returned.

- hw/npu2.c: Fix opal_npu_map_lpar to search for existing BDF
- hw/npu2-hw-procedures.c: Fix running of zcal procedure

  > The zcal procedure should only be run once per obus (ie. once per group of 3 links). Clean up the code and fix the potential buffer overflow due to a typo. Also updates the zcal settings to their proper values.

- hw/npu2.c: Add memory coherence directory programming

  The memory coherence directory (MCD) needs to know which system memory addresses belong to the GPU. This amounts to setting a BAR and a size in the MCD to cover the addresses assigned to each of the GPUs. To ease assignment we assume GPUs are assigned memory in a contiguous block per chip.

OCC/Power Management

With this release, it's possible to boot POWER9 systems with the OCC enabled and change CPU frequencies. Doing so does require other firmware components to also support this (otherwise the frequency will not be set).

Since *skiboot-5.6.0*:

- occ: Skip setting cores to nominal frequency in P9

  In P9, once OCC is up, it is supposed to setup the cores to nominal frequency. So skip this step in OPAL.

- occ: Fix Pstate ordering for P9

  In P9 the pstate values are positive. They are continuous set of unsigned integers [0 to +N] where Pmax is 0 and Pmin is N. The linear ordering of pstates for P9 has changed compared to P8. P8 has neagtive pstate values advertised as [0 to -N] where Pmax is 0 and Pmin is -N. This patch adds helper routines to abstract pstate comparison with pmax and adds sanity pstate limit checks. This patch also fixes pstate arithmetic by using labs().

- p8-i2c: occ: Add support for OCC to use I2C engines

  This patch adds support to share the I2C engines with host and OCC. OCC uses I2C engines to read DIMM temperatures and to communicate with GPU. OCC Flag register is used for locking between host and OCC. Host requests for the bus by setting a bit in OCC Flag register. OCC sends an interrupt to indicate the change in ownership.

opal-prd/PRD

Since *skiboot-5.6.0*:

- opal-prd: Handle SBE passthrough message passing

  This patch adds support to send SBE pass through command to HBRT.

- SBE: Add passthrough command support

  SBE sends passthrough command. We have to capture this interrupt and send event to HBRT via opal-prd (user space daemon).

- opal-prd: hook up reset_pm_complex

  This change provides the facility to invoke HBRT's reset_pm_complex, in the same manner is done with process_occ_reset previously.

  We add a control command for *opal-prd pm-complex reset*, which is just an alias for occ_reset at this stage.

- prd: Implement firmware side of opaque PRD channel

  This change introduces the firmware side of the opaque HBRT <–> OPAL message channel. We define a base message format to be shared with HBRT (in include/prd-fw-msg.h), and allow firmware requests and responses to be sent over this channel.

  We don't currently have any notifications defined, so have nothing to do for firmware_notify() at this stage.

- opal-prd: Add firmware_request & firmware_notify implementations

  This change adds the implementation of firmware_request() and firmware_notify(). To do this, we need to add a message queue, so that we can properly handle out-of-order messages coming from firmware.

- opal-prd: Add support for variable-sized messages

  With the introductuion of the opaque firmware channel, we want to support variable-sized messages. Rather than expecting to read an entire 'struct opal_prd_msg' in one read() call, we can split this over mutiple reads, potentially expanding our message buffer.

- opal-prd: Sync hostboot interfaces with HBRT

  This change adds new callbacks defined for p9, and the base thunks for the added calls.

- opal-prd: interpret log level prefixes from HBRT

  Interpret the (optional) *_MRK log prefixes on HBRT messages, and set the syslog log priority to suit.

- opal-prd: Add occ reset to usage text
- opal-prd: allow different chips for occ control actions

  The *occ reset* and *occ error* actions can both take a chip id argument, but we're currently just using zero. This change changes the control message format to pass the chip ID from the control process to the opal-prd daemon.

IBM FSP based platforms

Since *skiboot-5.7-rc2*:

- FSP/CONSOLE: Do not enable input irq in write path

  We use irq for reading input from console, but not in output path. Hence do not enable input irq in write path.

  Fixes : 583c8203 (fsp/console: Allocate irq for each hvc console)

Since *skiboot-5.6.0*:

- FSP/CONSOLE: Fix possible NULL dereference
- platforms/ibm-fsp/firenze: Fix PCI slot power-off pattern

  When powering off the PCI slot, the corresponding bits should be set to 0bxx00xx00 instead of 0bxx11xx11. Otherwise, the specified PCI slot can't be put into power-off state. Fortunately, it didn't introduce any side-effects so far.

- FSP/CONSOLE: Workaround for unresponsive ipmi daemon

  We use TCE mapped area to write data to console. Console header (fsp_serbuf_hdr) is modified by both FSP and OPAL (OPAL updates next_in pointer in fsp_serbuf_hdr and FSP updates next_out pointer).

  Kernel makes opal_console_write() OPAL call to write data to console. OPAL write data to TCE mapped area and sends MBOX command to FSP. If our console becomes full and we have data to write to console, we keep on waiting until FSP reads data.

  In some corner cases, where FSP is active but not responding to console MBOX message (due to buggy IPMI) and we have heavy console write happening from kernel, then eventually our console buffer becomes full. At this point OPAL starts sending OPAL_BUSY_EVENT to kernel. Kernel will keep on retrying. This is creating kernel soft lockups. In some extreme case when every CPU is trying to write to console, user will not be able to ssh and thinks system is hang.

  If we reset FSP or restart IPMI daemon on FSP, system recovers and everything becomes normal.

  This patch adds workaround to above issue by returning OPAL_HARDWARE when cosole is full. Side effect of this patch is, we may endup dropping latest console data. But better to drop console data than system hang.

- FSP: Set status field in response message for timed out message

  For timed out FSP messages, we set message status as "fsp_msg_timeout". But most FSP driver users (like surviellance) are ignoring this field. They always look for FSP returned status value in callback function (second byte in word1). So we endup treating timed out message as success response from FSP.

  Sample output:

  ```
  [69902.432509048,7] SURV: Sending the heartbeat command to FSP
  [70023.226860117,4] FSP: Response from FSP timed out, word0 = d66a00d7, word1 = 0␣
  ↪state: 3
  ....
  ```

(continues on next page)

(continued from previous page)

```
[70023.226901445,7] SURV: Received heartbeat acknowledge from FSP
[70023.226903251,3] FSP: fsp_trigger_reset() entry
```

Here SURV code thought it got valid response from FSP. But actually we didn't receive response from FSP.

This patch fixes above issue by updating status field in response structure.

- FSP: Improve timeout message
- FSP/RTC: Fix possible FSP R/R issue in rtc write path
- hw/fsp/rtc: read/write cached rtc tod on fsp hir.

  Currently fsp-rtc reads/writes the cached RTC TOD on an fsp reset. Use latest fsp_in_rr() function to properly read the cached rtc value when fsp reset initiated by the hir.

  Below is the kernel trace when we set hw clock, when hir process starts.

```
[ 1727.775824] NMI watchdog: BUG: soft lockup - CPU#57 stuck for 23s!␣
↪[hwclock:7688]
[ 1727.775856] Modules linked in: vmx_crypto ibmpowernv ipmi_powernv uio_pdrv_
↪genirq ipmi_devintf powernv_op_panel uio ipmi_msghandler powernv_rng leds_
↪powernv ip_tables x_tables autofs4 ses enclosure scsi_transport_sas crc32c_
↪vpmsum lpfc ipr tg3 scsi_transport_fc
[ 1727.775883] CPU: 57 PID: 7688 Comm: hwclock Not tainted 4.10.0-14-generic #16-
↪Ubuntu
[ 1727.775883] task: c000000fdfdc8400 task.stack: c000000fdfef4000
[ 1727.775884] NIP: c00000000090540c LR: c0000000000846f4 CTR: 000000003006dd70
[ 1727.775885] REGS: c000000fdfef79a0 TRAP: 0901   Not tainted  (4.10.0-14-
↪generic)
[ 1727.775886] MSR: 9000000000009033 <SF,HV,EE,ME,IR,DR,RI,LE>
[ 1727.775889]   CR: 28024442  XER: 20000000
[ 1727.775890] CFAR: c00000000008472c SOFTE: 1
               GPR00: 0000000030005128 c000000fdfef7c20 c00000000144c900␣
↪fffffffffffffff4
               GPR04: 0000000028024442 c00000000090540c 9000000000009033␣
↪0000000000000000
               GPR08: 0000000000000000 0000000031fc4000 c000000000084710␣
↪9000000000001003
               GPR12: c0000000000846e8 c00000000fba0100
[ 1727.775897] NIP [c00000000090540c] opal_set_rtc_time+0x4c/0xb0
[ 1727.775899] LR [c0000000000846f4] opal_return+0xc/0x48
[ 1727.775899] Call Trace:
[ 1727.775900] [c000000fdfef7c20] [c00000000090540c] opal_set_rtc_time+0x4c/0xb0␣
↪(unreliable)
[ 1727.775901] [c000000fdfef7c60] [c000000000900828] rtc_set_time+0xb8/0x1b0
[ 1727.775903] [c000000fdfef7ca0] [c000000000902364] rtc_dev_ioctl+0x454/0x630
[ 1727.775904] [c000000fdfef7d40] [c00000000035b1f4] do_vfs_ioctl+0xd4/0x8c0
[ 1727.775906] [c000000fdfef7de0] [c00000000035bab4] SyS_ioctl+0xd4/0xf0
[ 1727.775907] [c000000fdfef7e30] [c00000000000b184] system_call+0x38/0xe0
[ 1727.775908] Instruction dump:
[ 1727.775909] f821ffc1 39200000 7c832378 91210028 38a10020 39200000 38810028␣
↪f9210020
[ 1727.775911] 4bfffe6d e8810020 80610028 4b77f61d <60000000> 7c7f1b78 3860000a␣
↪2fbffff4
```

  This is found when executing the testcase https://github.com/open-power/op-test-framework/blob/master/testcases/fspresetReload.py

  With this fix ran fsp hir torture testcase in the above test which is working fine.

- occ: Set return variable to correct value

  When entering this section of code rc will be zero. If fsp_mkmsg() fails the code responsible for printing an error message won't be set. Resetting rc should allow for the error case to trigger if fsp_mkmsg fails.

- capp: Fix hang when CAPP microcode LID is missing on FSP machine

  When the LID is absent, we fail early with an error from start_preload_resource. In that case, capp_ucode_info.load_result isn't set properly causing a subsequent capp_lid_download() to call wait_for_resource_loaded() on something that isn't being loaded, thus hanging.

- FSP: Add check to detect FSP R/R inside fsp_sync_msg()

  OPAL sends MBOX message to FSP and updates message state from fsp_msg_queued -> fsp_msg_sent. fsp_sync_msg() queues message and waits until we get response from FSP. During FSP R/R we move outstanding MBOX messages from msgq to rr_queue including inflight message (fsp_reset_cmdclass()). But we are not resetting inflight message state.

  In extreme croner case where we sent message to FSP via fsp_sync_msg() path and FSP R/R happens before getting respose from FSP, then we will endup waiting in fsp_sync_msg() until everything becomes normal.

  This patch adds fsp_in_rr() check to fsp_sync_msg() and return error to caller if FSP is in R/R.

- FSP: Add check to detect FSP R/R inside fsp_sync_msg()

  OPAL sends MBOX message to FSP and updates message state from fsp_msg_queued -> fsp_msg_sent. fsp_sync_msg() queues message and waits until we get response from FSP. During FSP R/R we move outstanding MBOX messages from msgq to rr_queue including inflight message (fsp_reset_cmdclass()). But we are not resetting inflight message state.

  In extreme croner case where we sent message to FSP via fsp_sync_msg() path and FSP R/R happens before getting respose from FSP, then we will endup waiting in fsp_sync_msg() until everything becomes normal.

  This patch adds fsp_in_rr() check to fsp_sync_msg() and return error to caller if FSP is in R/R.

- capp: Fix hang when CAPP microcode LID is missing on FSP machine

  When the LID is absent, we fail early with an error from start_preload_resource. In that case, capp_ucode_info.load_result isn't set properly causing a subsequent capp_lid_download() to call wait_for_resource_loaded() on something that isn't being loaded, thus hanging.

- FSP/CONSOLE: Do not free fsp_msg in error path

  as we reuse same msg to send next output message.

- platform/zz: Acknowledge OCC_LOAD mbox message in ZZ

  In P9 FSP box, OCC image is pre-loaded. So do not handle the load command and send SUCCESS to FSP on recieving OCC_LOAD mbox message.

- FSP/RTC: Improve error log

astbmc systems

Since *skiboot-5.6.0*:

- platforms/astbmc: Don't validate model on palmetto

  The platform isn't compatible with palmetto until the root device-tree node's "model" property is NULL or "palmetto". However, we could have "TN71-BP012" for the property on palmetto.

  ```
  linux# cat /proc/device-tree/model
  TN71-BP012
  ```

This skips the validation on root device-tree node's "model" property on palmetto, meaning we check the "compatible" property only.

General

Since *skiboot-5.7-rc2*:

- core/pci: Fix mem-leak on fast-reboot

  Fast-reboot has a memory leak which causes the system to crash after about 250 fast-reboots. The patch fixes the memory leak. The cause of the leak was the pci_device's being freed, without freeing the pci_slot within it.

- gcov: properly handle gard and pflash code coverage

Since *skiboot-5.6.0*:

- Reduce log level on non-error log messages

  90% of what we print isn't useful to a normal user. This dramatically reduces the amount of messages printed by OPAL in normal circumstances.

- init: Silence messages and call ourselves "OPAL"
- psi: Switch to ESB mode later

  There's an errata, if we switch to ESB mode before setting up the various ESB mode related registers, a pending interrupts can go wrong.

- lpc: Enable "new" SerIRQ mode
- hw/ipmi/ipmi-sel: missing newline in prlog warning
- p8-i2c OCC lock: fix locking in p9_i2c_bus_owner_change
- Convert important polling loops to spin at lowest SMT priority

  The pattern of calling cpu_relax() inside a polling loop does not suit the powerpc SMT priority instructions. Prefrred is to set a low priority then spin until break condition is reached, then restore priority.

- Improve cpu_idle when PM is disabled

  Split cpu_idle() into cpu_idle_delay() and cpu_idle_job() rather than requesting the idle type as a function argument. Have those functions provide a default polling (non-PM) implentation which spin at the lowest SMT priority.

- core/fdt: Always add a reserve map

  Currently we skip adding the reserved ranges block to the generated FDT blob if we are excluding the root node. This can result in a DTB that dtc will barf on because the reserved memory ranges overlap with the start of the dt_struct block. As an example:

  ```
  $ fdtdump broken.dtb -d
  /dts-v1/;
  // magic:               0xd00dfeed
  // totalsize:           0x7f3 (2035)
  // off_dt_struct:       0x30  <----\
  // off_dt_strings:      0x7b8       | this is bad!
  // off_mem_rsvmap:      0x30  <----/
  // version:             17
  // last_comp_version:   16
  // boot_cpuid_phys:     0x0
  // size_dt_strings:     0x3b
  // size_dt_struct:      0x788
  ```

(continues on next page)

(continued from previous page)

```
/memreserve/ 0x100000000 0x300000004;
/memreserve/ 0x3300000001 0x169626d2c;
/memreserve/ 0x706369652d736c6f 0x7473000000000003;
        *continues*
```

With this patch:

```
$ fdtdump working.dtb -d
/dts-v1/;
// magic:               0xd00dfeed
// totalsize:           0x803 (2051)
// off_dt_struct:       0x40
// off_dt_strings:      0x7c8
// off_mem_rsvmap:      0x30
// version:             17
// last_comp_version:   16
// boot_cpuid_phys:     0x0
// size_dt_strings:     0x3b
// size_dt_struct:      0x788

// 0040: tag: 0x00000001 (FDT_BEGIN_NODE)
/ {
// 0048: tag: 0x00000003 (FDT_PROP)
// 07fb: string: phandle
// 0054: value
    phandle = <0x00000001>;
        *continues*
```

- hw/lpc-mbox: Use message registers for interrupts

  Currently the BMC raises the interrupt using the BMC control register. It does so on all accesses to the 16 'data' registers meaning that when the BMC only wants to set the ATTN (on which we have interrupts enabled) bit we will also get a control register based interrupt.

  The solution here is to mask that interrupt permanantly and enable interrupts on the protocol defined 'response' data byte.

PCI

Since *skiboot-5.6.0*:

- pci: Wait 20ms before checking presence detect on PCIe

  As the PHB presence logic has a debounce timer that can take a while to settle.

- phb3+iov: Fixup support for config space filters

  The filter should be called before the HW access and its return value control whether to perform the access or not

- core/pci: Use PCI slot's power facality in pci_enable_bridge()

  The current implmentation has incorrect assumptions: there is always a PCI slot associated with root port and PCIe switch downstream port and all of them are capable to change its power state by register PCICAP_EXP_SLOTCTL. Firstly, there might not a PCI slot associated with the root port or PCIe switch downstream port. Secondly, the power isn't controlled by standard config register (PCICAP_EXP_SLOTCTL). There are I2C slave devices used to control the power states on Tuleta.

In order to use the PCI slot's methods to manage the power states, this does:

  - Introduce PCI_SLOT_FLAG_ENFORCE, indicates the request operation is enforced to be applied.
  - pci_enable_bridge() is split into 3 functions: pci_bridge_power_on() to power it on; pci_enable_bridge() as a place holder and pci_bridge_wait_link() to wait the downstream link to come up.
  - In pci_bridge_power_on(), the PCI slot's specific power management methods are used if there is a PCI slot associated with the PCIe switch downstream port or root port.

- platforms/astbmc/slots.c: Allow comparison of bus numbers when matching slots

  When matching devices on multiple down stream PLX busses we need to compare more than just the device-id of the PCIe BDFN, so increase the mask to do so.

Debugging, Tests and simulators

Since *skiboot-5.7-rc2*:

- boot_tests: add PFLASH_TO_COPY for OpenBMC
- travis: Add debian stretch and unstable

  At the moment, we mark them both as being able to fail, as we're hitting an assert in one of the unit tests on debian stretch, and that hasn't yet been chased down.

- core/backtrace: Serialise printing backtraces

  Add a lock so that only one thread can print a backtrace at a time. This should prevent multiple threads from garbaling each other's backtraces.

Since *skiboot-5.7-rc1*:

- lpc: remove double LPC prefix from messages
- opal-ci/fetch-debian-jessie-installer: follow redirects Fixes some CI failures
- test/qemu-jessie: bail out fast on kernel panic
- test/qemu-jessie: dump boot log on failure
- travis: add fedora26
- xz: add fallthrough annotations to silence GCC7 warning

Since *skiboot-5.6.0*:

- boot-tests: add OpenBMC support
- boot_test.sh: Add SMC BMC support

  Your BMC needs a special debug image flashed to use this, the exact image and methods aren't something I can publish here, but if you work for IBM or SMC you can find out from the right sources.

  A few things are needed to move around to be able to flash to a SMC BMC.

  For a start, the SSH daemon will only accept connections after a special incantation (which I also can't share), but you should put that in the ~/.skiboot_boot_tests file along with some other default login information we don't publicise too broadly (because Security Through Obscurity is *obviously* a good idea. . . .)

  We also can't just directly "ssh /bin/true", we need an expect script, and we can't scp, but we can anonymous rsync!

  You also need a pflash binary to copy over.

- hdata_to_dt: Add PVR overrides to the usage text

- mambo: Add a reservation for the initramfs

  On most systems the initramfs is loaded inside the part of memory reserved for the OS [0x0-0x30000000] and skiboot will never touch it. On mambo it's loaded at 0x80000000 and if you're unlucky skiboot can allocate over the top of it and corrupt the initramfs blob.

  There might be the downside that the kernel cannot re-use the initramfs memory since it's marked as reserved, but the kernel might also free it anyway.

- mambo: Update P9 PVR to reflect Scale out 24 core chips

  The P9 PVR bits 48:51 don't indicate a revision but instead different configurations. From BookIV we have:

Bits	Configuration
0	Scale out 12 cores
1	Scale out 24 cores
2	Scale up 12 cores
3	Scale up 24 cores

  Skiboot will mostly the use "Scale out 24 core" configuration (ie. SMT4 not SMT8) so reflect this in mambo.

- core: Move enable_mambo_console() into chip initialisation

  Rather than having a wart in main_cpu_entry() that initialises the mambo console, we can move it into init_chips() which is where we discover that we're on mambo.

- mambo: Create multiple chips when we have multiple CPUs

  Currently when we boot mambo with multiple CPUs, we create multiple CPU nodes in the device tree, and each claims to be on a separate chip.

  However we don't create multiple xscom nodes, which means skiboot only knows about a single chip, and all CPUs end up on it. At the moment mambo is not able to create multiple xscom controllers. We can create fake ones, just by faking the device tree up, but that seems uglier than this solution.

  So create a mambo-chip for each CPU other than 0, to tell skiboot we want a separate chip created. This then enables Linux to see multiple chips:

```
smp: Brought up 2 nodes, 2 CPUs
numa: Node 0 CPUs: 0
numa: Node 1 CPUs: 1
```

- chip: Add support for discovering chips on mambo

  Currently the only way for skiboot to discover chips is by looking for xscom nodes. But on mambo it's currently not possible to create multiple xscom nodes, which means we can only simulate a single chip system.

  However it seems we can fairly cleanly add support for a special mambo chip node, and use that to instantiate multiple chips.

  Add a check in init_chip() that we're not clobbering an already initialised chip, now that we have two places that initialise chips.

- mambo: Make xscom claim to be DD 2.0

  In the mambo tcl we set the CPU version to DD 2.0, because mambo is not bug compatible with DD 1.

  But in xscom_read_cfam_chipid() we have a hard coded value, to work around the lack of the f000f register, which claims to be P9 DD 1.0.

  This doesn't seem to cause crashes or anything, but at boot we do see:

```
[    0.003893084,5] XSCOM: chip 0x0 at 0x1a0000000000 [P9N DD1.0]
```

So fix it to claim that the xscom is also DD 2.0 to match the CPU.

- mambo: Match whole string when looking up symbols with linsym/skisym

  linsym/skisym use a regex to match the symbol name, and accepts a partial match against the entry in the symbol map, which can lead to somewhat confusing results, eg:

  ```
  systemsim % linsym early_setup
  0xc000000000027890
  systemsim % linsym early_setup$
  0xc000000000aa8054
  systemsim % linsym early_setup_secondary
  0xc000000000027890
  ```

  I don't think that's the behaviour we want, so append a $ to the name so that the symbol has to match against the whole entry, eg:

  ```
  systemsim % linsym early_setup
  0xc000000000aa8054
  ```

- Disable nap on P8 Mambo, public release has bugs
- mambo: Allow loading multiple CPIOs

  Currently we have support for loading a single CPIO and telling Linux to use it as the initrd. But the Linux code actually supports having multiple CPIOs contiguously in memory, between initrd-start and end, and will unpack them all in order. That is a really nice feature as it means you can have a base CPIO with your root filesystem, and then tack on others as you need for various tests etc.

  So expand the logic to handle SKIBOOT_INITRD, and treat it as a comma separated list of CPIOs to load. I chose comma as it's fairly rare in filenames, but we could make it space, colon, whatever. Or we could add a new environment variable entirely. The code also supports trimming whitespace from the values, so you can have "cpio1, cpio2".

- hdata/test: Add memory reservations to hdata_to_dt

  Currently memory reservations are parsed, but since they are not processed until mem_region_init() they don't appear in the output device tree blob. Several bugs have been found with memory reservations so we want them to be part of the test output.

  Add them and clean up several usages of printf() since we want only the dtb to appear in standard out.

pflash/libffs

Since *skiboot-5.7-rc2*:

- pflash option to retrieve PNOR partition flags

  This commit extends pflash with an option to retrieve and print information for a particular partition, including the content from "pflash -i" and a verbose list of set miscellaneous flags. -i option is also updated to print a short list of flags in addition to the ECC flag, with one character per flag. A test of the new option is included in libflash/test.

Since *skiboot-5.6.0*:

- libflash/libffs: Zero checksum words

On writing ffs entries to flash libffs doesn't zero checksum words before calculating the checksum across the entire structure. This causes an inaccurate calculation of the checksum as it may calculate a checksum on non-zero checksum bytes.

- libffs: Fix ffs_lookup_part() return value

  It would return success when the part wasn't found

- libflash/libffs: Correctly update the actual size of the partition

  libffs has been updating FFS partition information in the wrong place which leads to incomplete erases and corruption.

- libflash: Initialise entries list earlier

  In the bail-out path we call ffs_close() to tear down the partially initialised ffs_handle. ffs_close() expects the entries list to be initialised so we need to do that earlier to prevent a null pointer dereference.

mbox-flash

mbox-flash is the emerging standard way of talking to host PNOR flash on POWER9 systems.

- libflash/mbox-flash: Implement MARK_WRITE_ERASED mbox call

  Version two of the mbox-flash protocol defines a new command: MARK_WRITE_ERASED.

  This command provides a simple way to mark a region of flash as all 0xff without the need to go and write all 0xff. This is an optimisation as there is no need for an erase before a write, it is the responsibility of the BMC to deal with the flash correctly, however in v1 it was ambiguous what a client should do if the flash should be erased but not actually written to. This allows of a optimal path to resolve this problem.

- libflash/mbox-flash: Update to V2 of the protocol

  Updated version 2 of the protocol can be found at: https://github.com/openbmc/mboxbridge/blob/master/Documentation/mbox_protocol.md

  This commit changes mbox-flash such that it will preferentially talk version 2 to any capable daemon but still remain capable of talking to v1 daemons.

  Version two changes some of the command definitions for increased consistency and usability. Version two includes more attention bits - these are now dealt with at a simple level.

- libflash/mbox-flash: Implement MARK_WRITE_ERASED mbox call

  Version two of the mbox-flash protocol defines a new command: MARK_WRITE_ERASED.

  This command provides a simple way to mark a region of flash as all 0xff without the need to go and write all 0xff. This is an optimisation as there is no need for an erase before a write, it is the responsibility of the BMC to deal with the flash correctly, however in v1 it was ambiguous what a client should do if the flash should be erased but not actually written to. This allows of a optimal path to resolve this problem.

- libflash/mbox-flash: Update to V2 of the protocol

  Updated version 2 of the protocol can be found at: https://github.com/openbmc/mboxbridge/blob/master/Documentation/mbox_protocol.md

  This commit changes mbox-flash such that it will preferentially talk version 2 to any capable daemon but still remain capable of talking to v1 daemons.

  Version two changes some of the command definitions for increased consistency and usability. Version two includes more attention bits - these are now dealt with at a simple level.

- hw/lpc-mbox: Use message registers for interrupts

  Currently the BMC raises the interrupt using the BMC control register. It does so on all accesses to the 16 'data' registers meaning that when the BMC only wants to set the ATTN (on which we have interrupts enabled) bit we will also get a control register based interrupt.

  The solution here is to mask that interrupt permanantly and enable interrupts on the protocol defined 'response' data byte.

Contributors

- Processed 232 csets from 29 developers.
- 1 employer found
- A total of 13043 lines added, 2517 removed (delta 10526)

Extending the analysis done for some previous releases, we can see our trends in code review across versions:

Release	csets	Ack %	Reviews %	Tested %	Reported %
5.0	329	15 (5%)	20 (6%)	1 (0%)	0 (0%)
5.1	372	13 (3%)	38 (10%)	1 (0%)	4 (1%)
5.2-rc1	334	20 (6%)	34 (10%)	6 (2%)	11 (3%)
5.3-rc1	302	36 (12%)	53 (18%)	4 (1%)	5 (2%)
5.4	361	16 (4%)	28 (8%)	1 (0%)	9 (2%)
5.5	408	11 (3%)	48 (12%)	14 (3%)	10 (2%)
5.6	87	12 (14%)	6 (7%)	5 (6%)	2 (2%)
5.7	232	30 (13%)	32 (14%)	5 (2%)	2 (1%)

This cycle has been good for reviews/acks, scoring second highest percentage ever on both, as well as being right up there on absolute numbers.

Developers with the most changesets

Developer	#	%
Benjamin Herrenschmidt	41	(17.7%)
Stewart Smith	31	(13.4%)
Michael Neuling	28	(12.1%)
Oliver O'Halloran	18	(7.8%)
Vasant Hegde	18	(7.8%)
Jeremy Kerr	12	(5.2%)
Alistair Popple	11	(4.7%)
Gavin Shan	10	(4.3%)
Russell Currey	9	(3.9%)
Michael Ellerman	9	(3.9%)
Madhavan Srinivasan	7	(3.0%)
Cyril Bur	6	(2.6%)
Christophe Lombard	5	(2.2%)
Shilpasri G Bhat	5	(2.2%)
Andrew Donnellan	3	(1.3%)
Nicholas Piggin	3	(1.3%)
Mahesh Salgaonkar	2	(0.9%)
Anju T Sudhakar	2	(0.9%)
Hemant Kumar	2	(0.9%)
Matt Brown	1	(0.4%)
Michael Tritz	1	(0.4%)
Joel Stanley	1	(0.4%)
Balbir Singh	1	(0.4%)
Frederic Barrat	1	(0.4%)
Andrew Jeffery	1	(0.4%)
Pridhiviraj Paidipeddi	1	(0.4%)
Reza Arbab	1	(0.4%)
Suraj Jitindar Singh	1	(0.4%)
Vaibhav Jain	1	(0.4%)

Developers with the most changed lines

Developer	#	%
Hemant Kumar	3056	(23.0%)
Stewart Smith	1826	(13.7%)
Benjamin Herrenschmidt	1348	(10.1%)
Christophe Lombard	937	(7.0%)
Shilpasri G Bhat	770	(5.8%)
Madhavan Srinivasan	755	(5.7%)
Jeremy Kerr	731	(5.5%)
Cyril Bur	674	(5.1%)
Alistair Popple	477	(3.6%)
Gavin Shan	414	(3.1%)
Russell Currey	396	(3.0%)
Michael Neuling	336	(2.5%)
Vasant Hegde	308	(2.3%)
Oliver O'Halloran	300	(2.3%)
Anju T Sudhakar	300	(2.3%)
Michael Tritz	167	(1.3%)
Frederic Barrat	113	(0.8%)
Nicholas Piggin	93	(0.7%)
Mahesh Salgaonkar	76	(0.6%)
Michael Ellerman	66	(0.5%)
Suraj Jitindar Singh	59	(0.4%)
Andrew Donnellan	53	(0.4%)
Joel Stanley	20	(0.2%)
Balbir Singh	12	(0.1%)
Reza Arbab	10	(0.1%)
Vaibhav Jain	9	(0.1%)
Pridhiviraj Paidipeddi	2	(0.0%)
Matt Brown	1	(0.0%)
Andrew Jeffery	1	(0.0%)

Developers with the most signoffs

(total 242)

Developer	#	%
Stewart Smith	201	(83.1%)
Michael Neuling	29	(12.0%)
Madhavan Srinivasan	4	(1.7%)
Suraj Jitindar Singh	3	(1.2%)
Anju T Sudhakar	2	(0.8%)
Hemant Kumar	2	(0.8%)
Cyril Bur	1	(0.4%)

Developers with the most reviews

(total 32)

Developer	#	%
Vasant Hegde	8	(25.0%)
Cyril Bur	7	(21.9%)
Andrew Donnellan	5	(15.6%)
Frederic Barrat	5	(15.6%)
Andrew Jeffery	2	(6.2%)
Gavin Shan	2	(6.2%)
Joel Stanley	1	(3.1%)
Oliver O'Halloran	1	(3.1%)
Alistair Popple	1	(3.1%)

Developers with the most test credits

(total 5)

Developer	#	%
Vasant Hegde	2	(40.0%)
Oliver O'Halloran	1	(20.0%)
Ananth N Mavinakayanahalli	1	(20.0%)
Michael Ellerman	1	(20.0%)

Developers who gave the most tested-by credits

(total 5)

Developer	#	%
Jeremy Kerr	2	(40.0%)
Vasant Hegde	1	(20.0%)
Oliver O'Halloran	1	(20.0%)
Michael Ellerman	1	(20.0%)

Developers with the most report credits

(total 2)

Developer	#	%
Oliver O'Halloran	1	(50.0%)
Alastair D'Silva	1	(50.0%)

Developers who gave the most report credits

(total 2)

Developer	#	%
Andrew Donnellan	1	(50.0%)
Stewart Smith	1	(50.0%)

5.1.83 skiboot-5.7-rc1

skiboot v5.7-rc1 was released on Monday July 3rd 2017. It is the first release candidate of skiboot 5.7, which will become the new stable release of skiboot following the 5.6 release, first released 24th May 2017.

skiboot v5.7-rc1 contains all bug fixes as of *skiboot-5.4.6* and *skiboot-5.1.19* (the currently maintained stable releases). We do not currently expect to do any 5.6.x stable releases.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.7 by July 12th, with skiboot 5.7 being for all POWER8 and POWER9 platforms in op-build v1.18 (Due July 12th). This is a short cycle as this release is mainly targetted towards POWER9 bringup efforts.

This is the second release using the new regular six week release cycle, similar to op-build, but slightly offset to allow for a short stabilisation period. Expected release dates and contents are tracked using GitHub milestone and issues: https://github.com/open-power/skiboot/milestones

Over skiboot-5.6, we have the following changes:

New Features

New features in this release for POWER9 systems:

- In Memory Counters (IMC) (See *OPAL/Skiboot In-Memory Collection (IMC) interface Documentation* for details)
- phb4: Activate shared PCI slot on witherspoon (see *Shared Slot*)
- phb4 capi (i.e. CAPI2): Enable capi mode for PHB4 (see *CAPI on PHB4*)

New feature for IBM FSP based systems:

- fsp/tpo: Provide support for disabling TPO alarm

  This patch adds support for disabling a preconfigured Timed-Power-On(TPO) alarm on FSP based systems. Presently once a TPO alarm is configured from the kernel it will be triggered even if its subsequently disabled.

  With this patch a TPO alarm can be disabled by passing y_m_d==hr_min==0 to fsp_opal_tpo_write(). A branch is added to the function to handle this case by sending FSP_CMD_TPO_DISABLE message to the FSP instead of usual FSP_CMD_TPO_WRITE message. The kernel is expected to call opal_tpo_write() with y_m_d==hr_min==0 to request opal to disable TPO alarm.

POWER9

Development on POWER9 systems continues in earnest.

This release includes the first support for POWER9 DD2 chips. Future releases will likely contain more bug fixes, this release has booted on real hardware.

- hdata: Reserve Trace Areas

  When hostboot is configured to setup in memory tracing it will reserve some memory for use by the hardware tracing facility. We need to mark these areas as off limits to the operating system and firmware.

- hdata: Make out-of-range idata print at PR_DEBUG

  Some fields just aren't populated on some systems.

- hdata: Ignore unnamed memory reservations.

  Hostboot should name any and all memory reservations that it provides. Currently some hostboots export a broken reservation covering the first 256MB of memory and this causes the system to crash at boot due to an invalid free because this overlaps with the static "ibm,os-reserve" region (which covers the first 768MB of memory).

  According to the hostboot team unnamed reservations are invalid and can be ignored.

- hdata: Check the Host I2C devices array version

  Currently this is not populated on FSP machines which causes some obnoxious errors to appear in the boot log. We also only want to parse version 1 of this structure since future versions will completely change the array item format.

- Ensure P9 DD1 workarounds apply only to Nimbus

  The workarounds for P9 DD1 are only needed for Nimbus. P9 Cumulus will be DD1 but don't need these same workarounds.

  This patch ensures the P9 DD1 workarounds only apply to Nimbus. It also renames some things to make clear what's what.

- cpu: Cleanup AMR and IAMR when re-initializing CPUs

  There's a bug in current Linux kernels leaving crap in those registers accross kexec and not sanitizing them on boot. This breaks kexec under some circumstances (such as booting a hash kernel from a radix one on P9 DD2.0).

  The long term fix is in Linux, but this workaround is a reasonable way of "sanitizing" those SPRs when Linux calls opal_reinit_cpus() and shouldn't have adverse effects.

  We could also use that same mechanism to cleanup other things as well such as restoring some other SPRs to their default value in the future.

- Set POWER9 RPR SPR to 0x00000103070F1F3F. Same value as P8.

  Without this, thread priorities inside a core don't work.

- cpu: Support setting HID[RADIX] and set it by default on P9

  This adds new opal_reinit_cpus() flags to setup radix or hash mode in HID[8] on POWER9.

  By default HID[8] will be set. On P9 DD1.0, Linux will change it as needed. On P9 DD2.0 hash works in radix mode (radix is really "dual" mode) so KVM won't break and existing kernels will work.

  Newer kernels built for hash will call this to clear the HID bit and thus get the full size of the TLB as an optimization.

- Add "cleanup_global_tlb" for P9 and later

  Uses broadcast TLBIE's to cleanup the TLB on all cores and on the nest MMU

- xive: DD2.0 updates

  Add support for StoreEOI, fix StoreEOI MMIO offset in ESB page, and other cleanups

- Update default TSCR value for P9 as recommended by HW folk.

- xive: Fix initialisation of xive_cpu_state struct

  When using XIVE emulation with DEBUG=1, we run into crashes in log_add() due to the xive_cpu_state->log_pos being uninitialised (and thus, with DEBUG enabled, initialised to the poison value of 0x99999999).

OCC/Power Management

With this release, it's possible to boot POWER9 systems with the OCC enabled and change CPU frequencies. Doing so does require other firmware components to also support this (otherwise the frequency will not be set).

- occ: Skip setting cores to nominal frequency in P9

  In P9, once OCC is up, it is supposed to setup the cores to nominal frequency. So skip this step in OPAL.

- occ: Fix Pstate ordering for P9

  In P9 the pstate values are positive. They are continuous set of unsigned integers [0 to +N] where Pmax is 0 and Pmin is N. The linear ordering of pstates for P9 has changed compared to P8. P8 has neagtive pstate values advertised as [0 to -N] where Pmax is 0 and Pmin is -N. This patch adds helper routines to abstract pstate comparison with pmax and adds sanity pstate limit checks. This patch also fixes pstate arithmetic by using labs().

- p8-i2c: occ: Add support for OCC to use I2C engines

  This patch adds support to share the I2C engines with host and OCC. OCC uses I2C engines to read DIMM temperatures and to communicate with GPU. OCC Flag register is used for locking between host and OCC. Host requests for the bus by setting a bit in OCC Flag register. OCC sends an interrupt to indicate the change in ownership.

opal-prd/PRD

- opal-prd: Handle SBE passthrough message passing

  This patch adds support to send SBE pass through command to HBRT.

- SBE: Add passthrough command support

  SBE sends passthrough command. We have to capture this interrupt and send event to HBRT via opal-prd (user space daemon).

- opal-prd: hook up reset_pm_complex

  This change provides the facility to invoke HBRT's reset_pm_complex, in the same manner is done with process_occ_reset previously.

  We add a control command for *opal-prd pm-complex reset*, which is just an alias for occ_reset at this stage.

- prd: Implement firmware side of opaque PRD channel

  This change introduces the firmware side of the opaque HBRT <–> OPAL message channel. We define a base message format to be shared with HBRT (in include/prd-fw-msg.h), and allow firmware requests and responses to be sent over this channel.

  We don't currently have any notifications defined, so have nothing to do for firmware_notify() at this stage.

- opal-prd: Add firmware_request & firmware_notify implementations

  This change adds the implementation of firmware_request() and firmware_notify(). To do this, we need to add a message queue, so that we can properly handle out-of-order messages coming from firmware.

- opal-prd: Add support for variable-sized messages

  With the introductuion of the opaque firmware channel, we want to support variable-sized messages. Rather than expecting to read an entire 'struct opal_prd_msg' in one read() call, we can split this over mutiple reads, potentially expanding our message buffer.

- opal-prd: Sync hostboot interfaces with HBRT

  This change adds new callbacks defined for p9, and the base thunks for the added calls.

- opal-prd: interpret log level prefixes from HBRT

  Interpret the (optional) *_MRK log prefixes on HBRT messages, and set the syslog log priority to suit.

- opal-prd: Add occ reset to usage text
- opal-prd: allow different chips for occ control actions

  The *occ reset* and *occ error* actions can both take a chip id argument, but we're currently just using zero. This change changes the control message format to pass the chip ID from the control process to the opal-prd daemon.

PCI/PHB4

- phb4: Fix number of index bits in IODA tables

  On PHB4 the number of index bits in the IODA table address register was bumped to 10 bits to accomodate for 1024 MSIs and 1024 TVEs (DD2).

  However our macro only defined the field to be 9 bits, thus causing "interesting" behaviours on some systems.

- phb4: Harden init with bad PHBs

  Currently if we read all 1's from the EEH or IRQ capabilities, we end up train wrecking on some other random code (eg. an assert() in xive).

  This hardens the PHB4 code to look for these bad reads and more gracefully fails the init for that PHB alone. This allows the rest of the system to boot and ignore those bad PHBs.

- phb4 capi (i.e. CAPI2): Handle HMI events

  Find the CAPP on the chip associated with the HMI event for PHB4. The recovery mode (re-initialization of the capp, resume of functional operations) is only available with P9 DD2. A new patch will be provided to support this feature.

- phb4 capi (i.e. CAPI2): Enable capi mode for PHB4

  Enable the Coherently attached processor interface. The PHB is used as a CAPI interface. CAPI Adapters can be connected to either PEC0 or PEC2. Single port CAPI adapter can be connected to either PEC0 or PEC2, but Dual-Port Adapter can be only connected to PEC2 * CAPP0 attached to PHB0(PEC0 - single port) * CAPP1 attached to PHB3(PEC2 - single or dual port)

- hw/phb4: Rework phb4_get_presence_state()

  There are two issues in current implementation: It should return errcode visibile to Linux, which has prefix OPAL_*. The code isn't very obvious.

  This returns OPAL_HARDWARE when the PHB is broken. Otherwise, OPAL_SUCCESS is always returned. In the mean while, It refactors the code to make it obvious: OPAL_PCI_SLOT_PRESENT is returned when the presence signal (low active) or PCIe link is active. Otherwise, OPAL_PCI_SLOT_EMPTY is returned.

- phb4: Error injection for config space

  Implement CFG (config space) error injection.

  This works the same as PHB3. MMIO and DMA error injection require a rewrite, so they're unsupported for now.

  While it's not feature complete, this at least provides an easy way to inject an error that will trigger EEH.

- phb4: Error clear implementation

- phb4: Mask link down errors during reset

  During a hot reset the PCI link will drop, so we need to mask link down events to prevent unnecessary errors.

- phb4: Implement root port initialization

  phb4_root_port_init() was a NOP before, so fix that.

- phb4: Complete reset implementation

  This implements complete reset (creset) functionality for POWER9 DD1.

  Only partially tested and contends with some DD1 errata, but it's a start.

- phb4: Activate shared PCI slot on witherspoon

  Witherspoon systems come with a 'shared' PCI slot: physically, it looks like a x16 slot, but it's actually two x8 slots connected to two PHBs of two different chips. Taking advantage of it requires some logic on the PCI adapter. Only the Mellanox CX5 adapter is known to support it at the time of this writing.

  This patch enables support for the shared slot on witherspoon if a x16 adapter is detected. Each x8 slot has a presence bit, so both bits need to be set for the activation to take place. Slot sharing is activated through a gpio.

  Note that there's no easy way to be sure that the card is indeed a shared-slot compatible PCI adapter and not a normal x16 card. Plugging a normal x16 adapter on the shared slot should be avoided on witherspoon, as the link won't train on the second slot, resulting in a timeout and a longer boot time. Only the first slot is usable and the x16 adapter will end up using only half the lines.

  If the PCI card plugged on the physical slot is only x8 (or less), then the presence bit of the second slot is not set, so this patch does nothing. The x8 (or less) adapter should work like on any other physical slot.

- phb4: Block D-state power management on direct slots

  As current revisions of PHB4 don't properly handle the resulting L1 link transition.

- phb4: Call pci config filters
- phb4: Mask out write-1-to-clear registers in RC cfg

  The root complex config space only supports 4-byte accesses. Thus, when the client requests a smaller size write, we do a read-modify-write to the register.

  However, some register have bits defined as "write 1 to clear".

  If we do a RMW cycles on such a register and such bits are 1 in the part that the client doesn't intend to modify, we will accidentally write back those 1's and clear the corresponding bit.

  This avoids it by masking out those magic bits from the "old" value read from the register.

- phb4: Properly mask out link down errors during reset
- phb3/4: Silence a useless warning

  PHB's don't have base location codes on non-FSP systems and it's normal.

- phb4: Workaround bug in spec 053

  Wait for DLP PGRESET to clear *after* lifting the PCIe core reset

- phb4: DD2.0 updates

  Support StoreEOI, full complements of PEs (twice as big TVT) and other updates.

  Also renumber init steps to match spec 063

NPU2

Note that currently NPU2 support is limited to POWER9 DD1 hardware.

- platforms/astbmc/witherspoon.c: Add NPU2 slot mappings

  For NVLink2 to function PCIe devices need to be associated with the right NVLinks. This association is supposed to be passed down to Skiboot via HDAT but those fields are still not correctly filled out. To work around this we add slot tables for the NVLinks similar to what we have for P8+.

- hw/npu2.c: Fix device aperture calculation

  The POWER9 NPU2 implements an address compression scheme to compress 56-bit P9 physical addresses to 47-bit GPU addresses. System software needs to know both addresses, unfortunately the calculation of the compressed address was incorrect. Fix it here.

- hw/npu2.c: Change MCD BAR allocation order

  MCD BARs need to be correctly aligned to the size of the region. As GPU memory is allocated from the top of memory down we should start allocating from the highest GPU memory address to the lowest to ensure correct alignment.

- NPU2: Add flag to nvlink config space indicating DL reset state

  Device drivers need to be able to determine if the DL is out of reset or not so they can safely probe to see if links have already been trained. This patch adds a flag to the vendor specific config space indicating if the DL is out of reset.

- hw/npu2.c: Hardcode MSR_SF when setting up npu XTS contexts

  We don't support anything other than 64-bit mode for address translations so we can safely hardcode it.

- hw/npu2-hw-procedures.c: Add nvram option to override zcal calculations

  In some rare cases the zcal state machine may fail and flag an error. According to hardware designers it is sometimes ok to ignore this failure and use nominal values for the calculations. In this case we add a nvram variable (nv_zcal_override) which will cause skiboot to ignore the failure and use the nominal value specified in nvram.

- npu2: Fix npu2_{read,write}_4b()

  When writing or reading 4-byte values, we need to use the upper half of the 64-bit SCOM register.

  Fix npu2_{read,write}_4b() and their callers to use uint32_t, and appropriately shift the value being written or returned.

- hw/npu2.c: Fix opal_npu_map_lpar to search for existing BDF
- hw/npu2-hw-procedures.c: Fix running of zcal procedure

  > The zcal procedure should only be run once per obus (ie. once per group of 3 links). Clean up the code and fix the potential buffer overflow due to a typo. Also updates the zcal settings to their proper values.

- hw/npu2.c: Add memory coherence directory programming

  The memory coherence directory (MCD) needs to know which system memory addresses belong to the GPU. This amounts to setting a BAR and a size in the MCD to cover the addresses assigned to each of the GPUs. To ease assignment we assume GPUs are assigned memory in a contiguous block per chip.

pflash/libflash

- libflash/libffs: Zero checksum words

  On writing ffs entries to flash libffs doesn't zero checksum words before calculating the checksum across the entire structure. This causes an inaccurate calculation of the checksum as it may calculate a checksum on non-zero checksum bytes.

- libffs: Fix ffs_lookup_part() return value

  It would return success when the part wasn't found

- libflash/libffs: Correctly update the actual size of the partition

  libffs has been updating FFS partition information in the wrong place which leads to incomplete erases and corruption.

- libflash: Initialise entries list earlier

  In the bail-out path we call ffs_close() to tear down the partially initialised ffs_handle. ffs_close() expects the entries list to be initialised so we need to do that earlier to prevent a null pointer dereference.

mbox-flash

mbox-flash is the emerging standard way of talking to host PNOR flash on POWER9 systems.

- libflash/mbox-flash: Implement MARK_WRITE_ERASED mbox call

  Version two of the mbox-flash protocol defines a new command: MARK_WRITE_ERASED.

  This command provides a simple way to mark a region of flash as all 0xff without the need to go and write all 0xff. This is an optimisation as there is no need for an erase before a write, it is the responsibility of the BMC to deal with the flash correctly, however in v1 it was ambiguous what a client should do if the flash should be erased but not actually written to. This allows of a optimal path to resolve this problem.

- libflash/mbox-flash: Update to V2 of the protocol

  Updated version 2 of the protocol can be found at: https://github.com/openbmc/mboxbridge/blob/master/Documentation/mbox_protocol.md

  This commit changes mbox-flash such that it will preferentially talk version 2 to any capable daemon but still remain capable of talking to v1 daemons.

  Version two changes some of the command definitions for increased consistency and usability. Version two includes more attention bits - these are now dealt with at a simple level.

- libflash/mbox-flash: Implement MARK_WRITE_ERASED mbox call

  Version two of the mbox-flash protocol defines a new command: MARK_WRITE_ERASED.

  This command provides a simple way to mark a region of flash as all 0xff without the need to go and write all 0xff. This is an optimisation as there is no need for an erase before a write, it is the responsibility of the BMC to deal with the flash correctly, however in v1 it was ambiguous what a client should do if the flash should be erased but not actually written to. This allows of a optimal path to resolve this problem.

- libflash/mbox-flash: Update to V2 of the protocol

  Updated version 2 of the protocol can be found at: https://github.com/openbmc/mboxbridge/blob/master/Documentation/mbox_protocol.md

  This commit changes mbox-flash such that it will preferentially talk version 2 to any capable daemon but still remain capable of talking to v1 daemons.

Version two changes some of the command definitions for increased consistency and usability. Version two includes more attention bits - these are now dealt with at a simple level.

- hw/lpc-mbox: Use message registers for interrupts

  Currently the BMC raises the interrupt using the BMC control register. It does so on all accesses to the 16 'data' registers meaning that when the BMC only wants to set the ATTN (on which we have interrupts enabled) bit we will also get a control register based interrupt.

  The solution here is to mask that interrupt permanantly and enable interrupts on the protocol defined 'response' data byte.

General fixes

- Reduce log level on non-error log messages

  90% of what we print isn't useful to a normal user. This dramatically reduces the amount of messages printed by OPAL in normal circumstances.

- init: Silence messages and call ourselves "OPAL"
- psi: Switch to ESB mode later

  There's an errata, if we switch to ESB mode before setting up the various ESB mode related registers, a pending interrupts can go wrong.

- lpc: Enable "new" SerIRQ mode
- hw/ipmi/ipmi-sel: missing newline in prlog warning
- p8-i2c OCC lock: fix locking in p9_i2c_bus_owner_change
- Convert important polling loops to spin at lowest SMT priority

  The pattern of calling cpu_relax() inside a polling loop does not suit the powerpc SMT priority instructions. Prefrred is to set a low priority then spin until break condition is reached, then restore priority.

- Improve cpu_idle when PM is disabled

  Split cpu_idle() into cpu_idle_delay() and cpu_idle_job() rather than requesting the idle type as a function argument. Have those functions provide a default polling (non-PM) implentation which spin at the lowest SMT priority.

- core/fdt: Always add a reserve map

  Currently we skip adding the reserved ranges block to the generated FDT blob if we are excluding the root node. This can result in a DTB that dtc will barf on because the reserved memory ranges overlap with the start of the dt_struct block. As an example:

```
$ fdtdump broken.dtb -d
/dts-v1/;
// magic:               0xd00dfeed
// totalsize:           0x7f3 (2035)
// off_dt_struct:       0x30  <----\
// off_dt_strings:      0x7b8       | this is bad!
// off_mem_rsvmap:      0x30  <----/
// version:             17
// last_comp_version:   16
// boot_cpuid_phys:     0x0
// size_dt_strings:     0x3b
// size_dt_struct:      0x788
```

(continues on next page)

(continued from previous page)

```
/memreserve/ 0x100000000 0x300000004;
/memreserve/ 0x3300000001 0x169626d2c;
/memreserve/ 0x706369652d736c6f 0x7473000000000003;
        *continues*
```

With this patch:

```
$ fdtdump working.dtb -d
/dts-v1/;
// magic:               0xd00dfeed
// totalsize:           0x803 (2051)
// off_dt_struct:       0x40
// off_dt_strings:      0x7c8
// off_mem_rsvmap:      0x30
// version:             17
// last_comp_version:   16
// boot_cpuid_phys:     0x0
// size_dt_strings:     0x3b
// size_dt_struct:      0x788

// 0040: tag: 0x00000001 (FDT_BEGIN_NODE)
/ {
// 0048: tag: 0x00000003 (FDT_PROP)
// 07fb: string: phandle
// 0054: value
    phandle = <0x00000001>;
        *continues*
```

- hw/lpc-mbox: Use message registers for interrupts

  Currently the BMC raises the interrupt using the BMC control register. It does so on all accesses to the 16 'data' registers meaning that when the BMC only wants to set the ATTN (on which we have interrupts enabled) bit we will also get a control register based interrupt.

  The solution here is to mask that interrupt permanantly and enable interrupts on the protocol defined 'response' data byte.

PCI

- pci: Wait 20ms before checking presence detect on PCIe

  As the PHB presence logic has a debounce timer that can take a while to settle.

- phb3+iov: Fixup support for config space filters

  The filter should be called before the HW access and its return value control whether to perform the access or not

- core/pci: Use PCI slot's power facality in pci_enable_bridge()

  The current implmentation has incorrect assumptions: there is always a PCI slot associated with root port and PCIe switch downstream port and all of them are capable to change its power state by register PCICAP_EXP_SLOTCTL. Firstly, there might not a PCI slot associated with the root port or PCIe switch downstream port. Secondly, the power isn't controlled by standard config register (PCICAP_EXP_SLOTCTL). There are I2C slave devices used to control the power states on Tuleta.

  In order to use the PCI slot's methods to manage the power states, this does:

  - Introduce PCI_SLOT_FLAG_ENFORCE, indicates the request operation is enforced to be applied.

  - pci_enable_bridge() is split into 3 functions: pci_bridge_power_on() to power it on; pci_enable_bridge() as a place holder and pci_bridge_wait_link() to wait the downstream link to come up.
  - In pci_bridge_power_on(), the PCI slot's specific power management methods are used if there is a PCI slot associated with the PCIe switch downstream port or root port.

- platforms/astbmc/slots.c: Allow comparison of bus numbers when matching slots

  When matching devices on multiple down stream PLX busses we need to compare more than just the device-id of the PCIe BDFN, so increase the mask to do so.

Tests and simulators

- boot-tests: add OpenBMC support
- boot_test.sh: Add SMC BMC support

  Your BMC needs a special debug image flashed to use this, the exact image and methods aren't something I can publish here, but if you work for IBM or SMC you can find out from the right sources.

  A few things are needed to move around to be able to flash to a SMC BMC.

  For a start, the SSH daemon will only accept connections after a special incantation (which I also can't share), but you should put that in the ~/.skiboot_boot_tests file along with some other default login information we don't publicise too broadly (because Security Through Obscurity is *obviously* a good idea. . . .)

  We also can't just directly "ssh /bin/true", we need an expect script, and we can't scp, but we can anonymous rsync!

  You also need a pflash binary to copy over.

- hdata_to_dt: Add PVR overrides to the usage text
- mambo: Add a reservation for the initramfs

  On most systems the initramfs is loaded inside the part of memory reserved for the OS [0x0-0x30000000] and skiboot will never touch it. On mambo it's loaded at 0x80000000 and if you're unlucky skiboot can allocate over the top of it and corrupt the initramfs blob.

  There might be the downside that the kernel cannot re-use the initramfs memory since it's marked as reserved, but the kernel might also free it anyway.

- mambo: Update P9 PVR to reflect Scale out 24 core chips

  The P9 PVR bits 48:51 don't indicate a revision but instead different configurations. From BookIV we have:

Bits	Configuration
0	Scale out 12 cores
1	Scale out 24 cores
2	Scale up 12 cores
3	Scale up 24 cores

  Skiboot will mostly the use "Scale out 24 core" configuration (ie. SMT4 not SMT8) so reflect this in mambo.

- core: Move enable_mambo_console() into chip initialisation

  Rather than having a wart in main_cpu_entry() that initialises the mambo console, we can move it into init_chips() which is where we discover that we're on mambo.

- mambo: Create multiple chips when we have multiple CPUs

Currently when we boot mambo with multiple CPUs, we create multiple CPU nodes in the device tree, and each claims to be on a separate chip.

However we don't create multiple xscom nodes, which means skiboot only knows about a single chip, and all CPUs end up on it. At the moment mambo is not able to create multiple xscom controllers. We can create fake ones, just by faking the device tree up, but that seems uglier than this solution.

So create a mambo-chip for each CPU other than 0, to tell skiboot we want a separate chip created. This then enables Linux to see multiple chips:

```
smp: Brought up 2 nodes, 2 CPUs
numa: Node 0 CPUs: 0
numa: Node 1 CPUs: 1
```

- chip: Add support for discovering chips on mambo

  Currently the only way for skiboot to discover chips is by looking for xscom nodes. But on mambo it's currently not possible to create multiple xscom nodes, which means we can only simulate a single chip system.

  However it seems we can fairly cleanly add support for a special mambo chip node, and use that to instantiate multiple chips.

  Add a check in init_chip() that we're not clobbering an already initialised chip, now that we have two places that initialise chips.

- mambo: Make xscom claim to be DD 2.0

  In the mambo tcl we set the CPU version to DD 2.0, because mambo is not bug compatible with DD 1.

  But in xscom_read_cfam_chipid() we have a hard coded value, to work around the lack of the f000f register, which claims to be P9 DD 1.0.

  This doesn't seem to cause crashes or anything, but at boot we do see:

```
[    0.003893084,5] XSCOM: chip 0x0 at 0x1a0000000000 [P9N DD1.0]
```

  So fix it to claim that the xscom is also DD 2.0 to match the CPU.

- mambo: Match whole string when looking up symbols with linsym/skisym

  linsym/skisym use a regex to match the symbol name, and accepts a partial match against the entry in the symbol map, which can lead to somewhat confusing results, eg:

```
systemsim % linsym early_setup
0xc000000000027890
systemsim % linsym early_setup$
0xc000000000aa8054
systemsim % linsym early_setup_secondary
0xc000000000027890
```

  I don't think that's the behaviour we want, so append a $ to the name so that the symbol has to match against the whole entry, eg:

```
systemsim % linsym early_setup
0xc000000000aa8054
```

- Disable nap on P8 Mambo, public release has bugs
- mambo: Allow loading multiple CPIOs

  Currently we have support for loading a single CPIO and telling Linux to use it as the initrd. But the Linux code actually supports having multiple CPIOs contiguously in memory, between initrd-start and end, and will unpack them all in order. That is a really nice feature as it means you can have a base CPIO with your root filesystem, and then tack on others as you need for various tests etc.

So expand the logic to handle SKIBOOT_INITRD, and treat it as a comma separated list of CPIOs to load. I chose comma as it's fairly rare in filenames, but we could make it space, colon, whatever. Or we could add a new environment variable entirely. The code also supports trimming whitespace from the values, so you can have "cpio1, cpio2".

- hdata/test: Add memory reservations to hdata_to_dt

  Currently memory reservations are parsed, but since they are not processed until mem_region_init() they don't appear in the output device tree blob. Several bugs have been found with memory reservations so we want them to be part of the test output.

  Add them and clean up several usages of printf() since we want only the dtb to appear in standard out.

IBM FSP systems

- FSP/CONSOLE: Fix possible NULL dereference
- platforms/ibm-fsp/firenze: Fix PCI slot power-off pattern

  When powering off the PCI slot, the corresponding bits should be set to 0bxx00xx00 instead of 0bxx11xx11. Otherwise, the specified PCI slot can't be put into power-off state. Fortunately, it didn't introduce any side-effects so far.

- FSP/CONSOLE: Workaround for unresponsive ipmi daemon

  We use TCE mapped area to write data to console. Console header (fsp_serbuf_hdr) is modified by both FSP and OPAL (OPAL updates next_in pointer in fsp_serbuf_hdr and FSP updates next_out pointer).

  Kernel makes opal_console_write() OPAL call to write data to console. OPAL write data to TCE mapped area and sends MBOX command to FSP. If our console becomes full and we have data to write to console, we keep on waiting until FSP reads data.

  In some corner cases, where FSP is active but not responding to console MBOX message (due to buggy IPMI) and we have heavy console write happening from kernel, then eventually our console buffer becomes full. At this point OPAL starts sending OPAL_BUSY_EVENT to kernel. Kernel will keep on retrying. This is creating kernel soft lockups. In some extreme case when every CPU is trying to write to console, user will not be able to ssh and thinks system is hang.

  If we reset FSP or restart IPMI daemon on FSP, system recovers and everything becomes normal.

  This patch adds workaround to above issue by returning OPAL_HARDWARE when cosole is full. Side effect of this patch is, we may endup dropping latest console data. But better to drop console data than system hang.

- FSP: Set status field in response message for timed out message

  For timed out FSP messages, we set message status as "fsp_msg_timeout". But most FSP driver users (like surviellance) are ignoring this field. They always look for FSP returned status value in callback function (second byte in word1). So we endup treating timed out message as success response from FSP.

  Sample output:

  ```
  [69902.432509048,7] SURV: Sending the heartbeat command to FSP
  [70023.226860117,4] FSP: Response from FSP timed out, word0 = d66a00d7, word1 = 0␣
  ↪state: 3
  ....
  [70023.226901445,7] SURV: Received heartbeat acknowledge from FSP
  [70023.226903251,3] FSP: fsp_trigger_reset() entry
  ```

Here SURV code thought it got valid response from FSP. But actually we didn't receive response from FSP.

This patch fixes above issue by updating status field in response structure.

- FSP: Improve timeout message
- FSP/RTC: Fix possible FSP R/R issue in rtc write path
- hw/fsp/rtc: read/write cached rtc tod on fsp hir.

  Currently fsp-rtc reads/writes the cached RTC TOD on an fsp reset. Use latest fsp_in_rr() function to properly read the cached rtc value when fsp reset initiated by the hir.

  Below is the kernel trace when we set hw clock, when hir process starts.

```
[ 1727.775824] NMI watchdog: BUG: soft lockup - CPU#57 stuck for 23s!␣
↪[hwclock:7688]
[ 1727.775856] Modules linked in: vmx_crypto ibmpowernv ipmi_powernv uio_pdrv_
↪genirq ipmi_devintf powernv_op_panel uio ipmi_msghandler powernv_rng leds_
↪powernv ip_tables x_tables autofs4 ses enclosure scsi_transport_sas crc32c_
↪vpmsum lpfc ipr tg3 scsi_transport_fc
[ 1727.775883] CPU: 57 PID: 7688 Comm: hwclock Not tainted 4.10.0-14-generic #16-
↪Ubuntu
[ 1727.775883] task: c000000fdfdc8400 task.stack: c000000fdfef4000
[ 1727.775884] NIP: c00000000090540c LR: c0000000000846f4 CTR: 000000003006dd70
[ 1727.775885] REGS: c000000fdfef79a0 TRAP: 0901   Not tainted  (4.10.0-14-
↪generic)
[ 1727.775886] MSR: 9000000000009033 <SF,HV,EE,ME,IR,DR,RI,LE>
[ 1727.775889]   CR: 28024442  XER: 20000000
[ 1727.775890] CFAR: c00000000008472c SOFTE: 1
               GPR00: 0000000030005128 c000000fdfef7c20 c00000000144c900␣
↪fffffffffffffff4
               GPR04: 0000000028024442 c00000000090540c 9000000000009033␣
↪0000000000000000
               GPR08: 0000000000000000 0000000031fc4000 c000000000084710␣
↪9000000000001003
               GPR12: c0000000000846e8 c00000000fba0100
[ 1727.775897] NIP [c00000000090540c] opal_set_rtc_time+0x4c/0xb0
[ 1727.775899] LR [c0000000000846f4] opal_return+0xc/0x48
[ 1727.775899] Call Trace:
[ 1727.775900] [c000000fdfef7c20] [c00000000090540c] opal_set_rtc_time+0x4c/0xb0␣
↪(unreliable)
[ 1727.775901] [c000000fdfef7c60] [c000000000900828] rtc_set_time+0xb8/0x1b0
[ 1727.775903] [c000000fdfef7ca0] [c000000000902364] rtc_dev_ioctl+0x454/0x630
[ 1727.775904] [c000000fdfef7d40] [c00000000035b1f4] do_vfs_ioctl+0xd4/0x8c0
[ 1727.775906] [c000000fdfef7de0] [c00000000035bab4] SyS_ioctl+0xd4/0xf0
[ 1727.775907] [c000000fdfef7e30] [c00000000000b184] system_call+0x38/0xe0
[ 1727.775908] Instruction dump:
[ 1727.775909] f821ffc1 39200000 7c832378 91210028 38a10020 39200000 38810028␣
↪f9210020
[ 1727.775911] 4bfffe6d e8810020 80610028 4b77f61d <60000000> 7c7f1b78 3860000a␣
↪2fbffff4
```

  This is found when executing the testcase https://github.com/open-power/op-test-framework/blob/master/testcases/fspresetReload.py

  With this fix ran fsp hir torture testcase in the above test which is working fine.

- occ: Set return variable to correct value

  When entering this section of code rc will be zero. If fsp_mkmsg() fails the code responsible for printing an error message won't be set. Resetting rc should allow for the error case to trigger if fsp_mkmsg fails.

- capp: Fix hang when CAPP microcode LID is missing on FSP machine

  When the LID is absent, we fail early with an error from start_preload_resource. In that case, capp_ucode_info.load_result isn't set properly causing a subsequent capp_lid_download() to call wait_for_resource_loaded() on something that isn't being loaded, thus hanging.

- FSP: Add check to detect FSP R/R inside fsp_sync_msg()

  OPAL sends MBOX message to FSP and updates message state from fsp_msg_queued -> fsp_msg_sent. fsp_sync_msg() queues message and waits until we get response from FSP. During FSP R/R we move outstanding MBOX messages from msgq to rr_queue including inflight message (fsp_reset_cmdclass()). But we are not resetting inflight message state.

  In extreme croner case where we sent message to FSP via fsp_sync_msg() path and FSP R/R happens before getting respose from FSP, then we will endup waiting in fsp_sync_msg() until everything becomes normal.

  This patch adds fsp_in_rr() check to fsp_sync_msg() and return error to caller if FSP is in R/R.

- FSP: Add check to detect FSP R/R inside fsp_sync_msg()

  OPAL sends MBOX message to FSP and updates message state from fsp_msg_queued -> fsp_msg_sent. fsp_sync_msg() queues message and waits until we get response from FSP. During FSP R/R we move outstanding MBOX messages from msgq to rr_queue including inflight message (fsp_reset_cmdclass()). But we are not resetting inflight message state.

  In extreme croner case where we sent message to FSP via fsp_sync_msg() path and FSP R/R happens before getting respose from FSP, then we will endup waiting in fsp_sync_msg() until everything becomes normal.

  This patch adds fsp_in_rr() check to fsp_sync_msg() and return error to caller if FSP is in R/R.

- capp: Fix hang when CAPP microcode LID is missing on FSP machine

  When the LID is absent, we fail early with an error from start_preload_resource. In that case, capp_ucode_info.load_result isn't set properly causing a subsequent capp_lid_download() to call wait_for_resource_loaded() on something that isn't being loaded, thus hanging.

- FSP/CONSOLE: Do not free fsp_msg in error path

  as we reuse same msg to send next output message.

- platform/zz: Acknowledge OCC_LOAD mbox message in ZZ

  In P9 FSP box, OCC image is pre-loaded. So do not handle the load command and send SUCCESS to FSP on recieving OCC_LOAD mbox message.

- FSP/RTC: Improve error log

astbmc systems

- platforms/astbmc: Don't validate model on palmetto

  The platform isn't compatible with palmetto until the root device-tree node's "model" property is NULL or "palmetto". However, we could have "TN71-BP012" for the property on palmetto.

  ```
  linux# cat /proc/device-tree/model
  TN71-BP012
  ```

  This skips the validation on root device-tree node's "model" property on palmetto, meaning we check the "compatible" property only.

5.1.84 skiboot-5.7-rc2

skiboot v5.7-rc2 was released on Thursday July 13th 2017. It is the second release candidate of skiboot 5.7, which will become the new stable release of skiboot following the 5.6 release, first released 24th May 2017.

skiboot v5.7-rc2 contains all bug fixes as of *skiboot-5.4.6* and *skiboot-5.1.19* (the currently maintained stable releases). We do not currently expect to do any 5.6.x stable releases.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.7 in the next week or so, with skiboot 5.7 being for all POWER8 and POWER9 platforms in op-build v1.18 (due July 12th, but will come *after* skiboot 5.7).

This is the second release using the new regular six week release cycle, similar to op-build, but slightly offset to allow for a short stabilisation period. Expected release dates and contents are tracked using GitHub milestone and issues: https://github.com/open-power/skiboot/milestones

Over *skiboot-5.7-rc1*, we have the following changes:

POWER9

There are many important changes for POWER9 DD1 and DD2 systems. POWER9 support should be considered in development and skiboot 5.7 is certainly **NOT** suitable for POWER9 production environments.

- HDAT: Add IPMI sensor data under /bmc node
- numa/associativity: Add a new level of NUMA for GPU's

  Today we have an issue where the NUMA nodes corresponding to GPU's have the same affinity/distance as normal memory nodes. Our reference-points today supports two levels [0x4, 0x4] for normal systems and [0x4, 0x3] for Power8E systems. This patch adds a new level [0x4, X, 0x2] and uses node-id as at all levels for the GPU.
- xive: Enable memory backing of queues

  This dedicates 6x64k pages of memory permanently for the XIVE to use for internal queue overflow. This allows the XIVE to deal with some corner cases where the internal queues might prove insufficient.
- xive: Properly get rid of donated indirect pages during reset

  Otherwise they keep being used accross kexec causing memory corruption in subsequent kernels once KVM has been used.
- cpu: Better handle unknown flags in opal_reinit_cpus()

  At the moment, if we get passed flags we don't know about, we return OPAL_UNSUPPORTED but we still perform whatever actions was requied by the flags we do support. Additionally, on P8, we attempt a SLW re-init which hasn't been supported since Murano DD2.0 and will crash your system.

  It's too late to fix on existing systems so Linux will have to be careful at least on P8, but to avoid future issues let's clean that up, make sure we only use slw_reinit() when HILE isn't supported.
- cpu: Unconditionally cleanup TLBs on P9 in opal_reinit_cpus()

  This can work around problems where Linux fails to properly cleanup part or all of the TLB on kexec.
- Fix scom addresses for power9 nx checkstop hmi handling.

  Scom addresses for NX status, DMA & ENGINE FIR and PBI FIR has changed for Power9. Fixup thoes while handling nx checkstop for Power9.

- Fix scom addresses for power9 core checkstop hmi handling.

  Scom addresses for CORE FIR (Fault Isolation Register) and Malfunction Alert Register has changed for Power9. Fixup those while handling core checkstop for Power9.

  Without this change HMI handler fails to check for correct reason for core checkstop on Power9.

- core/mem_region: check return value of add_region

  The only sensible thing to do if this fails is to abort() as we've likely just failed reserving reserved memory regions, and nothing good comes from that.

PHB4

- phb4: Do more retries on link training failures Currently we only retry once when we have a link training failure. This changes this to be 3 retries as 1 retry is not giving us enough reliablity.

  This will increase the boot time, especially on systems where we incorrectly detect a link presence when there really is nothing present. I'll post a followup patch to optimise our timings to help mitigate this later.

- phb4: Workaround phy lockup by doing full PHB reset on retry

  For PHB4 it's possible that the phy may end up in a bad state where it can no longer recieve data. This can manifest as the link not retraining. A simple PERST will not clear this. The PHB must be completely reset.

  This changes the retry state to CRESET to do this.

  This issue may also manifest itself as the link training in a degraded state (lower speed or narrower width). This patch doesn't attempt to fix that (will come later).

- pci: Add ability to trace timing

  PCI link training is responsible for a huge chunk of the skiboot boot time, so add the ability to trace it waiting in the main state machine.

- pci: Print resetting PHB notice at higher log level

  Currently during boot there a long delay while we wait for the PHBs to be reset and train. During this time, there is no output from skiboot and the last message doesn't give an indication of what's happening.

  This boosts the PHB reset message from info to notice so users can see what's happening during this long period of waiting.

- phb4: Only set one bit in nfir

  The MPIPL procedure says to only set bit 26 when forcing the PEC into freeze mode. Currently we set bits 24-27.

  This changes the code to follow spec and only set bit 26.

- phb4: Fix order of pfir/nfir clearing in CRESET

  According to the workbook, pfir must be cleared before the nfir. The way we have it now causes the nfir to not clear properly in some error circumstances.

  This swaps the order to match the workbook.

- phb4: Remove incorrect state transition

  When waiting in PHB4_SLOT_CRESET_WAIT_CQ for transations to end, we incorrectly move onto the next state. Generally we don't hit this as the transactions have ended already anyway.

  This removes the incorrect state transition.

- phb4: Set default lane equalisation

  Set default lane equalisation if there is nothing in the device-tree.

  Default value taken from hdat and confirmed by hardware team. Neatens the code up a bit too.

- hdata: Fix phb4 lane-eq property generation

  The lane-eq data we get from hdat is all 7s but what we end up in the device tree is:

```
xscom@603fc00000000/pbcq@4010c00/stack@0/ibm,lane-eq
                 00000000 31c339e0 00000000 0000000c
                 00000000 00000000 00000000 00000000
                 00000000 31c30000 77777777 77777777
                 77777777 77777777 77777777 77777777
```

  This fixes grabbing the properties from hdat and fixes the call to put them in the device tree.

- phb4: Fix PHB4 fence recovery.

  We had a few problems:

  - We used the wrong register to trigger the reset (spec bug)
  - We should clear the PFIR and NFIR while the reset is asserted
  - … and in the right order !
  - We should only apply the DD1 workaround after the reset has been lifted.
  - We should ensure we use ASB whenever we are fenced or doing a CRESET
  - Make config ops write with ASB

- phb4: Verbose EEH options

  Enabled via nvram pci-eeh-verbose=true. ie.

```
nvram -p ibm,skiboot --update-config pci-eeh-verbose=true
```

- phb4: Print more info when PHB fences

  For now at PHBERR level. We don't have room in the diags data passed to Linux for these unfortunately.

Testing/development

- lpc: remove double LPC prefix from messages
- opal-ci/fetch-debian-jessie-installer: follow redirects Fixes some CI failures
- test/qemu-jessie: bail out fast on kernel panic
- test/qemu-jessie: dump boot log on failure
- travis: add fedora26
- xz: add fallthrough annotations to silence GCC7 warning

5.1.85 skiboot-5.8

skiboot v5.8 was released on Thursday August 31st 2017. It is the first release of skiboot 5.8, which becomes the new stable release. It follows the 5.7 release, first released 25th July 2017.

skiboot v5.8 contains all bug fixes as of *skiboot-5.4.6* and *skiboot-5.1.20* (the currently maintained stable releases). We do not currently expect to do any 5.7.x stable releases.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

Over *skiboot-5.7*, we have the following changes:

New Features

- sensors: occ: Add support to clear sensor groups

  Adds a generic API to clear sensor groups. OCC inband sensor groups such as CSM, Profiler and Job Scheduler can be cleared using this API. It will clear the min/max of all sensors belonging to OCC sensor groups.

- sensors: occ: Add CSM_{min/max} sensors

  HWMON's lowest/highest attribute is used by CSM agent, so map min/max device-tree properties "sensor-data-min" and "sensor-data-max" to the min/max of CSM.

- sensors: occ: Add support for OCC inband sensors

  Add support to parse and export OCC inband sensors which are copied by OCC to main memory in P9. Each OCC writes three buffers which includes one names buffer for sensor meta data and two buffers for sensor readings. While OCC writes to one buffer the sensor values can be read from the other buffer. The sensors are updated every 100ms.

  This patch adds power, temperature, current and voltage sensors to `/ibm,opal/sensors` device-tree node which can be exported by the ibmpowernv-hwmon driver in Linux.

- psr: occ: Add support to change power-shifting-ratio

  Add support to set the CPU-GPU power shifting ratio which is used by the OCC power capping algorithm. PSR value of 100 takes all power away from CPU first and a PSR value of 0 caps GPU first.

- powercap: occ: Add a generic powercap framework

  This patch adds a generic powercap framework and exports OCC powercap sensors using which system powercap can be set inband through OPAL-OCC command-response interface.

- phb4: Enable PCI peer-to-peer

  P9 supports PCI peer-to-peer: a PCI device can write directly to the mmio space of another PCI device. It completely by-passes the CPU.

  It requires some configuration on the PHBs involved:

  1. on the initiating side, the address for the read/write operation is in the mmio space of the target, i.e. well outside the range normally allowed. So we disable range-checking on the TVT entry in bypass mode.
  2. on the target side, we need to explicitly enable p2p by setting a bit in a configuration register. It has the side-effect of reserving an outbound (as seen from the CPU) store queue for p2p. Therefore we only enable p2p on the PHBs using it, as we don't want to waste the resource if we don't have to.

  P9 supports p2p mmio writes. Reads are currently only supported if the two devices are under the same PHB but that is expected to change in the future, and it raises questions about intermediate switches configuration, so we report an error for the time being.

  The patch adds a new OPAL call to allow the OS to declare a p2p (initiator, target) pair.

- NX 842 and GZIP support on POWER9

POWER9 DD2

Further support for POWER9 DD2 revision chips. Notable changes include:

- xscom: Grab P9 DD2 revision level
- vas: Set mmio enable bits in DD2

  POWER9 DD2 added some new "enable" bits that must be set for VAS to work. These bits were unused in DD1.
- hdat: Add POWER9 DD2.0 specific pa_features

  Same as the default but with TM off.

POWER9

Since *skiboot-5.8-rc1*:

- hw/npu2.c: Add ibm,nvlink-speed device-tree property

  NVLink2 links can support multiple different speeds. However the device driver has no way of determining which speed was programmed so pass it down as a device tree property.
- hw/npu2-hw-procedures.c: Update PHY_RESET procedure

  Newer versions of Hostboot will have various clocks powered down by default to save power. Therefore we need to power them up before accessing the OBUS PHY.
- p8-i2c: Fix random data corruption (POWER9 specific) While waiting for the OCC to signal that it has finished using the I2C master we put the master into the, poorly named, occache_dis state. While in this state the transaction hasn't been started, but p8_i2c_check_status() will only skip it's checks when the master is in the idle state. Any action that checks that cranks the I2C state machine (interrupt, poll, etc) will call p8_i2c_check_status() and since the master is not idle, it will check the status register, see the transaction complete flag set and complete the i2c request without actually doing anything.

  If the transaction was a I2C read, the resulting output will be a zeroed data buffer.
- hw/p8-i2c: Fix OCC locking (POWER9 specific)

  There's a few issues with the Host<->OCC I2C bus handshaking. First up, skiboot is currently examining the wrong bit when checking if the OCC is currently using the bus. Secondly, when we need to wait for the OCC to release the bus we are scheduling a recovery timer to run zero timebase ticks after the current moment so the recovery timeout handler will run immediately after the bus was requested, which will in turn re-schedule itself, etc, etc. There's also a race between the OCC interrupt and the recovery handler which can result in an assertion failure in the recovery thread. All of this is bad.

  This patch addresses all these issues and sets the recovery timeout to 10ms.
- vas: export chip-id to vas platform device This is needed so VAS in the kernel can perform cpu to vas id mapping.
- slw: Modify the power9 stop0_lite latency & residency

  Currently skiboot exposes the exit-latency for stop0_lite as 200ns and the target-residency to be 2us.

  However, the kernel cpu-idle infrastructure rounds up the latency to microseconds and lists the stop0_lite latency as 0us, putting it on par with snooze state. As a result, when the predicted latency is small (< 1us), cpuidle will select stop0_lite instead of snooze. The difference between these states is that snooze doesn't require an interrupt to exit from the state, but stop0_lite does. And the value 200ns doesn't include the interrupt latency.

  This shows up in the context_switch2 benchmark (http://ozlabs.org/~anton/junkcode/context_switch2.c) where the number of context switches per second with the stop0_lite disabled is found to be roughly 30% more than with stop0_lite enabled. This can be correlated with the number of times cpuidle enters stop0_lite compared to snooze.

Hence, bump up the exit latency of stop0_lite to 1us. Since the target residency is chosen to be 10 times the exit latency, set the target residency to 10us.

With these values, we see a 50% improvement in the number of context switches.

Since *skiboot-5.7*:

- Base NPU2 support on POWER9 DD2
- hdata/i2c: Work around broken I2C array version

  Work around a bug in the I2C devices array that shows the array version as being v2 when only the v1 data is populated.

- Recognize the 2s2u zz platform

  OPAL currently doesn't know about the 2s2u zz. It recognizes such a box as a generic BMC machine and fails to boot. Add the 2s2u as a supported platform.

  There will subsequently be a 2s2u-L system which may have a different compatible property, which will need to be handled later.

- hdata/spira: POWER9 NX isn't software compatible with P7/P8 NX, don't claim so
- NX: Add P9 NX support for gzip compression engine

  Power 9 introduces NX gzip compression engine. This patch adds gzip compression support in NX. Virtual Accelerator Switch (VAS) is used to access NX gzip engine and the channel configuration will be done with the receive FIFO. So RxFIFO address, logical partition ID (lpid), process ID (pid) and thread ID (tid) are used to configure RxFIFO. P9 NX supports high and normal priority FIFOS. Skiboot configures User Mode Access Control (UMAC) noitify match register with these values and also enables other registers to enable / disable the engine.

  Creates the following device-tree entries to provide RxFIFO address, RxFIFO size, Fifo priority, lpid, pid and tid values so that kernel can drive P9 NX gzip engine.

  The following nodes are located under an xscom node: ::

  /xscom@<xscom_addr>/nx@<nx_addr>

  /ibm,gzip-high-fifo : High priority gzip RxFIFO /ibm,gzip-normal-fifo : Normal priority gzip RxFIFO

  Each RxFIFO node contain:s

  `compatible` `ibm,p9-nx-gzip`

  `priority` High or Normal

  `rx-fifo-address` RxFIFO address

  `rx-fifo-size` RxFIFO size

  `lpid` 0xfff (1's for 12 bits in UMAC notify match register)

  `pid` gzip coprocessor type

  `tid` counter for gzip

- NX: Add P9 NX support for 842 compression engine

  This patch adds changes needed for 842 compression engine on power 9. Virtual Accelerator Switch (VAS) is used to access NX 842 engine on P9 and the channel setup will be done with receive FIFO. So RxFIFO address, logical partition ID (lpid), process ID (pid) and thread ID (tid) are used for this setup. p9 NX supports high and normal priority FIFOs. skiboot is not involved to process data with 842 engine, but configures User Mode Access Control (UMAC) noitify match register with these values and export them to kernel with device-tree entries.

Also configure registers to setup and enable / disable the engine with the appropriate registers. Creates the following device-tree entries to provide RxFIFO address, RxFIFO size, Fifo priority, lpid, pid and tid values so that kernel can drive P9 NX 842 engine.

The following nodes are located under an xscom node: `/xscom@<xscom_addr>/nx@<nx_addr>`

`/ibm,842-high-fifo` High priority 842 RxFIFO

`/ibm,842-normal-fifo` Normal priority 842 RxFIFO

Each RxFIFO node contains:

`compatible` ibm,p9-nx-842

`priority` High or Normal

`rx-fifo-address` RxFIFO address

`rx-fifo-size` RXFIFO size

`lpid` 0xfff (1's for 12 bits set in UMAC notify match register)

`pid` 842 coprocessor type

`tid` Counter for 842

- vas: Create MMIO device tree node

  Create a device tree node for VAS and add properties that Linux will need to configure/use VAS.

- opal: Extract sw checkstop fir address from HDAT.

  Extract sw checkstop fir address info from HDAT and populate device tree node ibm,sw-checkstop-fir.

  This patch is required for OPAL_CEC_REBOOT2 OPAL call to work as expected on p9.

  With this patch a device property 'ibm,sw-checkstop-fir' is now properly populated:

```
# lsprop ibm,sw-checkstop-fir
ibm,sw-checkstop-fir
                 05012000 0000001f
```

PHB4

- hdat: Fix PCIe GEN4 lane-eq setting for DD2

  For PCIe GEN4, DD2 uses only 1 byte per PCIe lane for the lane-eq settings (DD1 uses 2 bytes)

- pci: Wait for CRS and switch link when restoring bus numbers

  When a complete reset occurs, after the PHB recovers it propagates a reset down the wire to every device. At the same time, skiboot talks to every device in order to restore the state of devices to what they were before the reset.

  In some situations, such as devices that recovered slowly and/or were behind a switch, skiboot attempted to access config space of the device before the link was up and the device could respond.

  Fix this by retrying CRS until the device responds correctly, and for devices behind a switch, making sure the switch has its link up first.

- pci: Track whether a PCI device is a virtual function

  This can be checked from config space, but we will need to know this when restoring the PCI topology, and it is not always safe to access config space during this period.

- phb4: Enhanced PCIe training tracing

  This add more details to the PCI training tracing (aka Rick Mata mode). It enables the PCIe Link Training and Status State Machine (LTSSM) tracing and details on speed and link width.

  Output now looks like this when enabled (via nvram):

```
[    1.096995141,3] PHB#0000[0:0]: TRACE:0x0000001101000000  0ms          ␣
↪GEN1:x16:detect
[    1.102849137,3] PHB#0000[0:0]: TRACE:0x0000102101000000 11ms presence␣
↪GEN1:x16:polling
[    1.104341838,3] PHB#0000[0:0]: TRACE:0x0000182101000000 14ms training␣
↪GEN1:x16:polling
[    1.104357444,3] PHB#0000[0:0]: TRACE:0x00001c5101000000 14ms training␣
↪GEN1:x16:recovery
[    1.104580394,3] PHB#0000[0:0]: TRACE:0x00001c5103000000 14ms training␣
↪GEN3:x16:recovery
[    1.123259359,3] PHB#0000[0:0]: TRACE:0x00001c5104000000 51ms training␣
↪GEN4:x16:recovery
[    1.141737656,3] PHB#0000[0:0]: TRACE:0x0000144104000000 87ms presence␣
↪GEN4:x16:L0
[    1.141752318,3] PHB#0000[0:0]: TRACE:0x0000154904000000 87ms trained ␣
↪GEN4:x16:L0
[    1.141757964,3] PHB#0000[0:0]: TRACE: Link trained.
[    1.096834019,3] PHB#0001[0:1]: TRACE:0x0000001101000000  0ms          ␣
↪GEN1:x16:detect
[    1.105578525,3] PHB#0001[0:1]: TRACE:0x0000102101000000 17ms presence␣
↪GEN1:x16:polling
[    1.112763075,3] PHB#0001[0:1]: TRACE:0x0000183101000000 31ms training␣
↪GEN1:x16:config
[    1.112778956,3] PHB#0001[0:1]: TRACE:0x00001c5081000000 31ms training␣
↪GEN1:x08:recovery
[    1.113002083,3] PHB#0001[0:1]: TRACE:0x00001c5083000000 31ms training␣
↪GEN3:x08:recovery
[    1.114833873,3] PHB#0001[0:1]: TRACE:0x0000144083000000 35ms presence␣
↪GEN3:x08:L0
[    1.114848832,3] PHB#0001[0:1]: TRACE:0x0000154883000000 35ms trained ␣
↪GEN3:x08:L0
[    1.114854650,3] PHB#0001[0:1]: TRACE: Link trained.
```

- phb4: Fix reading wrong size registers in EEH dump

  These registers are supposed to be 16bit, and it makes part of the register dump misleading.

- phb4: Ignore slot state if performing complete reset

  If a PHB is being completely reset, its state is about to be blown away anyway, so if it's not in an appropriate state, creset it regardless.

- phb4: Prepare for link down when creset called from kernel

  phb4_creset() is typically called by functions that prepare the link to go down. In cases where creset() is called directly by the kernel, this isn't the case and it can cause issues. Prepare for link down in creset, just like we do in freset and hreset.

- phb4: Skip attempting to fix PHBs broken on boot

If a PHB is marked broken it didn't work on boot, and if it didn't work on boot then there's no point trying to recover it later

- phb4: Fix duplicate in EEH register dump
- phb4: Be more conservative on link presence timeout

  In this patch we tuned our link timing to be more agressive: `cf960e2884 phb4:  Improve reset and link training timing`

  Cards should take only 32ms but unfortunately we've seen some take up to 440ms. Hence bump our timer up to 1000ms.

  This can hurt boot times on systems where slots indicate a hotplug status but no electrical link is present (which we've seen). Since we have to wait 1 second between PERST and touching config space anyway, it shouldn't hurt too much.
- phb4: Assert PERST before PHB reset

  Currently we don't assert PERST before issuing a PHB reset. This means any link issues while resetting the PHB will be logged as errors.

  This asserts PERST before we start resetting the PHB to avoid this.
- Revert "phb4: Read PERST signal rather than assuming it's asserted"

  This reverts commit b42ff2b904165addf32e77679cebb94a08086966

  The original patch assumes that PERST has been asserted well before (> 250ms) we hit here (ie. during hostboot).

  In a subesquent patch this will no longer be the case as we need to assert PERST during PHB reset, which may only be a few milliseconds before we hit this code.

  Hence revert this patch. Go back to the software mechanism using skip_perst to determine if PERST should be asserted or not. This allows us to keep the speed optimisation on boot.
- phb4: Set REGB error enables based on link state

  Currently we always set these enables when initing the PHB. If the link is already down, we shouldn't set them as it may cause spurious errors.

  This changes the code to only sets them if the link is up.
- phb4: Mark PHB as fenced on creset

  If we have to inject an error to trigger recover, we end up not marking the PHB as fenced in the PHB struct. This fixes that.
- phb4: Clear errors before deasserting reset

  During reset we may have logged some errors (eg. due to the link going down).

  Hence before we deassert PERST or Hot Reset, we need to clear these errors. This ensures that once link training starts, only new errors are logged.
- phb4: Disable device config space access when fenced

  On DD2 you can't access device config space when fenced, so just disable access whenever we are fenced.
- phb4: Dump devctl and devstat registers

  Dump devctl and devstat registers. These would have been useful when debugging the MPS issue.
- phb4: Only clear some PHB config space registers on errors

Currently on error we clear the entire PHB config space. This is a problem as the PCIe Maximum Payload Size (MPS) negotiation may have already occurred. Clearing MPS in the PHB back to a default of 128 bytes will result an error for a device which already has a larger MPS configured.

This will manifest itself as error due to a malformed TLP packet. ie. `phbPblErrorStatus bit 41 = "Malformed TLP error"`

This has been seen after kexec on with some adapters.

This fixes the problem by only clearing a subset of registers on a phb error.

Utilities

- external/xscom-utils: Add `--list-bits`

  When using getscom/putscom it's helpful to know what bits are set in the register. This patch adds an option to print out which bits are set along with the value that was read/written to the register. Note that this output indicates which bits are set using the IBM bit ordering since that's what the XSCOM documentation uses.

opal-prd

- opal-prd: Do not pass pnor file while starting daemon.

  This change to the included systemd init file means opal-prd can start and run on IBM FSP based systems.

  We do not have pnor support on all the system. Also we have logic to autodetect PNOR. Hence do not pass `--pnor` by default.

- opal-prd: Disable pnor access interface on FSP system

  On FSP system host does not have access to PNOR. Hence disable PNOR access interfaces.

OPAL Sensors

- sensor-groups : occ: Add 'ops' DT property

  Add new device-tree property 'ops' to define different operations supported on each sensor-group.

- OCC: Map OCC sensor to a chip-id

  Parse device tree to get chip-id for OCC sensor.

- HDAT: Add chip-id property to ipmi sensors

  Presently we do not have a way to map sensor to chip id. Hence we are always passing chip id 0 for occ_reset request (see occ_sensor_id_to_chip()).

  This patch adds chip-id property to sensors (whenever its available) so that we can map occ sensor to chip-id and pass valid chip-id to occ_reset request.

- xive: Check for valid PIR index when decoding

  This fixes an unlikely but possible assert() fail on kdump.

- sensors: occ: Skip the deconfigured core sensors

  This patch skips the deconfigured cores from the core sensors while parsing the sensor names in the main memory as these sensor values are not updated by OCC.

IBM FSP systems

Since *skiboot-5.8-rc1*:

- mktime: fix off-by-one error calling days_in_month

  From auditing all the mktime() users, there seems to be only a *very* small window around new years day where we could possibly return incorrect data to the OS, and even then, there would have to be FSP reset/reload on FSP machines. I don't *think* there's an opportunity on other machines.

Tests

Since *skiboot-5.8-rc1*:

- travis: Debian Stretch must pass
- test kernels: link with -N
- core/test/run-msg: don't depend on unittest mem layout

Since *skiboot-5.7*:

- hdata_to_dt: use a realistic PVR and chip revision
- nx: PR_INFO that NX RNG and Crypto not yet supported on POWER9
- external/pflash: Add tests
- external/pflash: Reinstate the progress bars

  Recent work did some optimising which unfortunately removed some of the progress bars in pflash.

  It turns out that there's only one thing people prefer to correctly programmed flash chips, it is the ability to watch little equals characters go across their screens for potentially minutes.

- external/pflash: Correct erase alignment checks

  pflash should check the alignment of addresses and sizes when asked to erase. There are two possibilities:

  1. The user has specified sizes manually in which case pflash should be as flexible as possible, blocklevel_smart_erase() permits this. To prevent possible mistakes pflash will require –force to perform a manual erase of unaligned sizes.
  2. The user used -P to specify a partition, partitions aren't necessarily erase granule aligned anymore, blocklevel_smart_erase() can handle. In this it doesn't make sense to warn/error about misalignment since the misalignment is inherent to the FFS partition and not really user input.

- external/pflash: Check the result of strtoul

  Also add 0x in front of –info output to avoid a copy and paste mistake.

- libflash/file: Break up MTD erase ioctl() calls

  Unfortunately not all drivers are created equal and several drivers on which pflash relies block in the kernel for quite some time and ignore signals.

  This is really only a problem if pflash is to perform large erases. So don't, perform these ops in small chunks.

  An in kernel fix is possible in most cases but it takes time and systems will be running older drivers for quite some time. Since sector erases aren't significantly slower than whole chip erases there isn't much of a performance penalty to breaking up the erase ioctl()s.

General

Since *skiboot-5.8-rc1*:

- gcov: support GCC 7.1+
- Tests build and pass on Debian A few things related to the Debian toolchain.

Since *skiboot-5.7*:

- opal-msg: Increase the max-async completion count by max chips possible
- occ: Add support for OPAL-OCC command/response interface

  This patch adds support for a shared memory based command/response interface between OCC and OPAL. In HOMER, there is an OPAL command buffer and an OCC response buffer which is used to send inband commands to OCC.
- HDAT/device-tree: only add lid-type on pre-POWER9 systems

  Largely a relic of back when we had multiple entry points into OPAL depending on which mechanism on an FSP we were using to get loaded, this isn't needed on modern P9 as we only have one entry point (we don't do the PHYP LID hack).

Contributors

- Processed 156 csets from 17 developers
- 1 employers found
- A total of 6888 lines added, 1089 removed (delta 5799)

Developers with the most changesets

Developer	#	%
Cyril Bur	35	(22.4%)
Stewart Smith	32	(20.5%)
Michael Neuling	23	(14.7%)
Sukadev Bhattiprolu	11	(7.1%)
Reza Arbab	10	(6.4%)
Russell Currey	9	(5.8%)
Shilpasri G Bhat	9	(5.8%)
Oliver O'Halloran	5	(3.2%)
Haren Myneni	5	(3.2%)
Alistair Popple	4	(2.6%)
Vasant Hegde	4	(2.6%)
Nicholas Piggin	3	(1.9%)
Andrew Donnellan	2	(1.3%)
Gautham R. Shenoy	1	(0.6%)
Mahesh Salgaonkar	1	(0.6%)
Ananth N Mavinakayanahalli	1	(0.6%)
Frederic Barrat	1	(0.6%)

Developers with the most changed lines

Developer	#	%
Shilpasri G Bhat	1935	(27.9%)
Cyril Bur	1868	(26.9%)
Stewart Smith	866	(12.5%)
Sukadev Bhattiprolu	663	(9.5%)
Haren Myneni	584	(8.4%)
Michael Neuling	384	(5.5%)
Frederic Barrat	168	(2.4%)
Reza Arbab	98	(1.4%)
Oliver O'Halloran	98	(1.4%)
Vasant Hegde	93	(1.3%)
Alistair Popple	77	(1.1%)
Russell Currey	60	(0.9%)
Mahesh Salgaonkar	28	(0.4%)
Andrew Donnellan	11	(0.2%)
Gautham R. Shenoy	6	(0.1%)
Nicholas Piggin	4	(0.1%)
Ananth N Mavinakayanahalli	1	(0.0%)

Developers with the most signoffs

Developer	#	%
Stewart Smith	124	(97.6%)
Benjamin Herrenschmidt	2	(1.6%)
Vaidyanathan Srinivasan	1	(0.8%)
Total	127	(100%)

Developers with the most reviews

Developer	#	%
Samuel Mendoza-Jonas	19	(52.8%)
Andrew Donnellan	11	(30.6%)
Vasant Hegde	2	(5.6%)
Cédric Le Goater	1	(2.8%)
Russell Currey	1	(2.8%)
Reza Arbab	1	(2.8%)
Cyril Bur	1	(2.8%)
Total	36	(100%)

Developers with the most test credits

Developer	#	%
Vasant Hegde	1	(50.0%)
Hari Bathini	1	(50.0%)

Developers who gave the most tested-by credits

Developer	#	%
Russell Currey	1	(50.0%)
Mahesh Salgaonkar	1	(50.0%)

Developers with the most report credits

Developer	#	%
Anton Blanchard	1	(16.7%)
Mark Linimon	1	(16.7%)
Pavaman Subramaniyam	1	(16.7%)
Pridhiviraj Paidipeddi	1	(16.7%)
Rob Lippert	1	(16.7%)
Michael Neuling	1	(16.7%)

Developers who gave the most report credits

Developer	#	%
Stewart Smith	2	(33.3%)
Michael Neuling	1	(16.7%)
Andrew Donnellan	1	(16.7%)
Cyril Bur	1	(16.7%)
Gautham R. Shenoy	1	(16.7%)

5.1.86 skiboot-5.8-rc1

skiboot v5.8-rc1 was released on Tuesday August 22nd 2017. It is the first release candidate of skiboot 5.8, which will become the new stable release of skiboot following the 5.7 release, first released 25th July 2017.

skiboot v5.8-rc1 contains all bug fixes as of *skiboot-5.4.6* and *skiboot-5.1.20* (the currently maintained stable releases). We do not currently expect to do any 5.7.x stable releases.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.8 by August 25th, with skiboot 5.8 being for all POWER8 and POWER9 platforms in op-build v1.19 (Due August 25th). This is a short cycle as this release is mainly targetted towards POWER9 bringup efforts.

Over skiboot-5.7, we have the following changes:

New Features

- sensors: occ: Add support to clear sensor groups

  Adds a generic API to clear sensor groups. OCC inband sensor groups such as CSM, Profiler and Job Scheduler can be cleared using this API. It will clear the min/max of all sensors belonging to OCC sensor groups.

- sensors: occ: Add CSM_{min/max} sensors

  HWMON's lowest/highest attribute is used by CSM agent, so map min/max device-tree properties "sensor-data-min" and "sensor-data-max" to the min/max of CSM.

- sensors: occ: Add support for OCC inband sensors

  Add support to parse and export OCC inband sensors which are copied by OCC to main memory in P9. Each OCC writes three buffers which includes one names buffer for sensor meta data and two buffers for sensor readings. While OCC writes to one buffer the sensor values can be read from the other buffer. The sensors are updated every 100ms.

  This patch adds power, temperature, current and voltage sensors to `/ibm,opal/sensors` device-tree node which can be exported by the ibmpowernv-hwmon driver in Linux.

- psr: occ: Add support to change power-shifting-ratio

  Add support to set the CPU-GPU power shifting ratio which is used by the OCC power capping algorithm. PSR value of 100 takes all power away from CPU first and a PSR value of 0 caps GPU first.

- powercap: occ: Add a generic powercap framework

  This patch adds a generic powercap framework and exports OCC powercap sensors using which system powercap can be set inband through OPAL-OCC command-response interface.

- phb4: Enable PCI peer-to-peer

  P9 supports PCI peer-to-peer: a PCI device can write directly to the mmio space of another PCI device. It completely by-passes the CPU.

  It requires some configuration on the PHBs involved:

  1. on the initiating side, the address for the read/write operation is in the mmio space of the target, i.e. well outside the range normally allowed. So we disable range-checking on the TVT entry in bypass mode.
  2. on the target side, we need to explicitly enable p2p by setting a bit in a configuration register. It has the side-effect of reserving an outbound (as seen from the CPU) store queue for p2p. Therefore we only enable p2p on the PHBs using it, as we don't want to waste the resource if we don't have to.

  P9 supports p2p mmio writes. Reads are currently only supported if the two devices are under the same PHB but that is expected to change in the future, and it raises questions about intermediate switches configuration, so we report an error for the time being.

  The patch adds a new OPAL call to allow the OS to declare a p2p (initiator, target) pair.

- NX 842 and GZIP support on POWER9

POWER9 DD2

Further support for POWER9 DD2 revision chips. Notable changes include:

- xscom: Grab P9 DD2 revision level
- vas: Set mmio enable bits in DD2

  POWER9 DD2 added some new "enable" bits that must be set for VAS to work. These bits were unused in DD1.

- hdat: Add POWER9 DD2.0 specific pa_features

  Same as the default but with TM off.

POWER9

- Base NPU2 support on POWER9 DD2
- hdata/i2c: Work around broken I2C array version

  Work around a bug in the I2C devices array that shows the array version as being v2 when only the v1 data is populated.
- Recognize the 2s2u zz platform

  OPAL currently doesn't know about the 2s2u zz. It recognizes such a box as a generic BMC machine and fails to boot. Add the 2s2u as a supported platform.

  There will subsequently be a 2s2u-L system which may have a different compatible property, which will need to be handled later.
- hdata/spira: POWER9 NX isn't software compatible with P7/P8 NX, don't claim so
- NX: Add P9 NX support for gzip compression engine

  Power 9 introduces NX gzip compression engine. This patch adds gzip compression support in NX. Virtual Accelerator Switch (VAS) is used to access NX gzip engine and the channel configuration will be done with the receive FIFO. So RxFIFO address, logical partition ID (lpid), process ID (pid) and thread ID (tid) are used to configure RxFIFO. P9 NX supports high and normal priority FIFOS. Skiboot configures User Mode Access Control (UMAC) noitify match register with these values and also enables other registers to enable / disable the engine.

  Creates the following device-tree entries to provide RxFIFO address, RxFIFO size, Fifo priority, lpid, pid and tid values so that kernel can drive P9 NX gzip engine.

  The following nodes are located under an xscom node: ::

  /xscom@<xscom_addr>/nx@<nx_addr>

  /ibm,gzip-high-fifo : High priority gzip RxFIFO /ibm,gzip-normal-fifo : Normal priority gzip RxFIFO

  Each RxFIFO node contain:s

  `compatible` `ibm,p9-nx-gzip`

  `priority` High or Normal

  `rx-fifo-address` RxFIFO address

  `rx-fifo-size` RxFIFO size

  `lpid` 0xfff (1's for 12 bits in UMAC notify match register)

  `pid` gzip coprocessor type

  `tid` counter for gzip
- NX: Add P9 NX support for 842 compression engine

  This patch adds changes needed for 842 compression engine on power 9. Virtual Accelerator Switch (VAS) is used to access NX 842 engine on P9 and the channel setup will be done with receive FIFO. So RxFIFO address, logical partition ID (lpid), process ID (pid) and thread ID (tid) are used for this setup. p9 NX supports high and normal priority FIFOs. skiboot is not involved to process data with 842 engine, but configures User Mode Access Control (UMAC) noitify match register with these values and export them to kernel with device-tree entries.

Also configure registers to setup and enable / disable the engine with the appropriate registers. Creates the following device-tree entries to provide RxFIFO address, RxFIFO size, Fifo priority, lpid, pid and tid values so that kernel can drive P9 NX 842 engine.

The following nodes are located under an xscom node: `/xscom@<xscom_addr>/nx@<nx_addr>`

`/ibm,842-high-fifo` High priority 842 RxFIFO

`/ibm,842-normal-fifo` Normal priority 842 RxFIFO

Each RxFIFO node contains:

`compatible` ibm,p9-nx-842

`priority` High or Normal

`rx-fifo-address` RxFIFO address

`rx-fifo-size` RXFIFO size

`lpid` 0xfff (1's for 12 bits set in UMAC notify match register)

`pid` 842 coprocessor type

`tid` Counter for 842

- vas: Create MMIO device tree node

  Create a device tree node for VAS and add properties that Linux will need to configure/use VAS.

- opal: Extract sw checkstop fir address from HDAT.

  Extract sw checkstop fir address info from HDAT and populate device tree node ibm,sw-checkstop-fir.

  This patch is required for OPAL_CEC_REBOOT2 OPAL call to work as expected on p9.

  With this patch a device property 'ibm,sw-checkstop-fir' is now properly populated:

```
# lsprop ibm,sw-checkstop-fir
ibm,sw-checkstop-fir
                 05012000 0000001f
```

PHB4

- hdat: Fix PCIe GEN4 lane-eq setting for DD2

  For PCIe GEN4, DD2 uses only 1 byte per PCIe lane for the lane-eq settings (DD1 uses 2 bytes)

- pci: Wait for CRS and switch link when restoring bus numbers

  When a complete reset occurs, after the PHB recovers it propagates a reset down the wire to every device. At the same time, skiboot talks to every device in order to restore the state of devices to what they were before the reset.

  In some situations, such as devices that recovered slowly and/or were behind a switch, skiboot attempted to access config space of the device before the link was up and the device could respond.

  Fix this by retrying CRS until the device responds correctly, and for devices behind a switch, making sure the switch has its link up first.

- pci: Track whether a PCI device is a virtual function

  This can be checked from config space, but we will need to know this when restoring the PCI topology, and it is not always safe to access config space during this period.

- phb4: Enhanced PCIe training tracing

  This add more details to the PCI training tracing (aka Rick Mata mode). It enables the PCIe Link Training and Status State Machine (LTSSM) tracing and details on speed and link width.

  Output now looks like this when enabled (via nvram):

```
[    1.096995141,3] PHB#0000[0:0]: TRACE:0x0000001101000000  0ms          ␣
↪GEN1:x16:detect
[    1.102849137,3] PHB#0000[0:0]: TRACE:0x0000102101000000 11ms presence␣
↪GEN1:x16:polling
[    1.104341838,3] PHB#0000[0:0]: TRACE:0x0000182101000000 14ms training␣
↪GEN1:x16:polling
[    1.104357444,3] PHB#0000[0:0]: TRACE:0x00001c5101000000 14ms training␣
↪GEN1:x16:recovery
[    1.104580394,3] PHB#0000[0:0]: TRACE:0x00001c5103000000 14ms training␣
↪GEN3:x16:recovery
[    1.123259359,3] PHB#0000[0:0]: TRACE:0x00001c5104000000 51ms training␣
↪GEN4:x16:recovery
[    1.141737656,3] PHB#0000[0:0]: TRACE:0x0000144104000000 87ms presence␣
↪GEN4:x16:L0
[    1.141752318,3] PHB#0000[0:0]: TRACE:0x0000154904000000 87ms trained ␣
↪GEN4:x16:L0
[    1.141757964,3] PHB#0000[0:0]: TRACE: Link trained.
[    1.096834019,3] PHB#0001[0:1]: TRACE:0x0000001101000000  0ms          ␣
↪GEN1:x16:detect
[    1.105578525,3] PHB#0001[0:1]: TRACE:0x0000102101000000 17ms presence␣
↪GEN1:x16:polling
[    1.112763075,3] PHB#0001[0:1]: TRACE:0x0000183101000000 31ms training␣
↪GEN1:x16:config
[    1.112778956,3] PHB#0001[0:1]: TRACE:0x00001c5081000000 31ms training␣
↪GEN1:x08:recovery
[    1.113002083,3] PHB#0001[0:1]: TRACE:0x00001c5083000000 31ms training␣
↪GEN3:x08:recovery
[    1.114833873,3] PHB#0001[0:1]: TRACE:0x0000144083000000 35ms presence␣
↪GEN3:x08:L0
[    1.114848832,3] PHB#0001[0:1]: TRACE:0x0000154883000000 35ms trained ␣
↪GEN3:x08:L0
[    1.114854650,3] PHB#0001[0:1]: TRACE: Link trained.
```

- phb4: Fix reading wrong size registers in EEH dump

  These registers are supposed to be 16bit, and it makes part of the register dump misleading.

- phb4: Ignore slot state if performing complete reset

  If a PHB is being completely reset, its state is about to be blown away anyway, so if it's not in an appropriate state, creset it regardless.

- phb4: Prepare for link down when creset called from kernel

  phb4_creset() is typically called by functions that prepare the link to go down. In cases where creset() is called directly by the kernel, this isn't the case and it can cause issues. Prepare for link down in creset, just like we do in freset and hreset.

- phb4: Skip attempting to fix PHBs broken on boot

  If a PHB is marked broken it didn't work on boot, and if it didn't work on boot then there's no point trying to recover it later

- phb4: Fix duplicate in EEH register dump

- phb4: Be more conservative on link presence timeout

  In this patch we tuned our link timing to be more agressive: `cf960e2884 phb4:  Improve reset and link training timing`

  Cards should take only 32ms but unfortunately we've seen some take up to 440ms. Hence bump our timer up to 1000ms.

  This can hurt boot times on systems where slots indicate a hotplug status but no electrical link is present (which we've seen). Since we have to wait 1 second between PERST and touching config space anyway, it shouldn't hurt too much.

- phb4: Assert PERST before PHB reset

  Currently we don't assert PERST before issuing a PHB reset. This means any link issues while resetting the PHB will be logged as errors.

  This asserts PERST before we start resetting the PHB to avoid this.

- Revert "phb4: Read PERST signal rather than assuming it's asserted"

  This reverts commit b42ff2b904165addf32e77679cebb94a08086966

  The original patch assumes that PERST has been asserted well before (> 250ms) we hit here (ie. during hostboot).

  In a subesquent patch this will no longer be the case as we need to assert PERST during PHB reset, which may only be a few milliseconds before we hit this code.

  Hence revert this patch. Go back to the software mechanism using skip_perst to determine if PERST should be asserted or not. This allows us to keep the speed optimisation on boot.

- phb4: Set REGB error enables based on link state

  Currently we always set these enables when initing the PHB. If the link is already down, we shouldn't set them as it may cause spurious errors.

  This changes the code to only sets them if the link is up.

- phb4: Mark PHB as fenced on creset

  If we have to inject an error to trigger recover, we end up not marking the PHB as fenced in the PHB struct. This fixes that.

- phb4: Clear errors before deasserting reset

  During reset we may have logged some errors (eg. due to the link going down).

  Hence before we deassert PERST or Hot Reset, we need to clear these errors. This ensures that once link training starts, only new errors are logged.

- phb4: Disable device config space access when fenced

  On DD2 you can't access device config space when fenced, so just disable access whenever we are fenced.

- phb4: Dump devctl and devstat registers

  Dump devctl and devstat registers. These would have been useful when debugging the MPS issue.

- phb4: Only clear some PHB config space registers on errors

  Currently on error we clear the entire PHB config space. This is a problem as the PCIe Maximum Payload Size (MPS) negotiation may have already occurred. Clearing MPS in the PHB back to a default of 128 bytes will result an error for a device which already has a larger MPS configured.

  This will manifest itself as error due to a malformed TLP packet. ie. `phbPblErrorStatus bit 41 = "Malformed TLP error"`

This has been seen after kexec on with some adapters.

This fixes the problem by only clearing a subset of registers on a phb error.

Utilities

- external/xscom-utils: Add `--list-bits`

  When using getscom/putscom it's helpful to know what bits are set in the register. This patch adds an option to print out which bits are set along with the value that was read/written to the register. Note that this output indicates which bits are set using the IBM bit ordering since that's what the XSCOM documentation uses.

opal-prd

- opal-prd: Do not pass pnor file while starting daemon.

  This change to the included systemd init file means opal-prd can start and run on IBM FSP based systems.

  We do not have pnor support on all the system. Also we have logic to autodetect PNOR. Hence do not pass `--pnor` by default.

- opal-prd: Disable pnor access interface on FSP system

  On FSP system host does not have access to PNOR. Hence disable PNOR access interfaces.

OPAL Sensors

- sensor-groups : occ: Add 'ops' DT property

  Add new device-tree property 'ops' to define different operations supported on each sensor-group.

- OCC: Map OCC sensor to a chip-id

  Parse device tree to get chip-id for OCC sensor.

- HDAT: Add chip-id property to ipmi sensors

  Presently we do not have a way to map sensor to chip id. Hence we are always passing chip id 0 for occ_reset request (see occ_sensor_id_to_chip()).

  This patch adds chip-id property to sensors (whenever its available) so that we can map occ sensor to chip-id and pass valid chip-id to occ_reset request.

- xive: Check for valid PIR index when decoding

  This fixes an unlikely but possible assert() fail on kdump.

- sensors: occ: Skip the deconfigured core sensors

  This patch skips the deconfigured cores from the core sensors while parsing the sensor names in the main memory as these sensor values are not updated by OCC.

Tests

- hdata_to_dt: use a realistic PVR and chip revision
- nx: PR_INFO that NX RNG and Crypto not yet supported on POWER9
- external/pflash: Add tests

- external/pflash: Reinstate the progress bars

  Recent work did some optimising which unfortunately removed some of the progress bars in pflash.

  It turns out that there's only one thing people prefer to correctly programmed flash chips, it is the ability to watch little equals characters go across their screens for potentially minutes.

- external/pflash: Correct erase alignment checks

  pflash should check the alignment of addresses and sizes when asked to erase. There are two possibilities:

  1. The user has specified sizes manually in which case pflash should be as flexible as possible, block-level_smart_erase() permits this. To prevent possible mistakes pflash will require –force to perform a manual erase of unaligned sizes.
  2. The user used -P to specify a partition, partitions aren't necessarily erase granule aligned anymore, block-level_smart_erase() can handle. In this it doesn't make sense to warn/error about misalignment since the misalignment is inherent to the FFS partition and not really user input.

- external/pflash: Check the result of strtoul

  Also add 0x in front of –info output to avoid a copy and paste mistake.

- libflash/file: Break up MTD erase ioctl() calls

  Unfortunately not all drivers are created equal and several drivers on which pflash relies block in the kernel for quite some time and ignore signals.

  This is really only a problem if pflash is to perform large erases. So don't, perform these ops in small chunks.

  An in kernel fix is possible in most cases but it takes time and systems will be running older drivers for quite some time. Since sector erases aren't significantly slower than whole chip erases there isn't much of a performance penalty to breaking up the erase ioctl()s.

General

- opal-msg: Increase the max-async completion count by max chips possible
- occ: Add support for OPAL-OCC command/response interface

  This patch adds support for a shared memory based command/response interface between OCC and OPAL. In HOMER, there is an OPAL command buffer and an OCC response buffer which is used to send inband commands to OCC.

- HDAT/device-tree: only add lid-type on pre-POWER9 systems

  Largely a relic of back when we had multiple entry points into OPAL depending on which mechanism on an FSP we were using to get loaded, this isn't needed on modern P9 as we only have one entry point (we don't do the PHYP LID hack).

5.1.87 skiboot-5.9

skiboot v5.9 was released on Tuesday October 31st 2017. It is the first release of skiboot 5.9 and becomes the new stable release of skiboot following the 5.8 release, first released August 31st 2017. In this cyle we have had five release candidate releases, mostly centered around bug fixing for POWER9 platforms.

This release should be considered suitable for early-access POWER9 systems.

skiboot v5.9 contains all bug fixes as of *skiboot-5.4.8* and *skiboot-5.1.21* (the currently maintained stable releases). There may be some 5.9.x stable releases, depending on what issues are found.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

Over *skiboot-5.8*, we have the following changes:

New Features

POWER8

- fast-reset by default (if possible)

  Currently, this is limited to POWER8 systems.

  A normal reboot will, rather than doing a full IPL, go through a fast reboot procedure. This reduces the "reboot to petitboot" time from minutes to a handful of seconds.

POWER9

Since *skiboot-5.9-rc3*:

- occ-sensors : Add OCC inband sensor region to exports (useful for debugging)

Two SRESET fixes (see below for feature description):

- core: direct-controls: Fix clearing of special wakeup

  'special_wakeup_count' is incremented on successfully asserting special wakeup. So we will never clear the special wakeup if we check 'special_wakeup_count' to be zero. Fix this issue by checking the 'special_wakeup_count' to 1 in dctl_clear_special_wakeup().

- core/direct-controls: increase special wakeup timeout on POWER9

  Some instances have been observed where the special wakeup assert times out. The current timeout is too short for deeper sleep states. Hostboot uses 100ms, so match that.

Since *skiboot-5.9-rc2*: - cpu: Add OPAL_REINIT_CPUS_TM_SUSPEND_DISABLED

> Add a new CPU reinit flag, "TM Suspend Disabled", which requests that CPUs be configured so that TM (Transactional Memory) suspend mode is disabled.
>
> Currently this always fails, because skiboot has no way to query the state. A future hostboot change will add a mechanism for skiboot to determine the status and return an appropriate error code.

Since *skiboot-5.8*:

- POWER9 power management during boot

  Less power should be consumed during boot.

- OPAL_SIGNAL_SYSTEM_RESET for POWER9

  This implements OPAL_SIGNAL_SYSTEM_RESET, using scom registers to quiesce the target thread and raise a system reset exception on it. It has been tested on DD2 with stop0 ESL=0 and ESL=1 shallow power saving modes.

  DD1 is not implemented because it is sufficiently different as to make support difficult.

- Enable deep idle states for POWER9
  - SLW: Add support for p9_stop_api

    p9_stop_api's are used to set SPR state on a core wakeup form a deeper low power state. p9_stop_api uses low level platform formware and self-restore microcode to restore the sprs to requested values.

Code is taken from : https://github.com/open-power/hostboot/tree/master/src/import/chips/p9/procedures/utils/stopreg

- SLW: Removing timebase related flags for stop4

  When a core enters stop4, it does not loose decrementer and time base. Hence removing flags OPAL_PM_DEC_STOP and OPAL_PM_TIMEBASE_STOP.

- SLW: Allow deep states if homer address is known

  Use a common variable has_wakeup_engine instead of has_slw to tell if the: - SLW image is populated in case of power8 - CME image is populated in case of power9

  Currently we expect CME to be loaded if homer address is known (except for simulators)

- SLW: Configure self-restore for HRMOR

  Make a stop api call using libpore to restore HRMOR register. HRMOR needs to be cleared so that when thread exits stop, they arrives at linux system_reset vector (0x100).

- SLW: Add opal_slw_set_reg support for power9

  This OPAL call is made from Linux to OPAL to configure values in various SPRs after wakeup from a deep idle state.

- PHB4: CAPP recovery

  CAPP recovery is initiated when a CAPP Machine Check is detected. The capp recovery procedure is initiated via a Hypervisor Maintenance interrupt (HMI).

  CAPP Machine Check may arise from either an error that results in a PHB freeze or from an internal CAPP error with CAPP checkstop FIR action. An error that causes a PHB freeze will result in the link down signal being asserted. The system continues running and the CAPP and PSL will be re-initialized.

  This implements CAPP recovery for POWER9 systems

- Add `wafer-location` property for POWER9

  Extract wafer-location from ECID and add property under xscom node. - bits 64:71 are the chip x location (7:0) - bits 72:79 are the chip y location (7:0)

  Sample output:

```
[root@wsp xscom@623fc00000000]# lsprop ecid
ecid             019a00d4 03100718 852c0000 00fd7911
[root@wsp xscom@623fc00000000]# lsprop wafer-location
wafer-location   00000085 0000002c
```

- Add `wafer-id` property for POWER9

  Wafer id is derived from ECID data. - bits 4:63 are the wafer id (ten 6 bit fields each containing a code)

  Sample output:

```
[root@wsp xscom@623fc00000000]# lsprop ecid
ecid             019a00d4 03100718 852c0000 00fd7911
[root@wsp xscom@623fc00000000]# lsprop wafer-id
wafer-id         "6Q0DG340SO"
```

- Add `ecid` property under `xscom` node for POWER9. Sample output:

```
[root@wsp xscom@623fc00000000]# lsprop ecid
ecid             019a00d4 03100718 852c0000 00fd7911
```

- Add ibm,firmware-versions device tree node

  In P8, hostboot provides mini device tree. It contains `/ibm,firmware-versions` node which has various firmware component version details.

  In P9, OPAL is building device tree. This patch adds support to parse VERSION section of PNOR and create `/ibm,firmware-versions` device tree node.

  Sample output:

  ```
  /sys/firmware/devicetree/base/ibm,firmware-versions # lsprop .
  occ                "6a00709"
  skiboot            "v5.7-rc1-p344fb62"
  buildroot          "2017.02.2-7-g23118ce"
  capp-ucode         "9c73e9f"
  petitboot          "v1.4.3-p98b6d83"
  sbe                "02021c6"
  open-power         "witherspoon-v1.17-128-gf1b53c7-dirty"
  ....
  ....
  ```

POWER9

Since *skiboot-5.9-rc5*:

- Suppress XSCOM chiplet-offline errors on P9

  Workaround on P9: PRD does operations it *knows* will fail with this error to work around a hardware issue where accesses via the PIB (FSI or OCC) work as expected, accesses via the ADU (what xscom goes through) do not. The chip logic will always return all FFs if there is any error on the scom.

- asm/head: initialize preferred DSCR value

  POWER7/8 use DSCR=0. POWER9 preferred value has “stride-N” enabled.

Since *skiboot-5.9-rc4*: - opal/hmi: Workaround Power9 hw logic bug for couple of TFMR TB errors. - opal/hmi: Fix TB reside and HDEC parity error recovery for power9

Since *skiboot-5.9-rc2*: - hw/imc: Fix IMC Catalog load for DD2.X processors

Since *skiboot-5.9-rc1*: - xive: Fix VP free block group mode false-positive parameter check

  The check to ensure the buddy allocation idx is aligned to its allocation order was not taking into account the allocation split. This would result in opal_xive_free_vp_block failures despite giving the same value as returned by opal_xive_alloc_vp_block.

  E.g., starting then stopping 4 KVM guests gives the following pattern in the host:

  ```
  opal_xive_alloc_vp_block(5)=0x45000020
  opal_xive_alloc_vp_block(5)=0x45000040
  opal_xive_alloc_vp_block(5)=0x45000060
  opal_xive_alloc_vp_block(5)=0x45000080
  opal_xive_free_vp_block(0x45000020)=-1
  opal_xive_free_vp_block(0x45000040)=0
  opal_xive_free_vp_block(0x45000060)=-1
  opal_xive_free_vp_block(0x45000080)=0
  ```

- hw/imc: pause microcode at boot

  IMC nest counters has both in-band (ucode access) and out of band access to it. Since not all nest counter configurations are supported by ucode, out of band tools are used to characterize other configuration.

So it is prefer to pause the nest microcode at boot to aid the nest out of band tools. If the ucode not paused and OS does not have IMC driver support, then out to band tools will race with ucode and end up getting undesirable values. Patch to check and pause the ucode at boot.

OPAL provides APIs to control IMC counters. OPAL_IMC_COUNTERS_INIT is used to initialize these counters at boot. OPAL_IMC_COUNTERS_START and OPAL_IMC_COUNTERS_STOP API calls should be used to start and pause these IMC engines. *doc/opal-api/opal-imc-counters.rst* details the OPAL APIs and their usage.

- hdata/i2c: update the list of known i2c devs

  This updates the list of known i2c devices - as of HDAT spec v10.5e - so that they can be properly identified during the hdat parsing.

- hdata/i2c: log unknown i2c devices

  An i2c device is unknown if either the i2c device list is outdated or the device is marked as unknown (0xFF) in the hdat.

Since *skiboot-5.8*:

- Disable Transactional Memory on Power9 DD 2.1

  Update pa_features_p9[] to disable TM (Transactional Memory). On DD 2.1 TM is not usable by Linux without other workarounds, so skiboot must disable it.

- xscom: Do not print error message for 'chiplet offline' return values

  xscom_read/write operations returns CHIPLET_OFFLINE when chiplet is offline. Some multicast xscom_read/write requests from HBRT results in xscom operation on offline chiplet(s) and printing below warnings in OPAL console:

```
[ 135.036327572,3] XSCOM: Read failed, ret = -14
[ 135.092689829,3] XSCOM: Read failed, ret = -14
```

  Some SCOM users can deal correctly with this error code (notably opal-prd), so the error message is (in practice) erroneous.

- IMC: Fix the core_imc_event_mask

  CORE_IMC_EVENT_MASK is a scom that contains bits to control event sampling for different machine state for core imc. The current event-mask setting sample events only on host kernel (hypervisor) and host userspace.

  Patch to enable the sampling of events in other machine states (like guest kernel and guest userspace).

- IMC: Update the nest_pmus array with occ/gpe microcode uav updates

  OOC/gpe nest microcode maintains the list of individual nest units supported. Sync the recent updates to the UAV with nest_pmus array.

  For reference occ/gpr microcode link for the UAV: https://github.com/open-power/occ/blob/master/src/occ_gpe1/gpe1_24x7.h

- Parse IOSLOT information from HDAT

  Add structure definitions that describe the physical PCIe topology of a system and parse them into the device-tree based PCIe slot description.

- idle: user context state loss flags fix for stop states

  The "lite" stop variants with PSSCR[ESL]=PSSCR[EC]=1 do not lose user context, while the non-lite variants do (ESL: enable state loss).

  Some of the POWER9 idle states had these wrong.

CAPI

- POWER9 DD2 update

  The CAPI initialization sequence has been updated in DD2. This patch adapts to the changes, retaining compatibility with DD1. The patch includes some changes to DD1 fix-ups as well.

- Load CAPP microcode for POWER9 DD2.0 and DD2.1
- capi: Mask Psl Credit timeout error for POWER9

  Mask the PSL credit timeout error in CAPP FIR Mask register bit(46). As per the h/w team this error is now deprecated and shouldn't cause any fir-action for P9.

NVLINK2

A notabale change is that we now generate the device tree description of NVLINK based on the HDAT we get from hostboot. Since Hostboot will generate HDAT based on VPD, you now *MUST* have correct VPD programmed or we will *default* to a Sequoia layout, which will lead to random problems if you are not booting a Sequoia Witherspoon planar. In the case of booting with old VPD and/or Hostboot, we print a **giant scary warning** in order to scare you.

Since *skiboot-5.9-rc2*: - Revert "npu2: Add vendor cap for IRQ testing"

> This reverts commit 9817c9e29b6fe00daa3a0e4420e69a97c90eb373 which seems to break setting the PCI dev flag and the link number in the PCIe vendor specific config space. This leads to the device driver attempting to re-init the DL when it shouldn't which can cause HMI's.

Since *skiboot-5.8*:

- npu2: Read slot label from the HDAT link node

  Binding GPU to emulated NPU PCI devices is done using the slot labels since the NPU devices do not have a patching slot node we need to copy the label in here.

- npu2: Copy link speed from the npu HDAT node

  This needs to be in the PCI device node so the speed of the NVLink can be passed to the GPU driver.

- npu2: hw-procedures: Add settings to PHY_RESET

  Set a few new values in the PHY_RESET procedure, as specified by our updated programming guide documentation.

- Parse NVLink information from HDAT

  Add the per-chip structures that descibe how the A-Bus/NVLink/OpenCAPI phy is configured. This generates the npu@xyz nodes for each chip on systems that support it.

- npu2: Add vendor cap for IRQ testing

  Provide a way to test recoverable data link interrupts via a new vendor capability byte.

- npu2: Enable recoverable data link (no-stall) interrupts

  Allow the NPU2 to trigger "recoverable data link" interrupts.

- npu2: Implement basic FLR (Function Level Reset)
- npu2: hw-procedures: Update PHY DC calibration procedure
- npu2: hw-procedures: Change rx_pr_phase_step value

XIVE

- xive: Fix opal_xive_dump_tm() to access W2 properly. The HW only supported limited access sizes.
- xive: Make opal_xive_allocate_irq() properly try all chips

  When requested via OPAL_XIVE_ANY_CHIP, we need to try all chips. We first try the current one (on which the caller sits) and if that fails, we iterate all chips until the allocation succeeds.
- xive: Fix initialization & cleanup of HW thread contexts

  Instead of trying to "pull" everything and clear VT (which didn't work and caused some FIRs to be set), instead just clear and then set the PTER thread enable bit. This has the side effect of completely resetting the corresponding thread context.

  This fixes the spurrious XIVE FIRs reported by PRD and fircheck
- xive: Add debug option for detecting misrouted IPI in emulation

  This is high overhead so we don't enable it by default even in debug builds, it's also a bit messy, but it allowed me to detect and debug a locking issue earlier so it can be useful.
- xive: Increase the interrupt "gap" on debug builds

  We normally allocate IPIs from 0x10. Make that 0x1000 on debug builds to limit the chances of overlapping with Linux interrupt numbers which makes debugging code that confuses them easier.

  Also add a warning in emulation if we get an interrupt in the queue whose number is below the gap.
- xive: Fix locking around cache scrub & watch

  Thankfully the missing locking only affects debug code and init code that doesn't run concurrently. Also adds a DEBUG option that checks the lock is properly held.
- xive: Workaround HW issue with scrub facility

  Without this, we sometimes don't observe from a CPU the values written to the ENDs or NVTs via the cache watch.
- xive: Add exerciser for cache watch/scrub facility in DEBUG builds
- xive: Make assertion in xive_eq_for_target() more informative
- xive: Add debug code to check initial cache updates
- xive: Ensure pressure relief interrupts are disabled

  We don't use them and we hijack the VP field with their configuration to store the EQ reference, so make sure the kernel or guest can't turn them back on by doing MMIO writes to ACK#
- xive: Don't try setting the reserved ACK# field in VPs

  That doesn't work, the HW doesn't implement it in the cache watch facility anyway.
- xive: Remove useless memory barriers in VP/EQ inits

  We no longer update "live" memory structures, we use a temporary copy on the stack and update the actual memory structure using the cache watch, so those barriers are pointless.

PHB4

Since *skiboot-5.9-rc4*:

- phb4: Escalate freeze to fence to avoid checkstop

  Freeze events such as MMIO loads can cause the PHB to lose it's limited powerbus credits. If all credits are used and a further MMIO will cause a checkstop.

  To work around this, we escalate the troublesome freeze events to a fence. The fence will cause a full PHB reset which resets the powerbus credits and avoids the checkstop.

- phb4: Update some init registers

  New inits based on next PHB4 workbook. Increases some timeouts to avoid some spurious error conditions.

- phb4: Enable PHB MMIO in phb4_root_port_init()

  Linux EEH flow is somewhat broken. It saves the PCIe config space of the PHB on boot, which it then uses to restore on EEH recovery. It does this to restore MMIO bars and some other pieces.

  Unfortunately this save is done before any drivers are bound to devices under the PHB. A number of other things are configured in the PHB after drivers start, hence some configuration space settings aren't saved correctly. These include bus master and MMIO bits in the command register.

  Linux tried to hack around this in this linux commit `bf898ec5cb` powerpc/eeh: Enable PCI_COMMAND_MASTER for PCI bridges This sets the bus master bit but ignores the MMIO bit.

  Hence we lose MMIO after a full PHB reset. This causes the next MMIO access to the device to fail and for us to perform a PE freeze recovery, which still doesn't set the MMIO bit and hence we still fail.

  This works around this by forcing MMIO on during phb4_root_port_init().

  With this we can recovery from a PHB fence event on POWER9.

- phb4: Reduce link degraded message log level to debug

  If we hit this message we'll retry and fix the problem. If we run out of retries and can't fix the problem, we'll still print a log message at error level indicating a problem.

- phb4: Fix GEN3 for DD2.00

  In this fix: `62ac7631ae phb4:  Fix PCIe GEN4 on DD2.1 and above` We fixed DD2.1 GEN4 but broke DD2.00 as GEN3.

  This fixes DD2.00 back to GEN3. This time for sure!

Since *skiboot-5.9-rc3*: - phb4: Fix PCIe GEN4 on DD2.1 and above

  In this change: eef0e197ab PHB4: Default to PCIe GEN3 on POWER9 DD2.00

  We clamped DD2.00 parts to GEN3 but unfortunately this change also applies to DD2.1 and above.

  This fixes this to only apply to DD2.00.

Since *skiboot-5.8*:

- phb4: Mask RXE_ARB: DEC Stage Valid Error

  Change the inits to mask out the RXE ARB: DEC Stage Valid Error (bit 370. This has been a fatal error but should be informational only.

  This update will be in the next version of the phb4 workbook.

- phb4: Add additional adapter to retrain whitelist

  The single port version of the ConnectX-5 has a different device ID 0x1017. Updated descriptions to match pciutils database.

- PHB4: Default to PCIe GEN3 on POWER9 DD2.00

  You can use the NVRAM override for DD2.00 screened parts.

- phb4: Retrain link if degraded

  On P9 Scale Out (Nimbus) DD2.0 and Scale in (Cumulus) DD1.0 (and below) the PCIe PHY can lockup causing training issues. This can cause a degradation in speed or width in ~5% of training cases (depending on the card). This is fixed in later chip revisions. This issue can also cause PCIe links to not train at all, but this case is already handled.

  This patch checks if the PCIe link has trained optimally and if not, does a full PHB reset (to fix the PHY lockup) and retrain.

  One complication is some devices are known to train degraded unless device specific configuration is performed. Because of this, we only retrain when the device is in a whitelist. All devices in the current whitelist have been testing on a P9DSU/Boston, ZZ and Witherspoon.

  We always gather information on the link and print it in the logs even if the card is not in the whitelist.

  For testing purposes, there's an nvram to retry all PCIe cards and all P9 chips when a degraded link is detected. The new option is 'pci-retry-all=true' which can be set using: *nvram -p ibm,skiboot –update-config pci-retry-all=true*. This option may increase the boot time if used on a badly behaving card.

IBM FSP platforms

Since *skiboot-5.9-rc5*: - FSP/CONSOLE: Disable notification on unresponsive consoles

> Commit fd6b71fc fixed the situation where ipmi console was open (hvc0) but got data on different console (hvc1).
>
> During FSP Reset/Reload OPAL closes all consoles. After Reset/Reload complete FSP requests to open hvc1 and sends data on this. If hvc1 registration failed or not opened in host kernel then it will not read data and results in RCU stalls.
>
> Note that this is workaround for older kernel where we don't have separate irq for each console. Latest kernel works fine without this patch.

Since *skiboot-5.9-rc1*:

- FSP/CONSOLE: Limit number of error logging

  Commit c8a7535f (FSP/CONSOLE: Workaround for unresponsive ipmi daemon) added error logging when buffer is full. In some corner cases kernel may call this function multiple time and we may endup logging error again and again.

  This patch fixes it by generating error log only once.

- FSP/CONSOLE: Fix fsp_console_write_buffer_space() call

  Kernel calls fsp_console_write_buffer_space() to check console buffer space availability. If there is enough buffer space to write data, then kernel will call fsp_console_write() to write actual data.

  In some extreme corner cases (like one explained in commit c8a7535f) console becomes full and this function returns 0 to kernel (or space available in console buffer < next incoming data size). Kernel will continue retrying until it gets enough space. So we will start seeing RCU stalls.

  This patch keeps track of previous available space. If previous space is same as current means not enough space in console buffer to write incoming data. It may be due to very high console write operation and slow response from FSP -OR- FSP has stopped processing data (ex: because of ipmi daemon died). At this point we will start timer with timeout of SER_BUFFER_OUT_TIMEOUT (10 secs). If situation is not improved within 10 seconds means something went bad. Lets return OPAL_RESOURCE so that kernel can drop console write and continue.

- FSP/CONSOLE: Close SOL session during R/R

  Presently we are not closing SOL and FW console sessions during R/R. Host will continue to write to SOL buffer during FSP R/R. If there is heavy console write operation happening during FSP R/R (like running *top* command inside console), then at some point console buffer becomes full. fsp_console_write_buffer_space() returns 0 (or less than required space to write data) to host. While one thread is busy writing to console, if some other threads tries to write data to console we may see RCU stalls (like below) in kernel.

```
[ 2082.828363] INFO: rcu_sched detected stalls on CPUs/tasks: { 32} (detected by␣
↪16, t=6002 jiffies, g=23154, c=23153, q=254769)
[ 2082.828365] Task dump for CPU 32:
[ 2082.828368] kworker/32:3    R  running task        0  4637      2 0x00000884
[ 2082.828375] Workqueue: events dump_work_fn
[ 2082.828376] Call Trace:
[ 2082.828382] [c000000f1633fa00] [c00000000013b6b0] console_unlock+0x570/0x600␣
↪(unreliable)
[ 2082.828384] [c000000f1633fae0] [c00000000013ba34] vprintk_emit+0x2f4/0x5c0
[ 2082.828389] [c000000f1633fb60] [c00000000099e644] printk+0x84/0x98
[ 2082.828391] [c000000f1633fb90] [c0000000000851a8] dump_work_fn+0x238/0x250
[ 2082.828394] [c000000f1633fc60] [c0000000000ecb98] process_one_work+0x198/0x4b0
[ 2082.828396] [c000000f1633fcf0] [c0000000000ed3dc] worker_thread+0x18c/0x5a0
[ 2082.828399] [c000000f1633fd80] [c0000000000f4650] kthread+0x110/0x130
[ 2082.828403] [c000000f1633fe30] [c000000000009674] ret_from_kernel_thread+0x5c/
↪0x68
```

  Hence lets close SOL (and FW console) during FSP R/R.

- FSP/CONSOLE: Do not associate unavailable console

  Presently OPAL sends associate/unassociate MBOX command for all FSP serial console (like below OPAL message). We have to check console is available or not before sending this message.

```
[ 5013.227994012,7] FSP: Reassociating HVSI console 1
[ 5013.227997540,7] FSP: Reassociating HVSI console 2
```

- FSP: Disable PSI link whenever FSP tells OPAL about impending R/R

  Commit 42d5d047 fixed scenario where DPO has been initiated, but FSP went into reset before the CEC power down came in. But this is generic issue that can happen in normal shutdown path as well.

  Hence disable PSI link as soon as we detect FSP impending R/R.

- fsp: return OPAL_BUSY_EVENT on failure sending FSP_CMD_POWERDOWN_NORM Also, return OPAL_BUSY_EVENT on failure sending FSP_CMD_REBOOT / DEEP_REBOOT.

  We had a race condition between FSP Reset/Reload and powering down the system from the host:

  Roughly:

#	FSP	Host
1	Power on	
2		Power on
3	(inject EPOW)	
4	(trigger FSP R/R)	
5		Processes EPOW event, starts shutting down
6		calls OPAL_CEC_POWER_DOWN
7	(is still in R/R)	
8		gets OPAL_INTERNAL_ERROR, spins in opal_poll_events
9	(FSP comes back)	
10		spinning in opal_poll_events
11	(thinks host is running)	

The call to OPAL_CEC_POWER_DOWN is only made once as the reset/reload error path for fsp_sync_msg() is to return -1, which means we give the OS OPAL_INTERNAL_ERROR, which is fine, except that our own API docs give us the opportunity to return OPAL_BUSY when trying again later may be successful, and we're ambiguous as to if you should retry on OPAL_INTERNAL_ERROR.

For reference, the linux code looks like this:

```
static void __noreturn pnv_power_off(void)
{
        long rc = OPAL_BUSY;

        pnv_prepare_going_down();

        while (rc == OPAL_BUSY || rc == OPAL_BUSY_EVENT) {
                rc = opal_cec_power_down(0);
                if (rc == OPAL_BUSY_EVENT)
                        opal_poll_events(NULL);
                else
                        mdelay(10);
        }
        for (;;)
                opal_poll_events(NULL);
}
```

Which means that *practically* our only option is to return OPAL_BUSY or OPAL_BUSY_EVENT.

We choose OPAL_BUSY_EVENT for FSP systems as we do want to ensure we're running pollers to communicate with the FSP and do the final bits of Reset/Reload handling before we power off the system.

Since *skiboot-5.8*:

- FSP/NVRAM: Handle "get vNVRAM statistics" command

  FSP sends MBOX command (cmd : 0xEB, subcmd : 0x05, mod : 0x00) to get vNVRAM statistics. OPAL doesn't maintain any such statistics. Hence return FSP_STATUS_INVALID_SUBCMD.

  Fixes these messages appearing in the OPAL log:

```
[16944.384670488,3] FSP: Unhandled message eb0500
[16944.474110465,3] FSP: Unhandled message eb0500
[16945.111280784,3] FSP: Unhandled message eb0500
[16945.293393485,3] FSP: Unhandled message eb0500
```

- fsp: Move common prints to trace

These two prints just end up filling the skiboot logs on any machine that's been booted for more than a few hours.

They have never been useful, so make them trace level. They were: :: SURV: Received heartbeat acknowledge from FSP SURV: Sending the heartbeat command to FSP

BMC based systems

- hw/lpc-uart: read from RBR to clear character timeout interrupts

  When using the aspeed SUART, we see a condition where the UART sends continuous character timeout interrupts. This change adds a (heavily commented) dummy read from the RBR to clear the interrupt condition on init.

  This was observed on p9dsu systems, but likely applies to other systems using the SUART.

- astbmc: Add methods for handing Device Tree based slots e.g. ones from HDAT on POWER9.

General

Since *skiboot-5.9-rc5*:

- p8-i2c: Further timeout reworks

  This patch reworks the way timeouts are set so that rather than imposing a hard deadline based on the transaction length it uses a kick-the-can-down-the-road approach where the timeout will be reset each time data is written to or received from the master. This fits better with the actual failure modes that timeouts are designed to handle, such as unusually slow or broken devices.

  Additionally this patch moves all the special case detection out of the timeout handler. This is help to improve the robustness of the driver and prepare for a more substantial rework of the driver as a whole later on.

- npu: Fix broken fast reset

  0679f61244b "fast-reset: by default (if possible)" broke NPU - now the NV links does not get enabled after reboot.

  This disables fast reboot for NPU machines till a better solution is found.

Since *skiboot-5.9-rc2*:

- Improvements to vpd device tree entries

  Previously we would miss some properties

Since *skiboot-5.9-rc1*:

- hw/p8-i2c: Fix deadlock in p9_i2c_bus_owner_change

  When debugging a system where Linux was taking soft lockup errors with two CPUs stuck in OPAL:

CPU0	CPU1
lock	
p8_i2c_recover	
opal_handle_interrupt	sync_timer cancel_timer p9_i2c_bus_owner_change occ_p9_interrupt xive_source_interrupt opal_handle_interrupt

  p8_i2c_recover() is a timer, and is stuck trying to take master->lock. p9_i2c_bus_owner_change() has taken master->lock, but then is stuck waiting for all timers to complete. We deadlock.

  Fix this by using cancel_timer_async().

- opal/cpu: Mark the core as bad while disabling threads of the core.

  If any of the core fails to sync its TB during chipTOD initialization, all the threads of that core are disabled. But this does not make linux kernel to ignore the core/cpus. It crashes while bringing them up with below backtrace:

```
[   38.883898] kexec_core: Starting new kernel
cpu 0x0: Vector: 300 (Data Access) at [c0000003f277b730]
    pc: c0000000001b9890: internal_create_group+0x30/0x304
    lr: c0000000001b9880: internal_create_group+0x20/0x304
    sp: c0000003f277b9b0
   msr: 900000000280b033
   dar: 40
 dsisr: 40000000
  current = 0xc0000003f9f41000
  paca    = 0xc00000000fe00000   softe: 0        irq_happened: 0x01
    pid   = 2572, comm = kexec
Linux version 4.13.2-openpower1 (jenkins@p89) (gcc version 6.4.0 (Buildroot 2017.
↪08-00006-g319c6e1)) #1 SMP Wed Sep 20 05:42:11 UTC 2017
enter ? for help
[c0000003f277b9b0] c0000000008a8780 (unreliable)
[c0000003f277ba50] c00000000041c3ac topology_add_dev+0x2c/0x40
[c0000003f277ba70] c00000000006b078 cpuhp_invoke_callback+0x88/0x170
[c0000003f277bac0] c00000000006b22c cpuhp_up_callbacks+0x54/0xb8
[c0000003f277bb10] c00000000006bc68 cpu_up+0x11c/0x168
[c0000003f277bbc0] c00000000002f0e0 default_machine_kexec+0x1fc/0x274
[c0000003f277bc50] c00000000002e2d8 machine_kexec+0x50/0x58
[c0000003f277bc70] c0000000000de4e8 kernel_kexec+0x98/0xb4
[c0000003f277bce0] c00000000008b0f0 SyS_reboot+0x1c8/0x1f4
[c0000003f277be30] c00000000000b118 system_call+0x58/0x6c
```

Since *skiboot-5.8*:

- ipmi: Convert common debug prints to trace

  OPAL logs messages for every IPMI request from host. Sometime OPAL console is filled with only these messages. This path is pretty stable now and we have enough logs to cover bad path. Hence lets convert these debug message to trace/info message. Examples are:

```
[ 1356.423958816,7] opal_ipmi_recv(cmd: 0xf0 netfn: 0x3b resp_size: 0x02)
[ 1356.430774496,7] opal_ipmi_send(cmd: 0xf0 netfn: 0x3a len: 0x3b)
[ 1356.430797392,7] BT: seq 0x20 netfn 0x3a cmd 0xf0: Message sent to host
[ 1356.431668496,7] BT: seq 0x20 netfn 0x3a cmd 0xf0: IPMI MSG done
```

- libflash/file: Handle short read()s and write()s correctly

  Currently we don't move the buffer along for a short read() or write() and nor do we request only the remaining amount.

- hw/p8-i2c: Rework timeout handling

  Currently we treat a timeout as a hard failure and will automatically fail any transations that hit their timeout. This results in unnecessarily failing I2C requests if interrupts are dropped, etc. Although these are bad things that we should log we can handle them better by checking the actual hardware status and completing the transation if there are no real errors. This patch reworks the timeout handling to check the status and continue the transaction if it can. if it can while logging an error if it detects a timeout due to a dropped interrupt.

- core/flash: Only expect ELF header for BOOTKERNEL partition flash resource

  When loading a flash resource which isn't signed (secure and trusted boot) and which doesn't have a subpartition, we assume it's the BOOTKERNEL since previously this was the only such resource. Thus we also assumed it had an ELF header which we parsed to get the size of the partition rather than trusting the actual_size field in the FFS header. A previous commit (9727fe3 DT: Add ibm,firmware-versions node) added the version resource which isn't signed and also doesn't have a subpartition, thus we expect it to have an ELF header. It doesn't so we print the error message "FLASH: Invalid ELF header part VERSION".

It is a fluke that this works currently since we load the secure boot header unconditionally and this happen to be the same size as the version partition. We also don't update the return code on error so happen to return OPAL_SUCCESS.

To make this explicitly correct; only check for an ELF header if we are loading the BOOTKERNEL resource, otherwise use the partition size from the FFS header. Also set the return code on error so we don't erroneously return OPAL_SUCCESS. Add a check that the resource will fit in the supplied buffer to prevent buffer overrun.

- flash: Support adding the no-erase property to flash

  The mbox protocol explicitly states that an erase is not required before a write. This means that issuing an erase from userspace, through the mtd device, and back returns a successful operation that does nothing. Unfortunately, this makes userspace tools unhappy. Linux MTD devices support the MTD_NO_ERASE flag which conveys that writes do not require erases on the underlying flash devices. We should set this property on all of our devices which do not require erases to be performed.

  NOTE: This still requires a linux kernel component to set the MTD_NO_ERASE flag from the device tree property.

Utilities

Since *skiboot-5.9-rc1*: - opal-prd: Fix memory leak

Since *skiboot-5.8*:

- external/gard: Clear entire guard partition instead of entry by entry

  When using the current implementation of the gard tool to ecc clear the entire GUARD partition it is done one gard record at a time. While this may be ok when accessing the actual flash this is very slow when done from the host over the mbox protocol (on the order of 4 minutes) because the bmc side is required to do many read, erase, writes under the hood.

  Fix this by rewriting the gard tool reset_partition() function. Now we allocate all the erased guard entries and (if required) apply ecc to the entire buffer. Then we can do one big erase and write of the entire partition. This reduces the time to clear the guard partition to on the order of 4 seconds.

- opal-prd: Fix opal-prd command line options

  HBRT OCC reset interface depends on service processor type.

  - FSP: reset_pm_complex()
  - BMC: process_occ_reset()

  We have both *occ* and *pm-complex* command line interfaces. This patch adds support to dispaly appropriate message depending on system type.

SP	Command	Action
FSP	opal-prd occ	display error message
FSP	opal-prd pm-complex	Call pm_complex_reset()
BMC	opal-prd occ	Call process_occ_reset()
BMC	opal-prd pm-complex	display error message

- opal-prd: detect service processor type and then make appropriate occ reset call.

- pflash: Fix erase command for unaligned start address

  The erase_range() function handles erasing the flash for a given start address and length, and can handle an unaligned start address and length. However in the unaligned start address case we are incorrectly calculating the remaining size which can lead to incomplete erases.

  If we're going to update the remaining size based on what the start address was then we probably want to do that before we overide the origin start address. So rearrange the code so that this is indeed the case.

- external/gard: Print an error if run on an FSP system

Simulators

- mambo: Add mambo socket program

  This adds a program that can be run inside a mambo simulator in linux userspace which enables TCP sockets to be proxied in and out of the simulator to the host.

  Unlike mambo bogusnet, it's requires no linux or skiboot specific drivers/infrastructure to run.

  Run inside the simulator:

  - to forward host ssh connections to sim ssh server: `./mambo-socket-proxy -h 10022 -s 22`, then connect to port 10022 on your host with `ssh -p 10022 localhost`
  - to allow http proxy access from inside the sim to local http proxy: `./mambo-socket-proxy -b proxy.mynetwork -h 3128 -s 3128`

  Multiple connections are supported.

- idle: disable stop*_lite POWER9 idle states for Mambo platform

  Mambo prior to Mambo.7.8.21 had a bug where the stop idle instruction with PSSCR[ESL]=PSSCR[EC]=0 would resume with MSR set as though it had taken a system reset interrupt.

  Linux currently executes this instruction with MSR already set that way, so the problem went unnoticed. A proposed patch to Linux changes that, and causes the idle code to crash. Work around this by disabling lite stop states for the mambo platform for now.

Contributors

- 209 csets from 32 developers
- 2 employers found
- A total of 9619 lines added, 1612 removed (delta 8007)

Extending the analysis done for some previous releases, we can see our trends in code review across versions:

Release	csets	Ack %	Reviews %	Tested %	Reported %
5.0	329	15 (5%)	20 (6%)	1 (0%)	0 (0%)
5.1	372	13 (3%)	38 (10%)	1 (0%)	4 (1%)
5.2-rc1	334	20 (6%)	34 (10%)	6 (2%)	11 (3%)
5.3-rc1	302	36 (12%)	53 (18%)	4 (1%)	5 (2%)
5.4	361	16 (4%)	28 (8%)	1 (0%)	9 (2%)
5.5	408	11 (3%)	48 (12%)	14 (3%)	10 (2%)
5.6	87	12 (14%)	6 (7%)	5 (6%)	2 (2%)
5.7	232	30 (13%)	32 (14%)	5 (2%)	2 (1%)
5.8	157	13 (8%)	36 (23%)	2 (1%)	6 (4%)
5.9	209	15 (7%)	78 (37%)	3 (1%)	10 (5%)

The review count here is largely bogus, there was a series of 25 whitespace patches that got "Reviewed-by" and if we exclude them, we're back to 14%, which is more like what I'd expect.

Developers with the most changesets

Developer	#	%
Stewart Smith	28	(13.4%)
Vasant Hegde	25	(12.0%)
Joel Stanley	25	(12.0%)
Michael Neuling	24	(11.5%)
Oliver O'Halloran	20	(9.6%)
Benjamin Herrenschmidt	16	(7.7%)
Nicholas Piggin	12	(5.7%)
Akshay Adiga	8	(3.8%)
Madhavan Srinivasan	7	(3.3%)
Reza Arbab	6	(2.9%)
Mahesh Salgaonkar	3	(1.4%)
Claudio Carvalho	3	(1.4%)
Suraj Jitindar Singh	3	(1.4%)
Sam Bobroff	3	(1.4%)
Shilpasri G Bhat	2	(1.0%)
Michael Ellerman	2	(1.0%)
Andrew Donnellan	2	(1.0%)
Vaibhav Jain	2	(1.0%)
Jeremy Kerr	2	(1.0%)
Cyril Bur	2	(1.0%)
Christophe Lombard	2	(1.0%)
Daniel Black	2	(1.0%)
Alexey Kardashevskiy	1	(0.5%)
Alistair Popple	1	(0.5%)
Anton Blanchard	1	(0.5%)
Guilherme G. Piccoli	1	(0.5%)
John W Walthour	1	(0.5%)
Anju T Sudhakar	1	(0.5%)
Balbir Singh	1	(0.5%)
Russell Currey	1	(0.5%)
William A. Kennington III	1	(0.5%)
Sukadev Bhattiprolu	1	(0.5%)

Developers with the most changed lines

Developer	#	%
Akshay Adiga	2731	(27.9%)
Oliver O'Halloran	1512	(15.5%)
Stewart Smith	1355	(13.9%)
Nicholas Piggin	929	(9.5%)
Vasant Hegde	827	(8.5%)
Michael Neuling	719	(7.4%)
Benjamin Herrenschmidt	522	(5.3%)

Continued on next page

Table 12 – continued from previous page

Developer	#	%
Madhavan Srinivasan	180	(1.8%)
Sam Bobroff	172	(1.8%)
Christophe Lombard	170	(1.7%)
Mahesh Salgaonkar	166	(1.7%)
Andrew Donnellan	125	(1.3%)
Joel Stanley	70	(0.7%)
Reza Arbab	64	(0.7%)
Claudio Carvalho	51	(0.5%)
Suraj Jitindar Singh	42	(0.4%)
Alistair Popple	28	(0.3%)
Jeremy Kerr	25	(0.3%)
Michael Ellerman	21	(0.2%)
Cyril Bur	18	(0.2%)
Shilpasri G Bhat	17	(0.2%)
Vaibhav Jain	8	(0.1%)
Daniel Black	6	(0.1%)
William A. Kennington III	4	(0.0%)
Sukadev Bhattiprolu	4	(0.0%)
Alexey Kardashevskiy	3	(0.0%)
John W Walthour	3	(0.0%)
Balbir Singh	3	(0.0%)
Guilherme G. Piccoli	2	(0.0%)
Anton Blanchard	1	(0.0%)
Anju T Sudhakar	1	(0.0%)
Russell Currey	1	(0.0%)

Developers with the most lines removed

Developer	#	%
Alistair Popple	28	(1.7%)

Developers with the most signoffs

Developer	#	%
Stewart Smith	180	(97.8%)
Shilpasri G Bhat	2	(1.1%)
Mukesh Ojha	1	(0.5%)
Michael Neuling	1	(0.5%)
Total	184	(100%)

Developers with the most reviews

Developer	#	%
Michael Neuling	25	(32.5%)
Russell Currey	25	(32.5%)
Vaidyanathan Srinivasan	9	(11.7%)
Oliver O'Halloran	4	(5.2%)
Andrew Donnellan	3	(3.9%)
Frederic Barrat	2	(2.6%)
Suraj Jitindar Singh	2	(2.6%)
Vasant Hegde	2	(2.6%)
Andrew Jeffery	1	(1.3%)
Samuel Mendoza-Jonas	1	(1.3%)
Alexey Kardashevskiy	1	(1.3%)
Cyril Bur	1	(1.3%)
Akshay Adiga	1	(1.3%)
Total	77	(100%)

Developers with the most test credits

Developer	#	%
Pridhiviraj Paidipeddi	3	(100.0%)

Developers who gave the most tested-by credits

Developer	#	%
Vasant Hegde	2	(66.7%)
Michael Neuling	1	(33.3%)

Developers with the most report credits

Developer	#	%
Pridhiviraj Paidipeddi	6	(60.0%)
Andrew Donnellan	1	(10.0%)
Stewart Smith	1	(10.0%)
Shriya	1	(10.0%)
Robert Lippert	1	(10.0%)
Total	10	(100%)

Developers who gave the most report credits

Developer	#	%
Stewart Smith	3	(30.0%)
Suraj Jitindar Singh	3	(30.0%)
Vasant Hegde	2	(20.0%)
Michael Neuling	1	(10.0%)
Madhavan Srinivasan	1	(10.0%)
Total	10	(100%)

Changesets and Employers

Top changeset contributors by employer:

Employer	#	%
IBM	208	(99.5%)
Google	1	(0.5%)

Top lines changed by employer:

Employer	#	%
IBM	9776	(100.0%)
Google	4	(0.0%)

Employers with the most signoffs (total 184):

Employer	#	%
IBM	184	(100.0%)

Employers with the most hackers (total 32):

Employer	#	%
IBM	31	(96.9%)
Google	1	(3.1%)

5.1.88 skiboot-5.9-rc1

skiboot v5.9-rc1 was released on Wednesday October 11th 2017. It is the first release candidate of skiboot 5.9, which will become the new stable release of skiboot following the 5.8 release, first released August 31st 2017.

skiboot v5.9-rc1 contains all bug fixes as of *skiboot-5.4.7* and *skiboot-5.1.21* (the currently maintained stable releases). We do not currently expect to do any 5.8.x stable releases.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.9 by October 17th, with skiboot 5.9 being for all POWER8 and POWER9 platforms in op-build v1.20 (Due October 18th). This release will be targetted to early POWER9 systems.

Over skiboot-5.8, we have the following changes:

New Features

POWER8

- fast-reset by default (if possible)

  Currently, this is limited to POWER8 systems.

  A normal reboot will, rather than doing a full IPL, go through a fast reboot procedure. This reduces the "reboot to petitboot" time from minutes to a handful of seconds.

POWER9

- POWER9 power management during boot

  Less power should be consumed during boot.

- OPAL_SIGNAL_SYSTEM_RESET for POWER9

  This implements OPAL_SIGNAL_SYSTEM_RESET, using scom registers to quiesce the target thread and raise a system reset exception on it. It has been tested on DD2 with stop0 ESL=0 and ESL=1 shallow power saving modes.

  DD1 is not implemented because it is sufficiently different as to make support difficult.

- Enable deep idle states for POWER9

  - SLW: Add support for p9_stop_api

    p9_stop_api's are used to set SPR state on a core wakeup form a deeper low power state. p9_stop_api uses low level platform formware and self-restore microcode to restore the sprs to requested values.

    Code is taken from : https://github.com/open-power/hostboot/tree/master/src/import/chips/p9/procedures/utils/stopreg

  - SLW: Removing timebase related flags for stop4

    When a core enters stop4, it does not loose decrementer and time base. Hence removing flags OPAL_PM_DEC_STOP and OPAL_PM_TIMEBASE_STOP.

  - SLW: Allow deep states if homer address is known

    Use a common variable has_wakeup_engine instead of has_slw to tell if the: - SLW image is populated in case of power8 - CME image is populated in case of power9

    Currently we expect CME to be loaded if homer address is known (except for simulators)

  - SLW: Configure self-restore for HRMOR

    Make a stop api call using libpore to restore HRMOR register. HRMOR needs to be cleared so that when thread exits stop, they arrives at linux system_reset vector (0x100).

  - SLW: Add opal_slw_set_reg support for power9

    This OPAL call is made from Linux to OPAL to configure values in various SPRs after wakeup from a deep idle state.

- PHB4: CAPP recovery

  CAPP recovery is initiated when a CAPP Machine Check is detected. The capp recovery procedure is initiated via a Hypervisor Maintenance interrupt (HMI).

CAPP Machine Check may arise from either an error that results in a PHB freeze or from an internal CAPP error with CAPP checkstop FIR action. An error that causes a PHB freeze will result in the link down signal being asserted. The system continues running and the CAPP and PSL will be re-initialized.

This implements CAPP recovery for POWER9 systems

- Add `wafer-location` property for POWER9

  Extract wafer-location from ECID and add property under xscom node. - bits 64:71 are the chip x location (7:0) - bits 72:79 are the chip y location (7:0)

  Sample output:

```
[root@wsp xscom@623fc00000000]# lsprop ecid
ecid              019a00d4 03100718 852c0000 00fd7911
[root@wsp xscom@623fc00000000]# lsprop wafer-location
wafer-location    00000085 0000002c
```

- Add `wafer-id` property for POWER9

  Wafer id is derived from ECID data. - bits 4:63 are the wafer id (ten 6 bit fields each containing a code)

  Sample output:

```
[root@wsp xscom@623fc00000000]# lsprop ecid
ecid              019a00d4 03100718 852c0000 00fd7911
[root@wsp xscom@623fc00000000]# lsprop wafer-id
wafer-id          "6Q0DG340SO"
```

- Add `ecid` property under `xscom` node for POWER9. Sample output:

```
[root@wsp xscom@623fc00000000]# lsprop ecid
ecid              019a00d4 03100718 852c0000 00fd7911
```

- Add ibm,firmware-versions device tree node

  In P8, hostboot provides mini device tree. It contains `/ibm,firmware-versions` node which has various firmware component version details.

  In P9, OPAL is building device tree. This patch adds support to parse VERSION section of PNOR and create `/ibm,firmware-versions` device tree node.

  Sample output:

```
/sys/firmware/devicetree/base/ibm,firmware-versions # lsprop .
occ               "6a00709"
skiboot           "v5.7-rc1-p344fb62"
buildroot         "2017.02.2-7-g23118ce"
capp-ucode        "9c73e9f"
petitboot         "v1.4.3-p98b6d83"
sbe               "02021c6"
open-power        "witherspoon-v1.17-128-gf1b53c7-dirty"
....
....
```

POWER9

- Disable Transactional Memory on Power9 DD 2.1

Update pa_features_p9[] to disable TM (Transactional Memory). On DD 2.1 TM is not usable by Linux without other workarounds, so skiboot must disable it.

- xscom: Do not print error message for 'chiplet offline' return values

  xscom_read/write operations returns CHIPLET_OFFLINE when chiplet is offline. Some multicast xscom_read/write requests from HBRT results in xscom operation on offline chiplet(s) and printing below warnings in OPAL console:

  ```
  [ 135.036327572,3] XSCOM: Read failed, ret = -14
  [ 135.092689829,3] XSCOM: Read failed, ret = -14
  ```

  Some SCOM users can deal correctly with this error code (notably opal-prd), so the error message is (in practice) erroneous.

- IMC: Fix the core_imc_event_mask

  CORE_IMC_EVENT_MASK is a scom that contains bits to control event sampling for different machine state for core imc. The current event-mask setting sample events only on host kernel (hypervisor) and host userspace.

  Patch to enable the sampling of events in other machine states (like guest kernel and guest userspace).

- IMC: Update the nest_pmus array with occ/gpe microcode uav updates

  OOC/gpe nest microcode maintains the list of individual nest units supported. Sync the recent updates to the UAV with nest_pmus array.

  For reference occ/gpr microcode link for the UAV: https://github.com/open-power/occ/blob/master/src/occ_gpe1/gpe1_24x7.h

- Parse IOSLOT information from HDAT

  Add structure definitions that describe the physical PCIe topology of a system and parse them into the device-tree based PCIe slot description.

- idle: user context state loss flags fix for stop states

  The "lite" stop variants with PSSCR[ESL]=PSSCR[EC]=1 do not lose user context, while the non-lite variants do (ESL: enable state loss).

  Some of the POWER9 idle states had these wrong.

CAPI

- POWER9 DD2 update

  The CAPI initialization sequence has been updated in DD2. This patch adapts to the changes, retaining compatibility with DD1. The patch includes some changes to DD1 fix-ups as well.

- Load CAPP microcode for POWER9 DD2.0 and DD2.1
- capi: Mask Psl Credit timeout error for POWER9

  Mask the PSL credit timeout error in CAPP FIR Mask register bit(46). As per the h/w team this error is now deprecated and shouldn't cause any fir-action for P9.

NVLINK2

A notabale change is that we now generate the device tree description of NVLINK based on the HDAT we get from hostboot. Since Hostboot will generate HDAT based on VPD, you now *MUST* have correct VPD programmed or we will *default* to a Sequoia layout, which will lead to random problems if you are not booting a Sequoia Witherspoon planar. In the case of booting with old VPD and/or Hostboot, we print a **giant scary warning** in order to scare you.

- npu2: Read slot label from the HDAT link node

  Binding GPU to emulated NPU PCI devices is done using the slot labels since the NPU devices do not have a patching slot node we need to copy the label in here.

- npu2: Copy link speed from the npu HDAT node

  This needs to be in the PCI device node so the speed of the NVLink can be passed to the GPU driver.

- npu2: hw-procedures: Add settings to PHY_RESET

  Set a few new values in the PHY_RESET procedure, as specified by our updated programming guide documentation.

- Parse NVLink information from HDAT

  Add the per-chip structures that descibe how the A-Bus/NVLink/OpenCAPI phy is configured. This generates the npu@xyz nodes for each chip on systems that support it.

- npu2: Add vendor cap for IRQ testing

  Provide a way to test recoverable data link interrupts via a new vendor capability byte.

- npu2: Enable recoverable data link (no-stall) interrupts

  Allow the NPU2 to trigger “recoverable data link” interrupts.

- npu2: Implement basic FLR (Function Level Reset)
- npu2: hw-procedures: Update PHY DC calibration procedure
- npu2: hw-procedures: Change rx_pr_phase_step value

XIVE

- xive: Fix opal_xive_dump_tm() to access W2 properly. The HW only supported limited access sizes.
- xive: Make opal_xive_allocate_irq() properly try all chips

  When requested via OPAL_XIVE_ANY_CHIP, we need to try all chips. We first try the current one (on which the caller sits) and if that fails, we iterate all chips until the allocation succeeds.

- xive: Fix initialization & cleanup of HW thread contexts

  Instead of trying to “pull” everything and clear VT (which didn’t work and caused some FIRs to be set), instead just clear and then set the PTER thread enable bit. This has the side effect of completely resetting the corresponding thread context.

  This fixes the spurrious XIVE FIRs reported by PRD and fircheck

- xive: Add debug option for detecting misrouted IPI in emulation

  This is high overhead so we don’t enable it by default even in debug builds, it’s also a bit messy, but it allowed me to detect and debug a locking issue earlier so it can be useful.

- xive: Increase the interrupt “gap” on debug builds

  We normally allocate IPIs from 0x10. Make that 0x1000 on debug builds to limit the chances of overlapping with Linux interrupt numbers which makes debugging code that confuses them easier.

  Also add a warning in emulation if we get an interrupt in the queue whose number is below the gap.

- xive: Fix locking around cache scrub & watch

  Thankfully the missing locking only affects debug code and init code that doesn't run concurrently. Also adds a DEBUG option that checks the lock is properly held.

- xive: Workaround HW issue with scrub facility

  Without this, we sometimes don't observe from a CPU the values written to the ENDs or NVTs via the cache watch.

- xive: Add exerciser for cache watch/scrub facility in DEBUG builds
- xive: Make assertion in xive_eq_for_target() more informative
- xive: Add debug code to check initial cache updates
- xive: Ensure pressure relief interrupts are disabled

  We don't use them and we hijack the VP field with their configuration to store the EQ reference, so make sure the kernel or guest can't turn them back on by doing MMIO writes to ACK#

- xive: Don't try setting the reserved ACK# field in VPs

  That doesn't work, the HW doesn't implement it in the cache watch facility anyway.

- xive: Remove useless memory barriers in VP/EQ inits

  We no longer update "live" memory structures, we use a temporary copy on the stack and update the actual memory structure using the cache watch, so those barriers are pointless.

PHB4

- phb4: Mask RXE_ARB: DEC Stage Valid Error

  Change the inits to mask out the RXE ARB: DEC Stage Valid Error (bit 370. This has been a fatal error but should be informational only.

  This update will be in the next version of the phb4 workbook.

- phb4: Add additional adapter to retrain whitelist

  The single port version of the ConnectX-5 has a different device ID 0x1017. Updated descriptions to match pciutils database.

- PHB4: Default to PCIe GEN3 on POWER9 DD2.00

  You can use the NVRAM override for DD2.00 screened parts.

- phb4: Retrain link if degraded

  On P9 Scale Out (Nimbus) DD2.0 and Scale in (Cumulus) DD1.0 (and below) the PCIe PHY can lockup causing training issues. This can cause a degradation in speed or width in ~5% of training cases (depending on the card). This is fixed in later chip revisions. This issue can also cause PCIe links to not train at all, but this case is already handled.

  This patch checks if the PCIe link has trained optimally and if not, does a full PHB reset (to fix the PHY lockup) and retrain.

  One complication is some devices are known to train degraded unless device specific configuration is performed. Because of this, we only retrain when the device is in a whitelist. All devices in the current whitelist have been testing on a P9DSU/Boston, ZZ and Witherspoon.

  We always gather information on the link and print it in the logs even if the card is not in the whitelist.

For testing purposes, there's an nvram to retry all PCIe cards and all P9 chips when a degraded link is detected. The new option is 'pci-retry-all=true' which can be set using: *nvram -p ibm,skiboot –update-config pci-retry-all=true*. This option may increase the boot time if used on a badly behaving card.

IBM FSP platforms

- FSP/NVRAM: Handle "get vNVRAM statistics" command

  FSP sends MBOX command (cmd : 0xEB, subcmd : 0x05, mod : 0x00) to get vNVRAM statistics. OPAL doesn't maintain any such statistics. Hence return FSP_STATUS_INVALID_SUBCMD.

  Fixes these messages appearing in the OPAL log:

```
[16944.384670488,3] FSP: Unhandled message eb0500
[16944.474110465,3] FSP: Unhandled message eb0500
[16945.111280784,3] FSP: Unhandled message eb0500
[16945.293393485,3] FSP: Unhandled message eb0500
```

- fsp: Move common prints to trace

  These two prints just end up filling the skiboot logs on any machine that's been booted for more than a few hours.

  They have never been useful, so make them trace level. They were: :: SURV: Received heartbeat acknowledge from FSP SURV: Sending the heartbeat command to FSP

BMC based systems

- hw/lpc-uart: read from RBR to clear character timeout interrupts

  When using the aspeed SUART, we see a condition where the UART sends continuous character timeout interrupts. This change adds a (heavily commented) dummy read from the RBR to clear the interrupt condition on init.

  This was observed on p9dsu systems, but likely applies to other systems using the SUART.

- astbmc: Add methods for handing Device Tree based slots e.g. ones from HDAT on POWER9.

General

- ipmi: Convert common debug prints to trace

  OPAL logs messages for every IPMI request from host. Sometime OPAL console is filled with only these messages. This path is pretty stable now and we have enough logs to cover bad path. Hence lets convert these debug message to trace/info message. Examples are:

```
[ 1356.423958816,7] opal_ipmi_recv(cmd: 0xf0 netfn: 0x3b resp_size: 0x02)
[ 1356.430774496,7] opal_ipmi_send(cmd: 0xf0 netfn: 0x3a len: 0x3b)
[ 1356.430797392,7] BT: seq 0x20 netfn 0x3a cmd 0xf0: Message sent to host
[ 1356.431668496,7] BT: seq 0x20 netfn 0x3a cmd 0xf0: IPMI MSG done
```

- libflash/file: Handle short read()s and write()s correctly

  Currently we don't move the buffer along for a short read() or write() and nor do we request only the remaining amount.

- hw/p8-i2c: Rework timeout handling

  Currently we treat a timeout as a hard failure and will automatically fail any transations that hit their timeout. This results in unnecessarily failing I2C requests if interrupts are dropped, etc. Although these are bad things that we should log we can handle them better by checking the actual hardware status and completing the transation if there are no real errors. This patch reworks the timeout handling to check the status and continue the transaction if it can. if it can while logging an error if it detects a timeout due to a dropped interrupt.

- core/flash: Only expect ELF header for BOOTKERNEL partition flash resource

  When loading a flash resource which isn't signed (secure and trusted boot) and which doesn't have a subpartition, we assume it's the BOOTKERNEL since previously this was the only such resource. Thus we also assumed it had an ELF header which we parsed to get the size of the partition rather than trusting the actual_size field in the FFS header. A previous commit (9727fe3 DT: Add ibm,firmware-versions node) added the version resource which isn't signed and also doesn't have a subpartition, thus we expect it to have an ELF header. It doesn't so we print the error message "FLASH: Invalid ELF header part VERSION".

  It is a fluke that this works currently since we load the secure boot header unconditionally and this happen to be the same size as the version partition. We also don't update the return code on error so happen to return OPAL_SUCCESS.

  To make this explicitly correct; only check for an ELF header if we are loading the BOOTKERNEL resource, otherwise use the partition size from the FFS header. Also set the return code on error so we don't erroneously return OPAL_SUCCESS. Add a check that the resource will fit in the supplied buffer to prevent buffer overrun.

- flash: Support adding the no-erase property to flash

  The mbox protocol explicitly states that an erase is not required before a write. This means that issuing an erase from userspace, through the mtd device, and back returns a successful operation that does nothing. Unfortunately, this makes userspace tools unhappy. Linux MTD devices support the MTD_NO_ERASE flag which conveys that writes do not require erases on the underlying flash devices. We should set this property on all of our devices which do not require erases to be performed.

  NOTE: This still requires a linux kernel component to set the MTD_NO_ERASE flag from the device tree property.

Utilities

- external/gard: Clear entire guard partition instead of entry by entry

  When using the current implementation of the gard tool to ecc clear the entire GUARD partition it is done one gard record at a time. While this may be ok when accessing the actual flash this is very slow when done from the host over the mbox protocol (on the order of 4 minutes) because the bmc side is required to do many read, erase, writes under the hood.

  Fix this by rewriting the gard tool reset_partition() function. Now we allocate all the erased guard entries and (if required) apply ecc to the entire buffer. Then we can do one big erase and write of the entire partition. This reduces the time to clear the guard partition to on the order of 4 seconds.

- opal-prd: Fix opal-prd command line options

  HBRT OCC reset interface depends on service processor type.

  - FSP: reset_pm_complex()
  - BMC: process_occ_reset()

  We have both *occ* and *pm-complex* command line interfaces. This patch adds support to dispaly appropriate message depending on system type.

SP	Command	Action
FSP	opal-prd occ	display error message
FSP	opal-prd pm-complex	Call pm_complex_reset()
BMC	opal-prd occ	Call process_occ_reset()
BMC	opal-prd pm-complex	display error message

- opal-prd: detect service processor type and then make appropriate occ reset call.
- pflash: Fix erase command for unaligned start address

  The erase_range() function handles erasing the flash for a given start address and length, and can handle an unaligned start address and length. However in the unaligned start address case we are incorrectly calculating the remaining size which can lead to incomplete erases.

  If we're going to update the remaining size based on what the start address was then we probably want to do that before we overide the origin start address. So rearrange the code so that this is indeed the case.
- external/gard: Print an error if run on an FSP system

Simulators

- mambo: Add mambo socket program

  This adds a program that can be run inside a mambo simulator in linux userspace which enables TCP sockets to be proxied in and out of the simulator to the host.

  Unlike mambo bogusnet, it's requires no linux or skiboot specific drivers/infrastructure to run.

  Run inside the simulator:

  – to forward host ssh connections to sim ssh server: `./mambo-socket-proxy -h 10022 -s 22`, then connect to port 10022 on your host with `ssh -p 10022 localhost`
  – to allow http proxy access from inside the sim to local http proxy: `./mambo-socket-proxy -b proxy.mynetwork -h 3128 -s 3128`

  Multiple connections are supported.
- idle: disable stop*_lite POWER9 idle states for Mambo platform

  Mambo prior to Mambo.7.8.21 had a bug where the stop idle instruction with PSSCR[ESL]=PSSCR[EC]=0 would resume with MSR set as though it had taken a system reset interrupt.

  Linux currently executes this instruction with MSR already set that way, so the problem went unnoticed. A proposed patch to Linux changes that, and causes the idle code to crash. Work around this by disabling lite stop states for the mambo platform for now.

5.1.89 skiboot-5.9-rc2

skiboot v5.9-rc2 was released on Monday October 16th 2017. It is the second release candidate of skiboot 5.9, which will become the new stable release of skiboot following the 5.8 release, first released August 31st 2017.

skiboot v5.9-rc2 contains all bug fixes as of *skiboot-5.4.8* and *skiboot-5.1.21* (the currently maintained stable releases). We do not currently expect to do any 5.8.x stable releases.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.9 by October 17th, with skiboot 5.9 being for all POWER8 and POWER9 platforms in op-build v1.20 (Due October 18th). This release will be targetted to early POWER9 systems.

Over *skiboot-5.9-rc1*, we have the following changes:

- opal-prd: Fix memory leak
- hdata/i2c: update the list of known i2c devs

  This updates the list of known i2c devices - as of HDAT spec v10.5e - so that they can be properly identified during the hdat parsing.

- hdata/i2c: log unknown i2c devices

  An i2c device is unknown if either the i2c device list is outdated or the device is marked as unknown (0xFF) in the hdat.

- opal/cpu: Mark the core as bad while disabling threads of the core.

  If any of the core fails to sync its TB during chipTOD initialization, all the threads of that core are disabled. But this does not make linux kernel to ignore the core/cpus. It crashes while bringing them up with below backtrace:

  ```
  [   38.883898] kexec_core: Starting new kernel
  cpu 0x0: Vector: 300 (Data Access) at [c0000003f277b730]
      pc: c0000000001b9890: internal_create_group+0x30/0x304
      lr: c0000000001b9880: internal_create_group+0x20/0x304
      sp: c0000003f277b9b0
     msr: 900000000280b033
     dar: 40
   dsisr: 40000000
    current = 0xc0000003f9f41000
    paca    = 0xc00000000fe00000   softe: 0        irq_happened: 0x01
      pid   = 2572, comm = kexec
  Linux version 4.13.2-openpower1 (jenkins@p89) (gcc version 6.4.0 (Buildroot 2017.
  ↪08-00006-g319c6e1)) #1 SMP Wed Sep 20 05:42:11 UTC 2017
  enter ? for help
  [c0000003f277b9b0] c0000000008a8780 (unreliable)
  [c0000003f277ba50] c00000000041c3ac topology_add_dev+0x2c/0x40
  [c0000003f277ba70] c00000000006b078 cpuhp_invoke_callback+0x88/0x170
  [c0000003f277bac0] c00000000006b22c cpuhp_up_callbacks+0x54/0xb8
  [c0000003f277bb10] c00000000006bc68 cpu_up+0x11c/0x168
  [c0000003f277bbc0] c0000000002f0e0 default_machine_kexec+0x1fc/0x274
  [c0000003f277bc50] c0000000002e2d8 machine_kexec+0x50/0x58
  [c0000003f277bc70] c0000000000de4e8 kernel_kexec+0x98/0xb4
  [c0000003f277bce0] c00000000008b0f0 SyS_reboot+0x1c8/0x1f4
  [c0000003f277be30] c00000000000b118 system_call+0x58/0x6c
  ```

- hw/imc: pause microcode at boot

  IMC nest counters has both in-band (ucode access) and out of band access to it. Since not all nest counter configurations are supported by ucode, out of band tools are used to characterize other configuration.

  So it is prefer to pause the nest microcode at boot to aid the nest out of band tools. If the ucode not paused and OS does not have IMC driver support, then out to band tools will race with ucode and end up getting undesirable values. Patch to check and pause the ucode at boot.

  OPAL provides APIs to control IMC counters. OPAL_IMC_COUNTERS_INIT is used to initialize these counters at boot. OPAL_IMC_COUNTERS_START and OPAL_IMC_COUNTERS_STOP API calls should be used to start and pause these IMC engines. *doc/opal-api/opal-imc-counters.rst* details the OPAL APIs and their usage.

- xive: Fix VP free block group mode false-positive parameter check

  The check to ensure the buddy allocation idx is aligned to its allocation order was not taking into account the allocation split. This would result in opal_xive_free_vp_block failures despite giving the same value as returned by opal_xive_alloc_vp_block.

E.g., starting then stopping 4 KVM guests gives the following pattern in the host:

```
opal_xive_alloc_vp_block(5)=0x45000020
opal_xive_alloc_vp_block(5)=0x45000040
opal_xive_alloc_vp_block(5)=0x45000060
opal_xive_alloc_vp_block(5)=0x45000080
opal_xive_free_vp_block(0x45000020)=-1
opal_xive_free_vp_block(0x45000040)=0
opal_xive_free_vp_block(0x45000060)=-1
opal_xive_free_vp_block(0x45000080)=0
```

- hw/p8-i2c: Fix deadlock in p9_i2c_bus_owner_change

  When debugging a system where Linux was taking soft lockup errors with two CPUs stuck in OPAL:

CPU0	CPU1
lock	
p8_i2c_recover	
opal_handle_interrupt	sync_timer cancel_timer p9_i2c_bus_owner_change occ_p9_interrupt xive_source_interrupt opal_handle_interrupt

  p8_i2c_recover() is a timer, and is stuck trying to take master->lock. p9_i2c_bus_owner_change() has taken master->lock, but then is stuck waiting for all timers to complete. We deadlock.

  Fix this by using cancel_timer_async().

- FSP/CONSOLE: Limit number of error logging

  Commit c8a7535f (FSP/CONSOLE: Workaround for unresponsive ipmi daemon) added error logging when buffer is full. In some corner cases kernel may call this function multiple time and we may endup logging error again and again.

  This patch fixes it by generating error log only once.

- FSP/CONSOLE: Fix fsp_console_write_buffer_space() call

  Kernel calls fsp_console_write_buffer_space() to check console buffer space availability. If there is enough buffer space to write data, then kernel will call fsp_console_write() to write actual data.

  In some extreme corner cases (like one explained in commit c8a7535f) console becomes full and this function returns 0 to kernel (or space available in console buffer < next incoming data size). Kernel will continue retrying until it gets enough space. So we will start seeing RCU stalls.

  This patch keeps track of previous available space. If previous space is same as current means not enough space in console buffer to write incoming data. It may be due to very high console write operation and slow response from FSP -OR- FSP has stopped processing data (ex: because of ipmi daemon died). At this point we will start timer with timeout of SER_BUFFER_OUT_TIMEOUT (10 secs). If situation is not improved within 10 seconds means something went bad. Lets return OPAL_RESOURCE so that kernel can drop console write and continue.

- FSP/CONSOLE: Close SOL session during R/R

  Presently we are not closing SOL and FW console sessions during R/R. Host will continue to write to SOL buffer during FSP R/R. If there is heavy console write operation happening during FSP R/R (like running *top* command inside console), then at some point console buffer becomes full. fsp_console_write_buffer_space() returns 0 (or less than required space to write data) to host. While one thread is busy writing to console, if some other threads tries to write data to console we may see RCU stalls (like below) in kernel.

```
[ 2082.828363] INFO: rcu_sched detected stalls on CPUs/tasks: { 32} (detected by␣
↪16, t=6002 jiffies, g=23154, c=23153, q=254769)
[ 2082.828365] Task dump for CPU 32:
[ 2082.828368] kworker/32:3    R  running task        0  4637      2 0x00000884
[ 2082.828375] Workqueue: events dump_work_fn
[ 2082.828376] Call Trace:
[ 2082.828382] [c000000f1633fa00] [c00000000013b6b0] console_unlock+0x570/0x600␣
↪(unreliable)
[ 2082.828384] [c000000f1633fae0] [c00000000013ba34] vprintk_emit+0x2f4/0x5c0
[ 2082.828389] [c000000f1633fb60] [c00000000099e644] printk+0x84/0x98
[ 2082.828391] [c000000f1633fb90] [c0000000000851a8] dump_work_fn+0x238/0x250
[ 2082.828394] [c000000f1633fc60] [c0000000000ecb98] process_one_work+0x198/0x4b0
[ 2082.828396] [c000000f1633fcf0] [c0000000000ed3dc] worker_thread+0x18c/0x5a0
[ 2082.828399] [c000000f1633fd80] [c0000000000f4650] kthread+0x110/0x130
[ 2082.828403] [c000000f1633fe30] [c000000000009674] ret_from_kernel_thread+0x5c/
↪0x68
```

Hence lets close SOL (and FW console) during FSP R/R.

- FSP/CONSOLE: Do not associate unavailable console

  Presently OPAL sends associate/unassociate MBOX command for all FSP serial console (like below OPAL message). We have to check console is available or not before sending this message.

```
[ 5013.227994012,7] FSP: Reassociating HVSI console 1
[ 5013.227997540,7] FSP: Reassociating HVSI console 2
```

- FSP: Disable PSI link whenever FSP tells OPAL about impending R/R

  Commit 42d5d047 fixed scenario where DPO has been initiated, but FSP went into reset before the CEC power down came in. But this is generic issue that can happen in normal shutdown path as well.

  Hence disable PSI link as soon as we detect FSP impending R/R.

- fsp: return OPAL_BUSY_EVENT on failure sending FSP_CMD_POWERDOWN_NORM Also, return OPAL_BUSY_EVENT on failure sending FSP_CMD_REBOOT / DEEP_REBOOT.

  We had a race condition between FSP Reset/Reload and powering down the system from the host:

  Roughly:

#	FSP	Host
1	Power on	
2		Power on
3	(inject EPOW)	
4	(trigger FSP R/R)	
5		Processes EPOW event, starts shutting down
6		calls OPAL_CEC_POWER_DOWN
7	(is still in R/R)	
8		gets OPAL_INTERNAL_ERROR, spins in opal_poll_events
9	(FSP comes back)	
10		spinning in opal_poll_events
11	(thinks host is running)	

  The call to OPAL_CEC_POWER_DOWN is only made once as the reset/reload error path for fsp_sync_msg() is to return -1, which means we give the OS OPAL_INTERNAL_ERROR, which is fine, except that our own API docs give us the opportunity to return OPAL_BUSY when trying again later may be successful, and we're ambiguous as to if you should retry on OPAL_INTERNAL_ERROR.

For reference, the linux code looks like this:

```
static void __noreturn pnv_power_off(void)
{
        long rc = OPAL_BUSY;

        pnv_prepare_going_down();

        while (rc == OPAL_BUSY || rc == OPAL_BUSY_EVENT) {
                rc = opal_cec_power_down(0);
                if (rc == OPAL_BUSY_EVENT)
                        opal_poll_events(NULL);
                else
                        mdelay(10);
        }
        for (;;)
                opal_poll_events(NULL);
}
```

Which means that *practically* our only option is to return OPAL_BUSY or OPAL_BUSY_EVENT.

We choose OPAL_BUSY_EVENT for FSP systems as we do want to ensure we're running pollers to communicate with the FSP and do the final bits of Reset/Reload handling before we power off the system.

5.1.90 skiboot-5.9-rc3

skiboot v5.9-rc3 was released on Wednesday October 18th 2017. It is the third release candidate of skiboot 5.9, which will become the new stable release of skiboot following the 5.8 release, first released August 31st 2017.

skiboot v5.9-rc3 contains all bug fixes as of *skiboot-5.4.8* and *skiboot-5.1.21* (the currently maintained stable releases). We do not currently expect to do any 5.8.x stable releases.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.9 by October 20th, with skiboot 5.9 being for all POWER8 and POWER9 platforms in op-build v1.20 (Due October 18th). This release will be targetted to early POWER9 systems.

Over *skiboot-5.9-rc2*, we have the following changes:

- Improvements to vpd device tree entries

  Previously we would miss some properties

- Revert "npu2: Add vendor cap for IRQ testing"

  This reverts commit 9817c9e29b6fe00daa3a0e4420e69a97c90eb373 which seems to break setting the PCI dev flag and the link number in the PCIe vendor specific config space. This leads to the device driver attempting to re-init the DL when it shouldn't which can cause HMI's.

- hw/imc: Fix IMC Catalog load for DD2.X processors

- cpu: Add OPAL_REINIT_CPUS_TM_SUSPEND_DISABLED

  Add a new CPU reinit flag, "TM Suspend Disabled", which requests that CPUs be configured so that TM (Transactional Memory) suspend mode is disabled.

  Currently this always fails, because skiboot has no way to query the state. A future hostboot change will add a mechanism for skiboot to determine the status and return an appropriate error code.

5.1.91 skiboot-5.9-rc4

skiboot v5.9-rc4 was released on Thursday October 19th 2017. It is the fourth release candidate of skiboot 5.9, which will become the new stable release of skiboot following the 5.8 release, first released August 31st 2017.

skiboot v5.9-rc4 contains all bug fixes as of *skiboot-5.4.8* and *skiboot-5.1.21* (the currently maintained stable releases). We do not currently expect to do any 5.8.x stable releases.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.9 by October 20th, with skiboot 5.9 being for all POWER8 and POWER9 platforms in op-build v1.20 (Due October 18th, so we're running a bit behind there). This release will be targetted to early POWER9 systems.

Over *skiboot-5.9-rc3*, we have the following changes:

- phb4: Fix PCIe GEN4 on DD2.1 and above

  In this change: eef0e197ab PHB4: Default to PCIe GEN3 on POWER9 DD2.00

  We clamped DD2.00 parts to GEN3 but unfortunately this change also applies to DD2.1 and above.

  This fixes this to only apply to DD2.00.

- occ-sensors : Add OCC inband sensor region to exports (useful for debugging)

Two SRESET fixes:

- core: direct-controls: Fix clearing of special wakeup

  'special_wakeup_count' is incremented on successfully asserting special wakeup. So we will never clear the special wakeup if we check 'special_wakeup_count' to be zero. Fix this issue by checking the 'special_wakeup_count' to 1 in dctl_clear_special_wakeup().

- core/direct-controls: increase special wakeup timeout on POWER9

  Some instances have been observed where the special wakeup assert times out. The current timeout is too short for deeper sleep states. Hostboot uses 100ms, so match that.

5.1.92 skiboot-5.9-rc5

skiboot v5.9-rc5 was released on Monday October 23rd 2017 approximately 32,000ft above somewhere north of Tucson, Arizona. It is the fifth release candidate of skiboot 5.9, which will become the new stable release of skiboot following the 5.8 release, first released August 31st 2017.

skiboot v5.9-rc5 contains all bug fixes as of *skiboot-5.4.8* and *skiboot-5.1.21* (the currently maintained stable releases). We do not currently expect to do any 5.8.x stable releases.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 5.9 very shortly, with skiboot 5.9 being for all POWER8 and POWER9 platforms in op-build v1.20 (Due October 18th, so we're running a bit behind there). This release will be targetted to early POWER9 systems.

Over *skiboot-5.9-rc3*, we have the following changes:

- opal/hmi: Workaround Power9 hw logic bug for couple of TFMR TB errors.
- opal/hmi: Fix TB reside and HDEC parity error recovery for power9
- phb4: Escalate freeze to fence to avoid checkstop

  Freeze events such as MMIO loads can cause the PHB to lose it's limited powerbus credits. If all credits are used and a further MMIO will cause a checkstop.

To work around this, we escalate the troublesome freeze events to a fence. The fence will cause a full PHB reset which resets the powerbus credits and avoids the checkstop.

- phb4: Update some init registers

  New inits based on next PHB4 workbook. Increases some timeouts to avoid some spurious error conditions.

- phb4: Enable PHB MMIO in phb4_root_port_init()

  Linux EEH flow is somewhat broken. It saves the PCIe config space of the PHB on boot, which it then uses to restore on EEH recovery. It does this to restore MMIO bars and some other pieces.

  Unfortunately this save is done before any drivers are bound to devices under the PHB. A number of other things are configured in the PHB after drivers start, hence some configuration space settings aren't saved correctly. These include bus master and MMIO bits in the command register.

  Linux tried to hack around this in this linux commit `bf898ec5cb` powerpc/eeh: Enable PCI_COMMAND_MASTER for PCI bridges This sets the bus master bit but ignores the MMIO bit.

  Hence we lose MMIO after a full PHB reset. This causes the next MMIO access to the device to fail and for us to perform a PE freeze recovery, which still doesn't set the MMIO bit and hence we still fail.

  This works around this by forcing MMIO on during phb4_root_port_init().

  With this we can recovery from a PHB fence event on POWER9.

- phb4: Reduce link degraded message log level to debug

  If we hit this message we'll retry and fix the problem. If we run out of retries and can't fix the problem, we'll still print a log message at error level indicating a problem.

- phb4: Fix GEN3 for DD2.00

  In this fix: `62ac7631ae` "phb4: Fix PCIe GEN4 on DD2.1 and above", We fixed DD2.1 GEN4 but broke DD2.00 as GEN3.

  This fixes DD2.00 back to GEN3. This time for sure!

5.1.93 skiboot-5.9.1

skiboot 5.9.1 was released on Tuesday November 14th, 2017. It replaces *skiboot-5.9* as the current stable release in the 5.9.x series.

Over *skiboot-5.9*, we have two NPU2 (NVLink2) fixes and two XIVE bug fixes:

- npu2: hw-procedures: Refactor reset_ntl procedure

  Change the implementation of reset_ntl to match the latest programming guide documentation.

- npu2: hw-procedures: Add phy_rx_clock_sel()

  Change the RX clk mux control to be done by software instead of HW. This avoids glitches caused by changing the mux setting.

- xive: Fix ability to clear some EQ flags

  We could never clear "unconditional notify" and "escalate"

- xive: Update inits for DD2.0

  This updates some inits based on information from the HW designers. This includes enabling some new DD2.0 features that we don't yet exploit.

5.1.94 skiboot-5.9.2

skiboot 5.9.2 was released on Thursday November 16th, 2017. It replaces *skiboot-5.9.1* as the current stable release in the 5.9.x series.

Over *skiboot-5.9.1*, we have a few PHB4 (PCI) fixes, an i2c fix for POWER9 platforms to avoid conflicting with the OCC use and an important NPU2 (NVLink2) fix.

- phb4: Fix lane equalisation setting

  Fix cut and paste from phb3. The sizes have changes now we have GEN4, so the check here needs to change also

  Without this we end up with the default settings (all '7') rather than what's in HDAT.

- phb4: Fix PE mapping of M32 BAR

  The M32 BAR is the PHB4 region used to map all the non-prefetchable or 32-bit device BARs. It's supposed to have its segments remapped via the MDT and Linux relies on that to assign them individual PE#.

  However, we weren't configuring that properly and instead used the mode where PE# == segment#, thus causing EEH to freeze the wrong device or PE#.

- phb4: Fix lost bit in PE number on config accesses

  A PE number can be up to 9 bits, using a uint8_t won't fly..

  That was causing error on config accesses to freeze the wrong PE.

- phb4: Update inits

  New init value from HW folks for the fence enable register.

  This clears bit 17 (CFG Write Error CA or UR response) and bit 22 (MMIO Write DAT_ERR Indication) and sets bit 21 (MMIO CFG Pending Error)

- npu2: Move to new GPU memory map

  There are three different ways we configure the MCD and memory map.

  1) Old way (current way) Skiboot configures the MCD and puts GPUs at 4TB and below
  2) New way with MCD Hostboot configures the MCD and skiboot puts GPU at 4TB and above
  3) New way without MCD No one configures the MCD and skiboot puts GPU at 4TB and below

  The change keeps option 1 and adds options 2 and 3.

  The different configurations are detected using certain scoms (see patch).

  Option 1 will go away eventually as it's a configuration that can cause xstops or data integrity problems. We are keeping it around to support existing hostboot.

  Option 2 supports only 4 GPUs and 512GB of memory per socket.

  Option 3 supports 6 GPUs and 4TB of memory but may have some performance impact.

- p8-i2c: Don't write the watermark register at init

  On P9 the I2C master is shared with the OCC. Currently the watermark values are set once at init time which is bad for two reasons:

  a) We don't take the OCC master lock before setting it. Which may cause issues if the OCC is currently using the master.
  b) The OCC might change the watermark levels and we need to reset them.

  Change this so that we set the watermark value when a new transaction is started rather than at init time.

5.1.95 skiboot-5.9.3

skiboot 5.9.3 was released on Wednesday November 22nd, 2017. It replaces *skiboot-5.9.2* as the current stable release in the 5.9.x series.

Over *skiboot-5.9.2*, we have one NPU2/NVLink2 fix that causes the machine to crash hard in the event of hardware error rather than crash mysteriously later on whenever the NVLink2 links are used.

That fix is:

- npu2: hw-procedures: Add check_credits procedure

  As an immediate mitigator for a current hardware glitch, add a procedure that can be used to validate NTL credit values. This will be called as a safeguard to check that link training succeeded.

  Assert that things are exactly as we expect, because if they aren't, the system will experience a catastrophic failure shortly after the start of link traffic.

5.1.96 skiboot-5.9.4

skiboot 5.9.4 was released on Wednesday November 29th, 2017. It replaces *skiboot-5.9.3* as the current stable release in the 5.9.x series.

Over *skiboot-5.9.3*, we have one NPU2/NVLink2 fix that works around a potential glitch (the one *skiboot-5.9.3* would hard crash on rather than let a system continue to run until it mysteriously crashed later on).

That fix is in two parts:

- npu2: hw-procedures: Change phy_rx_clock_sel values to recover from a potential glitch.
- npu2: hw-procedures: Manipulate IOVALID during training

  Ensure that the IOVALID bit for this brick is raised at the start of link training, in the reset_ntl procedure.

  Then, to protect us from a glitch when the PHY clock turns off or gets chopped, lower IOVALID for the duration of the phy_reset and phy_rx_dccal procedures.

5.1.97 skiboot-5.9.5

skiboot 5.9.5 was released on Wednesday December 13th, 2017. It replaces *skiboot-5.9.4* as the current stable release in the 5.9.x series.

Over *skiboot-5.9.4*, we have a few bug fixes, they are:

- Fix *extremely* rare race in timer code.
- xive: Ensure VC informational FIRs are masked

  Some HostBoot versions leave those as checkstop, they are harmless and can sometimes occur during normal operations.

- xive: Fix occasional VC checkstops in xive_reset

  The current workaround for the scrub bug described in __xive_cache_scrub() has an issue in that it can leave dirty invalid entries in the cache.

  When cleaning up EQs or VPs during reset, if we then remove the underlying indirect page for these entries, the XIVE will checkstop when trying to flush them out of the cache.

  This replaces the existing workaround with a new pair of workarounds for VPs and EQs:

- The VP one does the dummy watch on another entry than the one we scrubbed (which does the job of pushing old stores out) using an entry that is known to be backed by a permanent indirect page.
- The EQ one switches to a more efficient workaround which consists of doing a non-side-effect ESB load from the EQ's ESe control bits.

- io: Add load_wait() helper

  This uses the standard form twi/isync pair to ensure a load is consumed by the core before continuing. This can be necessary under some circumstances for example when having the following sequence:

  - Store reg A
  - Load reg A (ensure above store pushed out)
  - delay loop
  - Store reg A

  IE, a mandatory delay between 2 stores. In theory the first store is only guaranteed to rach the device after the load from the same location has completed. However the processor will start executing the delay loop without waiting for the return value from the load.

  This construct enforces that the delay loop isn't executed until the load value has been returned.

- xive: Do not return a trigger page for an escalation interrupt

  This is bogus, we don't support them. (Thankfully the callers didn't actually try to use this on escalation interrupts).

- xive: Mark a freed IRQ's IVE as valid and masked

  Removing the valid bit means a FIR will trip if it's accessed inadvertently. Under some circumstances, the XIVE will speculatively access an IVE for a masked interrupt and trip it. So make sure that freed entries are still marked valid (but masked).

- hw/nx: Fix NX BAR assignments

  The NX rng BAR is used by each core to source random numbers for the DARN instruction. Currently we configure each core to use the NX rng of the chip that it exists on. Unfortunately, the NX can be deconfigured by hostboot and in this case we need to use the NX of a different chip.

  This patch moves the BAR assignments for the NX into the normal nx-rng init path. This lets us check if the normal (chip local) NX is active when configuring which NX a core should use so that we can fallback gracefully.

5.1.98 skiboot-5.9.6

skiboot 5.9.6 was released on Friday December 15th, 2017. It replaces *skiboot-5.9.5* as the current stable release in the 5.9.x series.

Over *skiboot-5.9.5*, we have a few bug fixes, they are:

- sensors: occ: Skip counter type of sensors

  Don't add counter type of sensors to device-tree as they don't fit into hwmon sensor interface.

- p9_stop_api updates to support IMC across deep stop states.
- opal/xscom: Add recovery for lost core wakeup scom failures.

  Due to a hardware issue where core responding to scom was delayed due to thread reconfiguration, leaves the SCOM logic in a state where the subsequent scom to that core can get errors. This is affected for Core PC scom registers in the range of 20010A80-20010ABF

The solution is if a xscom timeout occurs to one of Core PC scom registers in the range of 20010A80-20010ABF, a clearing scom write is done to 0x20010800 with data of '0x00000000' which will also get a timeout but clears the scom logic errors. After the clearing write is done the original scom operation can be retried.

The scom timeout is reported as status 0x4 (Invalid address) in HMER[21-23].

5.1.99 skiboot-5.9.7

skiboot 5.9.7 was released on Friday December 22nd, 2017. It replaces *skiboot-5.9.6* as the current stable release in the 5.9.x series.

Over *skiboot-5.9.6*, we have two bug fixes, they are:

- phb4: Change PCI MMIO timers

  Currently we have a mismatch between the NCU and PCI timers for MMIO accesses. The PCI timers must be lower than the NCU timers otherwise it may cause checkstops.

  This changes PCI timeouts controlled by skiboot to 33-50ms. It should be forwards and backwards compatible with expected hostboot changes to the NCU timer.

- p8-i2c: Limit number of retry attempts

  Currently we will attempt to start an I2C transaction until it succeeds. In the event that the OCC does not release the lock on an I2C bus this results in an async token being held forever and the kernel thread that started the transaction will block forever while waiting for an async completion message. Fix this by limiting the number of attempts to start the transaction.

5.1.100 skiboot-5.9.8

skiboot-5.9.8 was released on Friday January 5th, 2018. It replaces *skiboot-5.9.7* as the current stable release in the 5.9.x series.

Over *skiboot-5.9.7*, we have one new feature:

- Parse IPL FW feature settings

  Add parsing for the firmware feature flags in the HDAT. This indicates the settings of various parameters which are set at IPL time by firmware.

5.1.101 skiboot-5.9.9

skiboot 5.9.9 was released on Monday May 28th, 2018. It replaces *skiboot-5.9.8* as the current stable release in the 5.9.x series.

Over *skiboot-5.9.8*, we have two bug fixes and a build fix, they are:

- OPAL_PCI_SET_POWER_STATE: fix locking in error paths

  Otherwise we could exit OPAL holding locks, potentially leading to all sorts of problems later on.

- lpc: Clear pending IRQs at boot

  When we come in from hostboot the LPC master has the bus reset indicator set. This error isn't handled until the host kernel unmasks interrupts, at which point we get the following suprious error:

  ```
  [   20.053560375,3] LPC: Got LPC reset on chip 0x0 !
  [   20.053564560,3] LPC[000]: Unknown LPC error Error address reg: 0x00000000
  ```

Fix this by clearing the various error bits in the LPC status register before we initalise the skiboot LPC bus driver.

- stb: Build fixes in constructing secure and trusted boot header

5.1.102 skiboot-6.0

skiboot v6.0 was released on Friday May 11th 2018. It is the first release of skiboot 6.0, which is the new stable release of skiboot following the 5.11 release, first released April 6th 2018.

Skiboot 6.0 is the basis for op-build v2.0 and will is *required* for POWER9 systems.

skiboot v6.0 contains all bug fixes as of *skiboot-5.11*, *skiboot-5.10.5*, and *skiboot-5.4.9* (the currently maintained stable releases). We do *not* expect any further stable releases in the 5.10.x series, nor in the 5.11.x series.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

Over skiboot-5.11, we have the following changes:

New Features

Since 6.0-rc1:

- Update default stop-state-disable mask to cut only stop11

  Stability improvements in microcode for stop4/stop5 are available in upstream hcode images. Stop4 and stop5 can be safely enabled by default.

  Use ~0xE0000000 to cut all but stop0,1,2 in case there are any issues with stop4/5.

  example:

  ```
  nvram -p ibm,skiboot --update-config opal-stop-state-disable-mask=0x1FFFFFFF
  ```

  Note: that DD2.1 chips that have a frequency <1867Mhz possible *need* to run a hcode image *different* than the default in op-build (set *BR2_HCODE_LATEST_VERSION=y* in your config)

- ibm,firmware-versions: add hcode to device tree

  op-build commit 736a08b996e292a449c4996edb264011dfe56a40 added hcode to the VERSION partition, let's parse it out and let the user know.

- ipmi: Add BMC firmware version to device tree

  BMC Get device ID command gives BMC firmware version details. Lets add this to device tree. User space tools will use this information to display BMC version details.

Since 5.11:

- Disable stop states from OPAL

  On ZZ, stop4,5,11 are enabled for PowerVM, even though doing so may cause problems with OPAL due to bugs in hcode.

  For other platforms, this isn't so much of an issue as we can just control stop states by the MRW. However the rebuild-the-world approach to changing values there is a bit annoying if you just want to rule out a specific stop state from being problematic.

  Provide an nvram option to override what's disabled in OPAL.

  The OPAL mask is currently ~0xE0000000 (i.e. all but stop 0,1,2)

  You can set an NVRAM override with:

```
nvram -p ibm,skiboot --update-config opal-stop-state-disable-mask=0xFFFFFFF
```

This nvram override will disable *all* stop states.

- interrupts: Create an "interrupts" property in the OPAL node

  Deprecate the old "opal-interrupts", it's still there, but the new property follows the standard and allow us to specify whether an interrupt is level or edge sensitive.

  Similarly create "interrupt-names" whose content is identical to "opal-interrupts-names".

- SBE: Add timer support on POWER9

  SBE on P9 provides one shot programmable timer facility. We can use this to implement OPAL timers and hence limit the reliance on the Linux heartbeat (similar to HW timer facility provided by SLW on P8).

- Add SBE driver support

  SBE (Self Boot Engine) on P9 has two different jobs: - Boot the chip up to the point the core is functional - Provide various services like timer, scom, stash MPIPL, etc., at runtime

  We will use SBE for various purposes like timer, MPIPL, etc.

- opal:hmi: Add missing processor recovery reason string.

  With this patch now we see reason string printed for CORE_WOF[43] bit.

```
[  477.352234986,7] HMI: [Loc: U78D3.001.WZS004A-P1-C48]: P:8 C:22 T:3: Processor␣
↪recovery occurred.
[  477.352240742,7] HMI: Core WOF = 0x0000000000100000 recovered error:
[  477.352242181,7] HMI: PC - Thread hang recovery
```

- Add DIMM actual speed to device tree

  Recent HDAT provides DIMM actuall speed. Lets add this to device tree.

- Fix DIMM size property

  Today we parse vpd blob to get DIMM size information. This is limited to FSP based system. HDAT provides DIMM size value. Lets use that to populate device tree. So that we can get size information on BMC based system as well.

- PCI: Set slot power limit when supported

  The PCIe slot capability can be implemented in a root or switch downstream port to set the maximum power a card is allowed to draw from the system. This patch adds support for setting the power limit when the platform has defined one.

- hdata/spira: parse vpd to add part-number and serial-number to xscom@ node

  Expected by FWTS and associates our processor with the part/serial number, which is obviously a good thing for one's own sanity.

Improved HMI Handling

- opal/hmi: Add documentation for opal_handle_hmi2 call
- opal/hmi: Generate hmi event for recovered HDEC parity error.
- opal/hmi: check thread 0 tfmr to validate latched tfmr errors.

Due to P9 errata, HDEC parity and TB residue errors are latched for non-zero threads 1-3 even if they are cleared. But these are not latched on thread 0. Hence, use xscom SCOMC/SCOMD to read thread 0 tfmr value and ignore them on non-zero threads if they are not present on thread 0.

- opal/hmi: Print additional debug information in rendezvous.
- opal/hmi: Fix handling of TFMR parity/corrupt error.

  While testing TFMR parity/corrupt error it has been observed that HMIs are delivered twice for this error

  - First time HMI is delivered with HMER[4,5]=1 and TFMR[60]=1.
  - Second time HMI is delivered with HMER[4,5]=1 and TFMR[60]=0 with valid TB.

  On second HMI we end up throwing "HMI: TB invalid without core error reported" even though TB is in a valid state.

- opal/hmi: Stop flooding HMI event for TOD errors.

  Fix the issue where every thread on the chip sends HMI event to host for TOD errors. TOD errors are reported to all the core/threads on the chip. Any one thread can fix the error and send event. Rest of the threads don't need to send HMI event unnecessarily.

- opal/hmi: Fix soft lockups during TOD errors

  There are some TOD errors which do not affect working of TOD and TB. They stay in valid state. Hence we don't need rendez vous for TOD errors that does not affect TB working.

  TOD errors that affects TOD/TB will report a global error on TFMR[44] alongwith bit 51, and they will go in rendez vous path as expected.

  But the TOD errors that does not affect TB register sets only TFMR bit 51. The TFMR bit 51 is cleared when any single thread clears the TOD error. Once cleared, the bit 51 is reflected to all the cores on that chip. Any thread that reads the TFMR register after the error is cleared will see TFMR bit 51 reset. Hence the threads that see TFMR[51]=1, falls through rendez-vous path and threads that see TFMR[51]=0, returns doing nothing. This ends up in a soft lockups in host kernel.

  This patch fixes this issue by not considering TOD interrupt (TFMR[51]) as a core-global error and hence avoiding rendez-vous path completely. Instead threads that see TFMR[51]=1 will now take different path that just do the TOD error recovery.

- opal/hmi: Do not send HMI event if no errors are found.

  For TOD errors, all the cores in the chip get HMIs. Any one thread from any core can fix the issue and TFMR will have error conditions cleared. Rest of the threads need take any action if TOD errors are already cleared. Hence thread 0 of every core should get a fresh copy of TFMR before going ahead recovery path. Initialize recover = -1, so that if no errors found that thread need not send a HMI event to linux. This helps in stop flooding host with hmi event by every thread even there are no errors found.

- opal/hmi: Initialize the hmi event with old value of HMER.

  Do this before we check for TFAC errors. Otherwise the event at host console shows no error reported in HMER register.

  Without this patch the console event show HMER with all zeros

  ```
  [  216.753417] Severe Hypervisor Maintenance interrupt [Recovered]
  [  216.753498]  Error detail: Timer facility experienced an error
  [  216.753509]  HMER: 0000000000000000
  [  216.753518]  TFMR: 3c12000870e04000
  ```

  After this patch it shows old HMER values on host console:

```
[ 2237.652533] Severe Hypervisor Maintenance interrupt [Recovered]
[ 2237.652651]  Error detail: Timer facility experienced an error
[ 2237.652766]  HMER: 0840000000000000
[ 2237.652837]  TFMR: 3c12000870e04000
```

- opal/hmi: Rework HMI handling of TFAC errors

  This patch reworks the HMI handling for TFAC errors by introducing 4 rendez-vous points improve the thread synchronization while handling timebase errors that requires all thread to clear dirty data from TB/HDEC register before clearing the errors.

- opal/hmi: Don't bother passing HMER to pre-recovery cleanup

  The test for TFAC error is now redundant so we remove it and remove the HMER argument.

- opal/hmi: Move timer related error handling to a separate function

  Currently no functional change. This is a first step to completely rewriting how these things are handled.

- opal/hmi: Add a new opal_handle_hmi2 that returns direct info to Linux

  It returns a 64-bit flags mask currently set to provide info about which timer facilities were lost, and whether an event was generated.

- opal/hmi: Remove races in clearing HMER

  Writing to HMER acts as an "AND". The current code writes back the value we originally read with the bits we handled cleared. This is racy, if a new bit gets set in HW after the original read, we'll end up clearing it without handling it.

  Instead, use an all 1's mask with only the bit handled cleared.

- opal/hmi: Don't re-read HMER multiple times

  We want to make sure all reporting and actions are based upon the same snapshot of HMER in case bits get added by HW while we are in OPAL.

libflash and ffspart

Many improvements to the *ffspart* utility and *libflash* have come in this release, making *ffspart* suitable for building bit-identical PNOR images as the existing tooling used by *op-build*. The plan is to switch *op-build* to use this infrastructure in the not too distant future.

- libflash/blocklevel: Make read/write be ECC agnostic for callers

  The blocklevel abstraction allows for regions of the backing store to be marked as ECC protected so that blocklevel can decode/encode the ECC bytes into the buffer automatically without the caller having to be ECC aware.

  Unfortunately this abstraction is far from perfect, this is only useful if reads and writes are performed at the start of the ECC region or in some circumstances at an ECC aligned position - which requires the caller be aware of the ECC regions.

  The problem that has arisen is that the blocklevel abstraction is initialised somewhere but when it is later called the caller is unaware if ECC exists in the region it wants to arbitrarily read and write to. This should not have been a problem since blocklevel knows. Currently misaligned reads will fail ECC checks and misaligned writes will overwrite ECC bytes and the backing store will become corrupted.

  This patch add the smarts to blocklevel_read() and blocklevel_write() to cope with the problem. Note that ECC can always be bypassed by calling blocklevel_raw_() functions.

  All this work means that the gard tool can can safely call blocklevel_read() and blocklevel_write() and as long as the blocklevel knows of the presence of ECC then it will deal with all cases.

This also commit removes code in the gard tool which compensated for inadequacies no longer present in blocklevel.

- libflash/blocklevel: Return region start from ecc_protected()

  Currently all ecc_protected() does is say if a region is ECC protected or not. Knowing a region is ECC protected is one thing but there isn't much that can be done afterwards if this is the only known fact. A lot more can be done if the caller is told where the ECC region begins.

  Knowing where the ECC region start it allows to caller to align its read/and writes. This allows for more flexibility calling read and write without knowing exactly how the backing store is organised.

- libflash/ecc: Add helpers to align a position within an ecc buffer

  As part of ongoing work to make ECC invisible to higher levels up the stack this function converts a 'position' which should be ECC agnostic to the equivalent position within an ECC region starting at a specified location.

- libflash/ecc: Add functions to deal with unaligned ECC memcpy
- external/ffspart: Improve error output
- libffs: Fix bad checks for partition overlap

  Not all TOCs are written at zero

- libflash/libffs: Allow caller to specifiy header partition

  An FFS TOC is comprised of two parts. A small header which has a magic and very minimmal information about the TOC which will be common to all partitions, things like number of patritions, block sizes and the like. Following this small header are a series of entries. Importantly there is always an entry which encompases the TOC its self, this is usually called the 'part' partition.

  Currently libffs always assumes that the 'part' partition is at zero. While there is always a TOC and zero there doesn't actually have to be. PNORs may have multiple TOCs within them, therefore libffs needs to be flexible enough to allow callers to specify TOCs not at zero.

  The 'part' partition is otherwise a regular partition which may have flags associated with it. libffs should allow the user to set the flags for the 'part' partition.

  This patch achieves both by allowing the caller to specify the 'part' partition. The caller can not and libffs will provide a sensible default.

- libflash/libffs: Refcount ffs entries

  Currently consumers can add an new ffs entry to multiple headers, this is fine but freeing any of the headers will cause the entry to be freed, this causes double free problems.

  Even if only one header is uses, the consumer of the library still has a reference to the entry, which they may well reuse at some other point.

  libffs will now refcount entries and only free when there are no more references.

  This patch also removes the pointless return value of ffs_hdr_free()

- libflash/libffs: Switch to storing header entries in an array

  Since the libffs no longer needs to sort the entries as they get added it makes little sense to have the complexity of a linked list when an array will suffice.

- libflash/libffs: Remove backup partition from TOC generation code

  It turns out this code was messy and not all that reliable. Doing it at the library level adds complexity to the library and restrictions to the caller.

  A simpler approach can be achived with the just instantiating multiple ffs_header structures pointing to different parts of the same file.

- libflash/libffs: Remove the 'sides' from the FFS TOC generation code

  It turns out this code was messy and not all that reliable. Doing it at the library level adds complexity to the library and restrictions to the caller.

  A simpler approach can be achived with the just instantiating multiple ffs_header structures pointing to different parts of the same file.

- libflash/libffs: Always add entries to the end of the TOC

  It turns out that sorted order isn't the best idea. This removes flexibility from the caller. If the user wants their partitions in sorted order, they should insert them in sorted order.

- external/ffspart: Remove side, order and backup options

  These options are currently flakey in libflash/libffs so there isn't much point to being able to use them in ffspart.

  Future reworks planned for libflash/libffs will render these options redundant anyway.

- libflash/libffs: ffs_close() should use ffs_hdr_free()
- libflash/libffs: Add setter for a partitions actual size
- pflash: Use ffs_entry_user_to_string() to standardise flag strings
- libffs: Standardise ffs partition flags

  It seems we've developed a character respresentation for ffs partition flags. Currently only pflash really prints them so it hasn't been a problem but now ffspart wants to read them in from user input.

  It is important that what libffs reads and what pflash prints remain consistent, we should move the code into libffs to avoid problems.

- external/ffspart: Allow # comments in input file

p9dsu Platform changes

The p9dsu platform from SuperMicro (also known as 'Boston') has received a number of updates, and the patches once carried by SuperMicro are now upstream.

Since 6.0-rc1:

- p9dsu: timeout for variant detection, default to 2uess

Since 5.11:

- p9dsu: detect p9dsu variant even when hostboot doesn't tell us

  The SuperMicro BMC can tell us what riser type we have, which dictates the PCI slot tables. Usually, in an environment that a customer would experience, Hostboot will do the query with an SMC specific patch (not upstream as there's no platform specific code in hostboot) and skiboot knows what variant it is based on the compatible string.

  However, if you're using upstream hostboot, you only get the bare 'p9dsu' compatible type. We can work around this by asking the BMC ourselves and setting the slot table appropriately. We do this syncronously in platform init so that we don't start probing PCI before we setup the slot table.

- p9dsu: add slot power limit.
- p9dsu: add pci slot table for Boston LC 1U/2U and Boston LA/ESS.
- p9dsu HACK: fix system-vpd eeprom
- p9dsu: change esel command from AMI to IBM 0x3a.

ZZ Platform Changes

- hdata/i2c: Fix up pci hotplug labels

  These labels are used on the devices used to do PCIe slot power control for implementing PCIe hotplug. I'm not sure how they ended up as "eeprom-pgood" and "eeprom-controller" since that doesn't make any sense.

- hdata/i2c: Ignore multi-port I2C devices

  Recent FSP firmware builds add support for multi-port I2C devices such as the GPIO expanders used for the presence detect of OpenCAPI devices and the PCIe hotplug controllers used to power cycle PCIe slots on ZZ.

  The OpenCAPI driver inside of skiboot currently uses a platform-specific method to talk to the relevant I2C device rather than relying on HDAT since not all platforms correctly report the I2C devices (hello Zaius). Additionally the nature of multi-port devices require that we a device specific handler so that we generate the correct DT bindings. Currently we don't and there is no immediate need for this support so just ignore the multi-port devices for now.

- hdata/i2c: Replace *i2c_* prefix with *dev_*

  The current naming scheme makes it easy to conflate "i2cm_port" and "i2c_port." The latter is used to describe multi-port I2C devices such as GPIO expanders and multi-channel PCIe hotplug controllers. Rename i2c_port to dev_port to make the two a bit more distinct.

  Also rename i2c_addr to dev_addr for consistency.

- hdata/i2c: Ignore CFAM I2C master

  Recent FSP firmware builds put in information about the CFAM I2C master in addition the to host I2C masters accessible via XSCOM. Odds are this information should not be there since there's no handshaking between the FSP/BMC and the host over who controls that I2C master, but it is so we need to deal with it.

  This patch adds filtering to the HDAT parser so it ignores the CFAM I2C master. Without this it will create a bogus i2cm@<addr> which migh cause issues.

- ZZ: hw/imc: Add support to load imc catalog lid file

  Add support to load the imc catalog from a lid file packaged as part of the system firmware. Lid number allocated is 0x80f00103.lid.

Bugs Fixed

Since 6.0-rc2:

- core/opal: Fix recursion check in opal_run_pollers()

  An earlier commit introduced a counter variable poller_recursion to limit to the number number of error messages shown when opal_pollers are run recursively. However the check for the counter value was placed in a way that the poller recursion was only detected first 16 times and then allowed afterwards.

  This patch fixes this by moving the check for the counter value inside the conditional branch with some re-factoring so that opal_poller recursion is not erroneously allowed after poll_recursion is detected first 16 times.

- phb4: Print WOF registers on fence detect

  Without the WOF registers it's hard to figure out what went wrong first, so print those when we print the FIRs when a fence is detected.

- p9dsu: detect variant in init only if probe fails to found.

  Currently the slot table init happens twice in both probe and init functions due to the variant detection logic called with in-correct condition check.

Since 6.0-rc1:

- core/direct-controls: improve p9_stop_thread error handling

  p9_stop_thread should fail the operation if it finds the thread was already quiescd. This implies something else is doing direct controls on the thread (e.g., pdbg) or there is some exceptional condition we don't know how to deal with. Proceeding here would cause things to trample on each other, for example the hard lockup watchdog trying to send a sreset to the core while it is stopped for debugging with pdbg will end in tears.

  If p9_stop_thread times out waiting for the thread to quiesce, do not hit it with a core_start direct control, because we don't know what state things are in and doing more things at this point is worse than doing nothing. There is no good recipe described in the workbook to de-assert the core_stop control if it fails to quiesce the thread. After timing out here, the thread may eventually quiesce and get stuck, but that's simpler to debug than undefied behaviour.

- core/direct-controls: fix p9_cont_thread for stopped/inactive threads

  Firstly, p9_cont_thread should check that the thread actually was quiesced before it tries to resume it. Anything could happen if we try this from an arbitrary thread state.

  Then when resuming a quiesced thread that is inactive or stopped (in a stop idle state), we must not send a core_start direct control, clear_maint must be used in these cases.

- hmi: Clear unknown debug trigger

  On some systems, seeing hangs like this when Linux starts:

  ```
  [ 170.027252763,5] OCC: All Chip Rdy after 0 ms
  [ 170.062930145,5] INIT: Starting kernel at 0x20011000, fdt at 0x30ae0530 366247␣
  ↪bytes)
  [ 171.238270428,5] OPAL: Switch to little-endian OS
  ```

  If you look at the in memory skiboot console (or do *nvram -p ibm,skiboot –update-config log-level-driver=7*) we see the console get spammed with:

  ```
  [ 5209.109790675,7] HMI: Received HMI interrupt: HMER = 0x0000400000000000
  [ 5209.109792716,7] HMI: Received HMI interrupt: HMER = 0x0000400000000000
  [ 5209.109794695,7] HMI: Received HMI interrupt: HMER = 0x0000400000000000
  [ 5209.109796689,7] HMI: Received HMI interrupt: HMER = 0x0000400000000000
  ```

  We're taking the debug trigger (bit 17) early on, before the hmi_debug_trigger function in the kernel is set up.

  This clears the HMI in Skiboot and reports to the kernel instead of bringing down the machine.

- core/hmi: assign flags=0 in case nothing set by handle_hmi_exception

  Theoretically we could have returned junk to the OS in this parameter.

- SLW: Fix mambo boot to use stop states

  After commit 35c66b8ce5a2 ("SLW: Move MAMBO simulator checks to slw_init"), mambo boot no longer calls add_cpu_idle_state_properties() and as such we never enable stop states.

  After adding the call back, we get more testing coverage as well as faster mambo SMT boots.

- phb4: Hardware init updates

  CFG Write Request Timeout was incorrectly set to informational and not fatal for both non-CAPI and CAPI, so set it to fatal. This was a mistake in the specification. Correcting this fixes a niche bug in escalation (which is necessary on pre-DD2.2) that can cause a checkstop due to a NCU timeout.

  In addition, set the values in the timeout control registers to match. This fixes an extremely rare and unreproducible bug, though the current timings don't make sense since they're higher than the NCU timeout (16) which will checkstop the machine anyway.

- SLW: quieten 'Configuring self-restore' for DARN,NCU_SPEC_BAR and HRMOR

Since 5.11:

- core: Fix iteration condition to skip garded cpu
- uart: fix uart_opal_flush to take console lock over uart_con_flush This bug meant that OPAL_CONSOLE_FLUSH didn't take the appropriate locks. Luckily, since this call is only currently used in the crash path.
- xive: fix missing unlock in error path
- OPAL_PCI_SET_POWER_STATE: fix locking in error paths

  Otherwise we could exit OPAL holding locks, potentially leading to all sorts of problems later on.
- hw/slw: Don't assert on a unknown chip

  For some reason skiboot populates nodes in /cpus/ for the cores on chips that are deconfigured. As a result Linux includes the threads of those cores in it's set of possible CPUs in the system and attempts to set the SPR values that should be used when waking a thread from a deep sleep state.

  However, in the case where we have deconfigured chip we don't create a xscom node for that chip and as a result we don't have a proc_chip structure for that chip either. In turn, this results in an assertion failure when calling opal_slw_set_reg() since it expects the chip structure to exist. Fix this up and print an error instead.
- opal/hmi: Generate one event per core for processor recovery.

  Processor recovery is per core error. All threads on that core receive HMI. All threads don't need to generate HMI event for same error.

  Let thread 0 only generate the event.
- sensors: Dont add DTS sensors when OCC inband sensors are available

  There are two sets of core temperature sensors today. One is DTS scom based core temperature sensors and the second group is the sensors provided by OCC. DTS is the highest temperature among the different temperature zones in the core while OCC core temperature sensors are the average temperature of the core. DTS sensors are read directly by the host by SCOMing the DTS sensors while OCC sensors are read and updated by OCC to main memory.

  Reading DTS sensors by SCOMing is a heavy and slower operation as compared to reading OCC sensors which is as good as reading memory. So dont add DTS sensors when OCC sensors are available.
- core/fast-reboot: Increase timeout for dctl sreset to 1sec

  Direct control xscom can take more time to complete. We seem to wait too little on Boston failing fast-reboot for no good reason.

  Increase timeout to 1 sec as a reasonable value for sreset to be delivered and core to start executing instructions.
- occ: sensors-groups: Add DT properties to mark HWMON sensor groups

  Fix the sensor type to match HWMON sensor types. Add compatible flag to indicate the environmental sensor groups so that operations on these groups can be handled by HWMON linux interface.
- core: Correctly load initramfs in stb container

  Skiboot does not calculate the actual size and start location of the initramfs if it is wrapped by an STB container (for example if loading an initramfs from the ROOTFS partition).

  Check if the initramfs is in an STB container and determine the size and location correctly in the same manner as the kernel. Since load_initramfs() is called after load_kernel() move the call to trustedboot_exit_boot_services() into load_and_boot_kernel() so it is called after both of these.
- hdat/i2c.c: quieten "v2 found, parsing as v1"

- hw/imc: Check for pause_microcode_at_boot() return status

  pause_microcode_at_boot() loops through all the chip's ucode control block and pause the ucode if it is in the running state. But it does not fail if any of the chip's ucode is not initialised.

  Add code to return a failure if ucode is not initialized in any of the chip. Since pause_microcode_at_boot() is called just before attaching the IMC device nodes in imc_init(), add code to check for the function return.

Slot location code fixes:

- npu2: Use ibm, loc-code rather than ibm, slot-label

  The ibm,slot-label property is to name the slot that appears under a PCIe bridge. In the past we (ab)used the slot tables to attach names to GPU devices and their corresponding NVLinks which resulted in npu2.c using slot-label as a location code rather than as a way to name slots.

  Fix this up since it's confusing.

- hdata/slots: Apply slot label to the parent slot

  Slot names only really make sense when applied to an actual slot rather than a device. On witherspoon the GPU devices have a name associated with the device rather than the slot for the GPUs. Add a hack that moves the slot label to the parent slot rather than on the device itself.

- pci-dt-slot: Big ol' cleanup

  The underlying data that we get from HDAT can only really describe a PCIe system. As such we can simplify the devicetree slot lookup code by only caring about the important cases, namly, root ports and switch downstream ports.

  This also fixes a bug where root port didn't get a Slot label applied which results in devices under that port not having ibm,loc-code set. This results in the EEH core being unable to report the location of EEHed devices under that port.

opal-prd

- opal-prd: Insert powernv_flash module

  Explictly load powernv_flash module on BMC based system so that we are sure that flash device is created before starting opal-prd daemon.

  Note that I have replaced pnor_available() check with is_fsp_system(). As we want to load module on BMC system only. Also pnor_init has enough logic to detect flash device. Hence pnor_available() becomes redundant check.

NPU2/NVLINK2

- npu2/hw-procedures: fence bricks on GPU reset

  The NPU workbook defines a way of fencing a brick and getting the brick out of fence state. We do have an implementation of bringing the brick out of fenced/quiesced state. We do the latter in our procedures, but to support run time reset we need to do the former.

  The fencing ensures that access to memory behind the links will not lead to HMI's, but instead SUE's will be populated in cache (in the case of speculation). The expectation is then that prior to and after reset, the operating system components will flush the cache for the region of memory behind the GPU.

  This patch does the following:

  1. Implements a npu2_dev_fence_brick() function to set/clear fence state

2. Clear FIR bits prior to clearing the fence status
3. Clear's the fence status
4. We take the powerbus out of CQ fence much later now, in credits_check() which is the last hardware procedure called after link training.

- hw/npu2.c: Remove static configuration of NPU2 register

  The NPU_SM_CONFIG0 register currently needs to be configured in Skiboot to select NVLink mode, however Hostboot should configure other bits in this register.

  For some reason Skiboot was explicitly clearing bit-6 (CONFIG_DISABLE_VG_NOT_SYS). It is unclear why this bit was getting cleared as recent Hostboot versions explicitly set it to the correct value based on the specific system configuration. Therefore Skiboot should not alter it.

  Bit-58 (CONFIG_NVLINK_MODE) selects if NVLink mode should be enabled or not. Hostboot does not configure this bit so Skiboot should continue to configure it.

- npu2: Improve log output of GPU-to-link mapping

  Debugging issues related to unconnected NVLinks can be a little less irritating if we use the NPU2DEV{DBG,INF}() macros instead of prlog().

  In short, change this:

```
NPU2: comparing GPU 'GPU2' and NPU2 'GPU1'
NPU2: comparing GPU 'GPU3' and NPU2 'GPU1'
NPU2: comparing GPU 'GPU4' and NPU2 'GPU1'
NPU2: comparing GPU 'GPU5' and NPU2 'GPU1'
      :
npu2_dev_bind_pci_dev: No PCI device for NPU2 device 0006:00:01.0 to bind to. If␣
↪you expect a GPU to be there, this is a problem.
```

  to this:

```
NPU6:0:1.0 Comparing GPU 'GPU2' and NPU2 'GPU1'
NPU6:0:1.0 Comparing GPU 'GPU3' and NPU2 'GPU1'
NPU6:0:1.0 Comparing GPU 'GPU4' and NPU2 'GPU1'
NPU6:0:1.0 Comparing GPU 'GPU5' and NPU2 'GPU1'
      :
NPU6:0:1.0 No PCI device found for slot 'GPU1'
```

- npu2: Move NPU2_XTS_BDF_MAP_VALID assignment to context init

  A bad GPU or other condition may leave us with a subset of links that never get initialized. If an ATSD is sent to one of those bricks, it will never complete, leaving us waiting forever for a response:

```
watchdog: BUG: soft lockup - CPU#23 stuck for 23s! [acos:2050]
...
Modules linked in: nvidia_uvm(O) nvidia(O)
CPU: 23 PID: 2050 Comm: acos Tainted: G         W  O    4.14.0 #2
task: c0000000285cfc00 task.stack: c000001fea860000
NIP:  c0000000000abdf0 LR: c0000000000acc48 CTR: c0000000000ace60
REGS: c000001fea863550 TRAP: 0901   Tainted: G         W  O     (4.14.0)
MSR:  9000000000009033 <SF,HV,EE,ME,IR,DR,RI,LE>  CR: 28004484  XER: 20040000
CFAR: c0000000000abdf4 SOFTE: 1
GPR00: c0000000000acc48 c000001fea8637d0 c0000000011f7c00 c000001fea863820
GPR04: 0000000002000000 0004100026000000 c0000000012778c8 c00000000127a560
GPR08: 0000000000000001 0000000000000080 c000201cc7cb7750 ffffffffffffffff
GPR12: 0000000000008000 c000000003167e80
```

(continues on next page)

(continued from previous page)

```
NIP [c0000000000abdf0] mmio_invalidate_wait+0x90/0xc0
LR [c0000000000acc48] mmio_invalidate.isra.11+0x158/0x370
```

ATSDs are only sent to bricks which have a valid entry in the XTS_BDF table. So to prevent the hang, don't set NPU2_XTS_BDF_MAP_VALID unless we make it all the way to creating a context for the BDF.

Secure and Trusted Boot

- hdata/tpmrel: detect tpm not present by looking up the stinfo->status

  Skiboot detects if tpm is present by checking if a secureboot_tpm_info entry exists. However, if a tpm is not present, hostboot also creates a secureboot_tpm_info entry. In this case, hostboot creates an empty entry, but setting the field tpm_status to TPM_NOT_PRESENT.

  This detects if tpm is not present by looking up the stinfo->status.

  This fixes the "TPMREL: TPM node not found for chip_id=0 (HB bug)" issue, reproduced when skiboot is running on a system that has no tpm.

PCI

- phb4: Restore bus numbers after CRS

  Currently we restore PCIe bus numbers right after the link is up. Unfortunately as this point we haven't done CRS so config space may not be accessible.

  This moves the bus number restore till after CRS has happened.

- romulus: Add a barebones slot table
- phb4: Quieten and improve "Timeout waiting for electrical link"

  This happens normally if a slot doesn't have a working HW presence detect and relies instead of inband presence detect.

  The message we display is scary and not very useful unless ou are debugging, so quiten it up and change it to something more meaningful.

- pcie-slot: Don't fail powering on an already on switch

  If the power state is already the required value, return OPAL_SUCCESS rather than OPAL_PARAMETER to avoid spurrious errors during boot.

CAPI/OpenCAPI

- capi: Keep the current mmio windows in the mbt cache table.

  When the phb is used as a CAPI interface, the current mmio windows list is cleaned before adding the capi and the prefetchable memory (M64) windows, which implies that the non-prefetchable BAR is no more configured. This patch allows to set only the mbt bar to pass capi mmio window and to keep, as defined, the other mmio values (M32 and M64).

- npu2-opencapi: Fix 'link internal error' FIR, take 2

  When setting up an opencapi link, we set the transport muxes first, then set the PHY training config register, which includes disabling nvlink mode for the bricks. That's the order of the init sequence, as found in the NPU workbook.

In reality, doing so works, but it raises 2 FIR bits in the PowerBus OLL FIR Register for the 2 links when we configure the transport muxes. Presumably because nvlink is not disabled yet and we are configuring the transport muxes for opencapi.

bit 60: link0 internal error

bit 61: link1 internal error

Overall the current setup ends up being correct and everything works, but we raise 2 FIR bits.

So tweak the order of operations to disable nvlink before configuring the transport muxes. Incidentally, this is what the scripts from the opencapi enablement team were doing all along.

- npu2-opencapi: Fix 'link internal error' FIR, take 1

  When we setup a link, we always enable ODL0 and ODL1 at the same time in the PHY training config register, even though we are setting up only one OTL/ODL, so it raises a "link internal error" FIR bit in the PowerBus OLL FIR Register for the second link. The error is harmless, as we'll eventually setup the second link, but there's no reason to raise that FIR bit.

  The fix is simply to only enable the ODL we are using for the link.

- phb4: Do not set the PBCQ Tunnel BAR register when enabling capi mode.

  The cxl driver will set the capi value, like other drivers already do.

- phb4: set TVT1 for tunneled operations in capi mode

  The ASN indication is used for tunneled operations (as_notify and atomics). Tunneled operation messages can be sent in PCI mode as well as CAPI mode.

  The address field of as_notify messages is hijacked to encode the LPID/PID/TID of the target thread, so those messages should not go through address translation. Therefore bit 59 is part of the ASN indication.

  This patch sets TVT#1 in bypass mode when capi mode is enabled, to prevent as_notify messages from being dropped.

Debugging/Testing improvements

Since 6.0-rc1:

- mambo: Enable XER CA32 and OV32 bits on P9

  POWER9 adds 32 bit carry and overflow bits to the XER, but we need to set the relevant CTRL1 bit to enable them.

- Makefile: Fix building natively on ppc64le

  When on ppc64le and CROSS is not set by the environment, make assumes ppc64 and sets a default CROSS. Check for ppc64le as well, so that 'make' works out of the box on ppc64le.

- Experimental support for building with Clang
- Improvements to testing and Travis CI

Since 5.11:

- core/stack: backtrace unwind basic OPAL call details

  Put OPAL callers' r1 into the stack back chain, and then use that to unwind back to the OPAL entry frame (as opposed to boot entry, which has a 0 back chain).

  From there, dump the OPAL call token and the caller's r1. A backtrace looks like this:

```
CPU 0000 Backtrace:
 S: 0000000031c03ba0 R: 000000003001a548   ._abort+0x4c
 S: 0000000031c03c20 R: 000000003001baac   .opal_run_pollers+0x3c
 S: 0000000031c03ca0 R: 000000003001bcbc   .opal_poll_events+0xc4
 S: 0000000031c03d20 R: 00000000300051dc   opal_entry+0x12c
 --- OPAL call entry token: 0xa caller R1: 0xc0000000006d3b90 ---
```

This is pretty basic for the moment, but it does give you the bottom of the Linux stack. It will allow some interesting improvements in future.

First, with the eframe, all the call's parameters can be printed out as well. The ___backtrace / ___print_backtrace API needs to be reworked in order to support this, but it's otherwise very simple (see opal_trace_entry()).

Second, it will allow Linux's stack to be passed back to Linux via a debugging opal call. This will allow Linux's BUG() or xmon to also print the Linux back trace in case of a NMI or MCE or watchdog lockup that hits in OPAL.

- asm/head: implement quiescing without stack or clobbering regs

  Quiescing currently is implmeented in C in opal_entry before the opal call handler is called. This works well enough for simple cases like fast reset when one CPU wants all others out of the way.

  Linux would like to use it to prevent an sreset IPI from interrupting firmware, which could lead to deadlocks when crash dumping or entering the debugger. Linux interrupts do not recover well when returning back to general OPAL code, due to r13 not being restored. OPAL also can't be re-entered, which may happen e.g., from the debugger.

  So move the quiesce hold/reject to entry code, beore the stack or r1 or r13 registers are switched. OPAL can be interrupted and returned to or re-entered during this period.

  This does not completely solve all such problems. OPAL will be interrupted with sreset if the quiesce times out, and it can be interrupted by MCEs as well. These still have the issues above.

- core/opal: Allow poller re-entry if OPAL was re-entered

  If an NMI interrupts the middle of running pollers and the OS invokes pollers again (e.g., for console output), the poller re-entrancy check will prevent it from running and spam the console.

  That check was designed to catch a poller calling opal_run_pollers, OPAL re-entrancy is something different and is detected elsewhere. Avoid the poller recursion check if OPAL has been re-entered. This is a best-effort attempt to cope with errors.

- core/opal: Emergency stack for re-entry

  This detects OPAL being re-entered by the OS, and switches to an emergency stack if it was. This protects the firmware's main stack from re-entrancy and allows the OS to use NMI facilities for crash / debug functionality.

  Further nested re-entry will destroy the previous emergency stack and prevent returning, but those should be rare cases.

  This stack is sized at 16kB, which doubles the size of CPU stacks, so as not to introduce a regression in primary stack size. The 16kB stack originally had a 4kB machine check stack at the top, which was removed by 80eee1946 ("opal: Remove machine check interrupt patching in OPAL."). So it is possible the size could be tightened again, but that would require further analysis.

- hdat_to_dt: hash_prop the same on all platforms Fixes this unit test on ppc64le hosts.
- mambo: Add persistent memory disk support

  This adds support to for mapping disks images using persistent memory. Disks can be added by setting this ENV variable:

      PMEM_DISK="/mydisks/disk1.img,/mydisks/disk2.img"

These will show up in Linux as /dev/pmem0 and /dev/pmem1.

This uses a new feature in mambo "mysim memory mmap .." which is only available since mambo commit 0131f0fc08 (from 24/4/2018).

This also needs the of_pmem.c driver in Linux which is only available since v4.17. It works with powernv_defconfig + CONFIG_OF_PMEM.

- external/mambo: Add di command to decode instructions

  By default you get 16 instructions but you can specify the number you want. i.e.

```
systemsim % di 0x100 4
0x0000000000000100: Enc:0xA64BB17D : mtspr   HSPRG1,r13
0x0000000000000104: Enc:0xA64AB07D : mfspr   r13,HSPRG0
0x0000000000000108: Enc:0xF0092DF9 : std     r9,0x9F0(r13)
0x000000000000010C: Enc:0xA6E2207D : mfspr   r9,PPR
```

  Using di since it's what xmon uses.

- mambo/mambo_utils.tcl: Inject an MCE at a specified address

  Currently we don't support injecting an MCE on a specific address. This is useful for testing functionality like memcpy_mcsafe() (see https://patchwork.ozlabs.org/cover/893339/)

  The core of the functionality is a routine called inject_mce_ue_on_addr, which takes an addr argument and injects an MCE (load/store with UE) when the specified address is accessed by code. This functionality can easily be enhanced to cover instruction UE's as well.

  A sample use case to create an MCE on stack access would be

```
set addr [mysim display gpr 1]
inject_mce_ue_on_addr $addr
```

  This would cause an mce on any r1 or r1 based access

- external/mambo: improve helper for machine checks

  Improve workarounds for stop injection, because mambo often will trigger on 0x104/204 when injecting sreset/mces.

  This also adds a workaround to skip injecting on reservations to avoid infinite loops when doing inject_mce_step.

- travis: Enable ppc64le builds

  At least on the IBM Travis Enterprise instance, we can now do ppc64le builds!

  We can only build a subset of our matrix due to availability of ppc64le distros. The Dockerfiles need some tweaking to only attempt to install (x86_64 only) Mambo binaries, as well as the build scripts.

- external: Add "lpc" tool

  This is a little front-end to the lpc debugfs files to access the LPC bus from userspace on the host.

- core/test/run-trace: fix on ppc64el

5.1.103 skiboot-6.0-rc1

skiboot v6.0-rc1 was released on Tuesday May 1st 2018. It is the first release candidate of skiboot 6.0, which will become the new stable release of skiboot following the 5.11 release, first released April 6th 2018.

Skiboot 6.0 will mark the basis for op-build v2.0 and will be required for POWER9 systems.

skiboot v6.0-rc1 contains all bug fixes as of *skiboot-5.11*, *skiboot-5.10.5*, and *skiboot-5.4.9* (the currently maintained stable releases). Once 6.0 is released, we do *not* expect any further stable releases in the 5.10.x series, nor in the 5.11.x series.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 6.0 in early May, with skiboot 6.0 being for all POWER8 and POWER9 platforms in op-build v2.0.

Over skiboot-5.11, we have the following changes:

New Features

- Disable stop states from OPAL

  On ZZ, stop4,5,11 are enabled for PowerVM, even though doing so may cause problems with OPAL due to bugs in hcode.

  For other platforms, this isn't so much of an issue as we can just control stop states by the MRW. However the rebuild-the-world approach to changing values there is a bit annoying if you just want to rule out a specific stop state from being problematic.

  Provide an nvram option to override what's disabled in OPAL.

  The OPAL mask is currently ~0xE0000000 (i.e. all but stop 0,1,2)

  You can set an NVRAM override with:

  ```
  nvram -p ibm,skiboot --update-config opal-stop-state-disable-mask=0xFFFFFFF
  ```

  This nvram override will disable *all* stop states.

- interrupts: Create an "interrupts" property in the OPAL node

  Deprecate the old "opal-interrupts", it's still there, but the new property follows the standard and allow us to specify whether an interrupt is level or edge sensitive.

  Similarly create "interrupt-names" whose content is identical to "opal-interrupts-names".

- SBE: Add timer support on POWER9

  SBE on P9 provides one shot programmable timer facility. We can use this to implement OPAL timers and hence limit the reliance on the Linux heartbeat (similar to HW timer facility provided by SLW on P8).

- Add SBE driver support

  SBE (Self Boot Engine) on P9 has two different jobs: - Boot the chip up to the point the core is functional - Provide various services like timer, scom, stash MPIPL, etc., at runtime

  We will use SBE for various purposes like timer, MPIPL, etc.

- opal:hmi: Add missing processor recovery reason string.

  With this patch now we see reason string printed for CORE_WOF[43] bit.

  ```
  [  477.352234986,7] HMI: [Loc: U78D3.001.WZS004A-P1-C48]: P:8 C:22 T:3: Processor␣
  ↪recovery occurred.
  [  477.352240742,7] HMI: Core WOF = 0x0000000000100000 recovered error:
  [  477.352242181,7] HMI: PC - Thread hang recovery
  ```

- Add DIMM actual speed to device tree

  Recent HDAT provides DIMM actuall speed. Lets add this to device tree.

- Fix DIMM size property

  Today we parse vpd blob to get DIMM size information. This is limited to FSP based system. HDAT provides DIMM size value. Lets use that to populate device tree. So that we can get size information on BMC based system as well.

- PCI: Set slot power limit when supported

  The PCIe slot capability can be implemented in a root or switch downstream port to set the maximum power a card is allowed to draw from the system. This patch adds support for setting the power limit when the platform has defined one.

- hdata/spira: parse vpd to add part-number and serial-number to xscom@ node

  Expected by FWTS and associates our processor with the part/serial number, which is obviously a good thing for one's own sanity.

Improved HMI Handling

- opal/hmi: Add documentation for opal_handle_hmi2 call
- opal/hmi: Generate hmi event for recovered HDEC parity error.
- opal/hmi: check thread 0 tfmr to validate latched tfmr errors.

  Due to P9 errata, HDEC parity and TB residue errors are latched for non-zero threads 1-3 even if they are cleared. But these are not latched on thread 0. Hence, use xscom SCOMC/SCOMD to read thread 0 tfmr value and ignore them on non-zero threads if they are not present on thread 0.

- opal/hmi: Print additional debug information in rendezvous.
- opal/hmi: Fix handling of TFMR parity/corrupt error.

  While testing TFMR parity/corrupt error it has been observed that HMIs are delivered twice for this error

  - First time HMI is delivered with HMER[4,5]=1 and TFMR[60]=1.
  - Second time HMI is delivered with HMER[4,5]=1 and TFMR[60]=0 with valid TB.

  On second HMI we end up throwing "HMI: TB invalid without core error reported" even though TB is in a valid state.

- opal/hmi: Stop flooding HMI event for TOD errors.

  Fix the issue where every thread on the chip sends HMI event to host for TOD errors. TOD errors are reported to all the core/threads on the chip. Any one thread can fix the error and send event. Rest of the threads don't need to send HMI event unnecessarily.

- opal/hmi: Fix soft lockups during TOD errors

  There are some TOD errors which do not affect working of TOD and TB. They stay in valid state. Hence we don't need rendez vous for TOD errors that does not affect TB working.

  TOD errors that affects TOD/TB will report a global error on TFMR[44] alongwith bit 51, and they will go in rendez vous path as expected.

  But the TOD errors that does not affect TB register sets only TFMR bit 51. The TFMR bit 51 is cleared when any single thread clears the TOD error. Once cleared, the bit 51 is reflected to all the cores on that chip. Any thread that reads the TFMR register after the error is cleared will see TFMR bit 51 reset. Hence the threads that see TFMR[51]=1, falls through rendez-vous path and threads that see TFMR[51]=0, returns doing nothing. This ends up in a soft lockups in host kernel.

This patch fixes this issue by not considering TOD interrupt (TFMR[51]) as a core-global error and hence avoiding rendez-vous path completely. Instead threads that see TFMR[51]=1 will now take different path that just do the TOD error recovery.

- opal/hmi: Do not send HMI event if no errors are found.

  For TOD errors, all the cores in the chip get HMIs. Any one thread from any core can fix the issue and TFMR will have error conditions cleared. Rest of the threads need take any action if TOD errors are already cleared. Hence thread 0 of every core should get a fresh copy of TFMR before going ahead recovery path. Initialize recover = -1, so that if no errors found that thread need not send a HMI event to linux. This helps in stop flooding host with hmi event by every thread even there are no errors found.

- opal/hmi: Initialize the hmi event with old value of HMER.

  Do this before we check for TFAC errors. Otherwise the event at host console shows no error reported in HMER register.

  Without this patch the console event show HMER with all zeros

```
[  216.753417] Severe Hypervisor Maintenance interrupt [Recovered]
[  216.753498]  Error detail: Timer facility experienced an error
[  216.753509]  HMER: 0000000000000000
[  216.753518]  TFMR: 3c12000870e04000
```

  After this patch it shows old HMER values on host console:

```
[ 2237.652533] Severe Hypervisor Maintenance interrupt [Recovered]
[ 2237.652651]  Error detail: Timer facility experienced an error
[ 2237.652766]  HMER: 0840000000000000
[ 2237.652837]  TFMR: 3c12000870e04000
```

- opal/hmi: Rework HMI handling of TFAC errors

  This patch reworks the HMI handling for TFAC errors by introducing 4 rendez-vous points improve the thread synchronization while handling timebase errors that requires all thread to clear dirty data from TB/HDEC register before clearing the errors.

- opal/hmi: Don't bother passing HMER to pre-recovery cleanup

  The test for TFAC error is now redundant so we remove it and remove the HMER argument.

- opal/hmi: Move timer related error handling to a separate function

  Currently no functional change. This is a first step to completely rewriting how these things are handled.

- opal/hmi: Add a new opal_handle_hmi2 that returns direct info to Linux

  It returns a 64-bit flags mask currently set to provide info about which timer facilities were lost, and whether an event was generated.

- opal/hmi: Remove races in clearing HMER

  Writing to HMER acts as an "AND". The current code writes back the value we originally read with the bits we handled cleared. This is racy, if a new bit gets set in HW after the original read, we'll end up clearing it without handling it.

  Instead, use an all 1's mask with only the bit handled cleared.

- opal/hmi: Don't re-read HMER multiple times

  We want to make sure all reporting and actions are based upon the same snapshot of HMER in case bits get added by HW while we are in OPAL.

libflash and ffspart

Many improvements to the *ffspart* utility and *libflash* have come in this release, making *ffspart* suitable for building bit-identical PNOR images as the existing tooling used by *op-build*. The plan is to switch *op-build* to use this infrastructure in the not too distant future.

- libflash/blocklevel: Make read/write be ECC agnostic for callers

  The blocklevel abstraction allows for regions of the backing store to be marked as ECC protected so that blocklevel can decode/encode the ECC bytes into the buffer automatically without the caller having to be ECC aware.

  Unfortunately this abstraction is far from perfect, this is only useful if reads and writes are performed at the start of the ECC region or in some circumstances at an ECC aligned position - which requires the caller be aware of the ECC regions.

  The problem that has arisen is that the blocklevel abstraction is initialised somewhere but when it is later called the caller is unaware if ECC exists in the region it wants to arbitrarily read and write to. This should not have been a problem since blocklevel knows. Currently misaligned reads will fail ECC checks and misaligned writes will overwrite ECC bytes and the backing store will become corrupted.

  This patch add the smarts to blocklevel_read() and blocklevel_write() to cope with the problem. Note that ECC can always be bypassed by calling blocklevel_raw_() functions.

  All this work means that the gard tool can can safely call blocklevel_read() and blocklevel_write() and as long as the blocklevel knows of the presence of ECC then it will deal with all cases.

  This also commit removes code in the gard tool which compensated for inadequacies no longer present in blocklevel.

- libflash/blocklevel: Return region start from ecc_protected()

  Currently all ecc_protected() does is say if a region is ECC protected or not. Knowing a region is ECC protected is one thing but there isn't much that can be done afterwards if this is the only known fact. A lot more can be done if the caller is told where the ECC region begins.

  Knowing where the ECC region start it allows to caller to align its read/and writes. This allows for more flexibility calling read and write without knowing exactly how the backing store is organised.

- libflash/ecc: Add helpers to align a position within an ecc buffer

  As part of ongoing work to make ECC invisible to higher levels up the stack this function converts a 'position' which should be ECC agnostic to the equivalent position within an ECC region starting at a specified location.

- libflash/ecc: Add functions to deal with unaligned ECC memcpy
- external/ffspart: Improve error output
- libffs: Fix bad checks for partition overlap

  Not all TOCs are written at zero

- libflash/libffs: Allow caller to specifiy header partition

  An FFS TOC is comprised of two parts. A small header which has a magic and very minimmal information about the TOC which will be common to all partitions, things like number of patritions, block sizes and the like. Following this small header are a series of entries. Importantly there is always an entry which encompases the TOC its self, this is usually called the 'part' partition.

  Currently libffs always assumes that the 'part' partition is at zero. While there is always a TOC and zero there doesn't actually have to be. PNORs may have multiple TOCs within them, therefore libffs needs to be flexible enough to allow callers to specify TOCs not at zero.

  The 'part' partition is otherwise a regular partition which may have flags associated with it. libffs should allow the user to set the flags for the 'part' partition.

This patch achieves both by allowing the caller to specify the 'part' partition. The caller can not and libffs will provide a sensible default.

- libflash/libffs: Refcount ffs entries

  Currently consumers can add an new ffs entry to multiple headers, this is fine but freeing any of the headers will cause the entry to be freed, this causes double free problems.

  Even if only one header is uses, the consumer of the library still has a reference to the entry, which they may well reuse at some other point.

  libffs will now refcount entries and only free when there are no more references.

  This patch also removes the pointless return value of ffs_hdr_free()

- libflash/libffs: Switch to storing header entries in an array

  Since the libffs no longer needs to sort the entries as they get added it makes little sense to have the complexity of a linked list when an array will suffice.

- libflash/libffs: Remove backup partition from TOC generation code

  It turns out this code was messy and not all that reliable. Doing it at the library level adds complexity to the library and restrictions to the caller.

  A simpler approach can be achived with the just instantiating multiple ffs_header structures pointing to different parts of the same file.

- libflash/libffs: Remove the 'sides' from the FFS TOC generation code

  It turns out this code was messy and not all that reliable. Doing it at the library level adds complexity to the library and restrictions to the caller.

  A simpler approach can be achived with the just instantiating multiple ffs_header structures pointing to different parts of the same file.

- libflash/libffs: Always add entries to the end of the TOC

  It turns out that sorted order isn't the best idea. This removes flexibility from the caller. If the user wants their partitions in sorted order, they should insert them in sorted order.

- external/ffspart: Remove side, order and backup options

  These options are currently flakey in libflash/libffs so there isn't much point to being able to use them in ffspart.

  Future reworks planned for libflash/libffs will render these options redundant anyway.

- libflash/libffs: ffs_close() should use ffs_hdr_free()
- libflash/libffs: Add setter for a partitions actual size
- pflash: Use ffs_entry_user_to_string() to standardise flag strings
- libffs: Standardise ffs partition flags

  It seems we've developed a character respresentation for ffs partition flags. Currently only pflash really prints them so it hasn't been a problem but now ffspart wants to read them in from user input.

  It is important that what libffs reads and what pflash prints remain consistent, we should move the code into libffs to avoid problems.

- external/ffspart: Allow # comments in input file

p9dsu Platform changes

The p9dsu platform from SuperMicro (also known as ‘Boston’) has received a number of updates, and the patches once carried by SuperMicro are now upstream.

- p9dsu: detect p9dsu variant even when hostboot doesn’t tell us

  The SuperMicro BMC can tell us what riser type we have, which dictates the PCI slot tables. Usually, in an environment that a customer would experience, Hostboot will do the query with an SMC specific patch (not upstream as there’s no platform specific code in hostboot) and skiboot knows what variant it is based on the compatible string.

  However, if you’re using upstream hostboot, you only get the bare ‘p9dsu’ compatible type. We can work around this by asking the BMC ourselves and setting the slot table appropriately. We do this syncronously in platform init so that we don’t start probing PCI before we setup the slot table.
- p9dsu: add slot power limit.
- p9dsu: add pci slot table for Boston LC 1U/2U and Boston LA/ESS.
- p9dsu HACK: fix system-vpd eeprom
- p9dsu: change esel command from AMI to IBM 0x3a.

ZZ Platform Changes

- hdata/i2c: Fix up pci hotplug labels

  These labels are used on the devices used to do PCIe slot power control for implementing PCIe hotplug. I’m not sure how they ended up as “eeprom-pgood” and “eeprom-controller” since that doesn’t make any sense.
- hdata/i2c: Ignore multi-port I2C devices

  Recent FSP firmware builds add support for multi-port I2C devices such as the GPIO expanders used for the presence detect of OpenCAPI devices and the PCIe hotplug controllers used to power cycle PCIe slots on ZZ.

  The OpenCAPI driver inside of skiboot currently uses a platform-specific method to talk to the relevant I2C device rather than relying on HDAT since not all platforms correctly report the I2C devices (hello Zaius). Additionally the nature of multi-port devices require that we a device specific handler so that we generate the correct DT bindings. Currently we don’t and there is no immediate need for this support so just ignore the multi-port devices for now.
- hdata/i2c: Replace *i2c_* prefix with *dev_*

  The current naming scheme makes it easy to conflate “i2cm_port” and “i2c_port.” The latter is used to describe multi-port I2C devices such as GPIO expanders and multi-channel PCIe hotplug controllers. Rename i2c_port to dev_port to make the two a bit more distinct.

  Also rename i2c_addr to dev_addr for consistency.
- hdata/i2c: Ignore CFAM I2C master

  Recent FSP firmware builds put in information about the CFAM I2C master in addition the to host I2C masters accessible via XSCOM. Odds are this information should not be there since there’s no handshaking between the FSP/BMC and the host over who controls that I2C master, but it is so we need to deal with it.

  This patch adds filtering to the HDAT parser so it ignores the CFAM I2C master. Without this it will create a bogus i2cm@<addr> which migh cause issues.
- ZZ: hw/imc: Add support to load imc catalog lid file

  Add support to load the imc catalog from a lid file packaged as part of the system firmware. Lid number allocated is 0x80f00103.lid.

Bugs Fixed

- core: Fix iteration condition to skip garded cpu
- uart: fix uart_opal_flush to take console lock over uart_con_flush This bug meant that OPAL_CONSOLE_FLUSH didn't take the appropriate locks. Luckily, since this call is only currently used in the crash path.
- xive: fix missing unlock in error path
- OPAL_PCI_SET_POWER_STATE: fix locking in error paths

  Otherwise we could exit OPAL holding locks, potentially leading to all sorts of problems later on.
- hw/slw: Don't assert on a unknown chip

  For some reason skiboot populates nodes in /cpus/ for the cores on chips that are deconfigured. As a result Linux includes the threads of those cores in it's set of possible CPUs in the system and attempts to set the SPR values that should be used when waking a thread from a deep sleep state.

  However, in the case where we have deconfigured chip we don't create a xscom node for that chip and as a result we don't have a proc_chip structure for that chip either. In turn, this results in an assertion failure when calling opal_slw_set_reg() since it expects the chip structure to exist. Fix this up and print an error instead.
- opal/hmi: Generate one event per core for processor recovery.

  Processor recovery is per core error. All threads on that core receive HMI. All threads don't need to generate HMI event for same error.

  Let thread 0 only generate the event.
- sensors: Dont add DTS sensors when OCC inband sensors are available

  There are two sets of core temperature sensors today. One is DTS scom based core temperature sensors and the second group is the sensors provided by OCC. DTS is the highest temperature among the different temperature zones in the core while OCC core temperature sensors are the average temperature of the core. DTS sensors are read directly by the host by SCOMing the DTS sensors while OCC sensors are read and updated by OCC to main memory.

  Reading DTS sensors by SCOMing is a heavy and slower operation as compared to reading OCC sensors which is as good as reading memory. So dont add DTS sensors when OCC sensors are available.
- core/fast-reboot: Increase timeout for dctl sreset to 1sec

  Direct control xscom can take more time to complete. We seem to wait too little on Boston failing fast-reboot for no good reason.

  Increase timeout to 1 sec as a reasonable value for sreset to be delivered and core to start executing instructions.
- occ: sensors-groups: Add DT properties to mark HWMON sensor groups

  Fix the sensor type to match HWMON sensor types. Add compatible flag to indicate the environmental sensor groups so that operations on these groups can be handled by HWMON linux interface.
- core: Correctly load initramfs in stb container

  Skiboot does not calculate the actual size and start location of the initramfs if it is wrapped by an STB container (for example if loading an initramfs from the ROOTFS partition).

  Check if the initramfs is in an STB container and determine the size and location correctly in the same manner as the kernel. Since load_initramfs() is called after load_kernel() move the call to trustedboot_exit_boot_services() into load_and_boot_kernel() so it is called after both of these.
- hdat/i2c.c: quieten "v2 found, parsing as v1"

- hw/imc: Check for pause_microcode_at_boot() return status

  pause_microcode_at_boot() loops through all the chip's ucode control block and pause the ucode if it is in the running state. But it does not fail if any of the chip's ucode is not initialised.

  Add code to return a failure if ucode is not initialized in any of the chip. Since pause_microcode_at_boot() is called just before attaching the IMC device nodes in imc_init(), add code to check for the function return.

Slot location code fixes:

- npu2: Use ibm, loc-code rather than ibm, slot-label

  The ibm,slot-label property is to name the slot that appears under a PCIe bridge. In the past we (ab)used the slot tables to attach names to GPU devices and their corresponding NVLinks which resulted in npu2.c using slot-label as a location code rather than as a way to name slots.

  Fix this up since it's confusing.

- hdata/slots: Apply slot label to the parent slot

  Slot names only really make sense when applied to an actual slot rather than a device. On witherspoon the GPU devices have a name associated with the device rather than the slot for the GPUs. Add a hack that moves the slot label to the parent slot rather than on the device itself.

- pci-dt-slot: Big ol' cleanup

  The underlying data that we get from HDAT can only really describe a PCIe system. As such we can simplify the devicetree slot lookup code by only caring about the important cases, namly, root ports and switch downstream ports.

  This also fixes a bug where root port didn't get a Slot label applied which results in devices under that port not having ibm,loc-code set. This results in the EEH core being unable to report the location of EEHed devices under that port.

opal-prd

- opal-prd: Insert powernv_flash module

  Explictly load powernv_flash module on BMC based system so that we are sure that flash device is created before starting opal-prd daemon.

  Note that I have replaced pnor_available() check with is_fsp_system(). As we want to load module on BMC system only. Also pnor_init has enough logic to detect flash device. Hence pnor_available() becomes redundant check.

NPU2/NVLINK2

- npu2/hw-procedures: fence bricks on GPU reset

  The NPU workbook defines a way of fencing a brick and getting the brick out of fence state. We do have an implementation of bringing the brick out of fenced/quiesced state. We do the latter in our procedures, but to support run time reset we need to do the former.

  The fencing ensures that access to memory behind the links will not lead to HMI's, but instead SUE's will be populated in cache (in the case of speculation). The expectation is then that prior to and after reset, the operating system components will flush the cache for the region of memory behind the GPU.

  This patch does the following:

  1. Implements a npu2_dev_fence_brick() function to set/clear fence state

2. Clear FIR bits prior to clearing the fence status
3. Clear's the fence status
4. We take the powerbus out of CQ fence much later now, in credits_check() which is the last hardware procedure called after link training.

- hw/npu2.c: Remove static configuration of NPU2 register

  The NPU_SM_CONFIG0 register currently needs to be configured in Skiboot to select NVLink mode, however Hostboot should configure other bits in this register.

  For some reason Skiboot was explicitly clearing bit-6 (CONFIG_DISABLE_VG_NOT_SYS). It is unclear why this bit was getting cleared as recent Hostboot versions explicitly set it to the correct value based on the specific system configuration. Therefore Skiboot should not alter it.

  Bit-58 (CONFIG_NVLINK_MODE) selects if NVLink mode should be enabled or not. Hostboot does not configure this bit so Skiboot should continue to configure it.

- npu2: Improve log output of GPU-to-link mapping

  Debugging issues related to unconnected NVLinks can be a little less irritating if we use the NPU2DEV{DBG,INF}() macros instead of prlog().

  In short, change this:

```
NPU2: comparing GPU 'GPU2' and NPU2 'GPU1'
NPU2: comparing GPU 'GPU3' and NPU2 'GPU1'
NPU2: comparing GPU 'GPU4' and NPU2 'GPU1'
NPU2: comparing GPU 'GPU5' and NPU2 'GPU1'
      :
npu2_dev_bind_pci_dev: No PCI device for NPU2 device 0006:00:01.0 to bind to. If␣
↪you expect a GPU to be there, this is a problem.
```

  to this:

```
NPU6:0:1.0 Comparing GPU 'GPU2' and NPU2 'GPU1'
NPU6:0:1.0 Comparing GPU 'GPU3' and NPU2 'GPU1'
NPU6:0:1.0 Comparing GPU 'GPU4' and NPU2 'GPU1'
NPU6:0:1.0 Comparing GPU 'GPU5' and NPU2 'GPU1'
      :
NPU6:0:1.0 No PCI device found for slot 'GPU1'
```

- npu2: Move NPU2_XTS_BDF_MAP_VALID assignment to context init

  A bad GPU or other condition may leave us with a subset of links that never get initialized. If an ATSD is sent to one of those bricks, it will never complete, leaving us waiting forever for a response:

```
watchdog: BUG: soft lockup - CPU#23 stuck for 23s! [acos:2050]
...
Modules linked in: nvidia_uvm(O) nvidia(O)
CPU: 23 PID: 2050 Comm: acos Tainted: G         W  O    4.14.0 #2
task: c0000000285cfc00 task.stack: c000001fea860000
NIP:  c0000000000abdf0 LR: c0000000000acc48 CTR: c0000000000ace60
REGS: c000001fea863550 TRAP: 0901   Tainted: G         W  O     (4.14.0)
MSR:  9000000000009033 <SF,HV,EE,ME,IR,DR,RI,LE>  CR: 28004484  XER: 20040000
CFAR: c0000000000abdf4 SOFTE: 1
GPR00: c0000000000acc48 c000001fea8637d0 c0000000011f7c00 c000001fea863820
GPR04: 0000000002000000 0004100026000000 c0000000012778c8 c00000000127a560
GPR08: 0000000000000001 0000000000000080 c000201cc7cb7750 ffffffffffffffff
GPR12: 0000000000008000 c000000003167e80
```

(continues on next page)

(continued from previous page)

```
NIP [c0000000000abdf0] mmio_invalidate_wait+0x90/0xc0
LR [c0000000000acc48] mmio_invalidate.isra.11+0x158/0x370
```

ATSDs are only sent to bricks which have a valid entry in the XTS_BDF table. So to prevent the hang, don't set NPU2_XTS_BDF_MAP_VALID unless we make it all the way to creating a context for the BDF.

Secure and Trusted Boot

- hdata/tpmrel: detect tpm not present by looking up the stinfo->status

  Skiboot detects if tpm is present by checking if a secureboot_tpm_info entry exists. However, if a tpm is not present, hostboot also creates a secureboot_tpm_info entry. In this case, hostboot creates an empty entry, but setting the field tpm_status to TPM_NOT_PRESENT.

  This detects if tpm is not present by looking up the stinfo->status.

  This fixes the "TPMREL: TPM node not found for chip_id=0 (HB bug)" issue, reproduced when skiboot is running on a system that has no tpm.

PCI

- phb4: Restore bus numbers after CRS

  Currently we restore PCIe bus numbers right after the link is up. Unfortunately as this point we haven't done CRS so config space may not be accessible.

  This moves the bus number restore till after CRS has happened.

- romulus: Add a barebones slot table
- phb4: Quieten and improve "Timeout waiting for electrical link"

  This happens normally if a slot doesn't have a working HW presence detect and relies instead of inband presence detect.

  The message we display is scary and not very useful unless ou are debugging, so quiten it up and change it to something more meaningful.

- pcie-slot: Don't fail powering on an already on switch

  If the power state is already the required value, return OPAL_SUCCESS rather than OPAL_PARAMETER to avoid spurrious errors during boot.

CAPI/OpenCAPI

- capi: Keep the current mmio windows in the mbt cache table.

  When the phb is used as a CAPI interface, the current mmio windows list is cleaned before adding the capi and the prefetchable memory (M64) windows, which implies that the non-prefetchable BAR is no more configured. This patch allows to set only the mbt bar to pass capi mmio window and to keep, as defined, the other mmio values (M32 and M64).

- npu2-opencapi: Fix 'link internal error' FIR, take 2

  When setting up an opencapi link, we set the transport muxes first, then set the PHY training config register, which includes disabling nvlink mode for the bricks. That's the order of the init sequence, as found in the NPU workbook.

In reality, doing so works, but it raises 2 FIR bits in the PowerBus OLL FIR Register for the 2 links when we configure the transport muxes. Presumably because nvlink is not disabled yet and we are configuring the transport muxes for opencapi.

bit 60: link0 internal error

bit 61: link1 internal error

Overall the current setup ends up being correct and everything works, but we raise 2 FIR bits.

So tweak the order of operations to disable nvlink before configuring the transport muxes. Incidentally, this is what the scripts from the opencapi enablement team were doing all along.

- npu2-opencapi: Fix 'link internal error' FIR, take 1

  When we setup a link, we always enable ODL0 and ODL1 at the same time in the PHY training config register, even though we are setting up only one OTL/ODL, so it raises a "link internal error" FIR bit in the PowerBus OLL FIR Register for the second link. The error is harmless, as we'll eventually setup the second link, but there's no reason to raise that FIR bit.

  The fix is simply to only enable the ODL we are using for the link.

- phb4: Do not set the PBCQ Tunnel BAR register when enabling capi mode.

  The cxl driver will set the capi value, like other drivers already do.

- phb4: set TVT1 for tunneled operations in capi mode

  The ASN indication is used for tunneled operations (as_notify and atomics). Tunneled operation messages can be sent in PCI mode as well as CAPI mode.

  The address field of as_notify messages is hijacked to encode the LPID/PID/TID of the target thread, so those messages should not go through address translation. Therefore bit 59 is part of the ASN indication.

  This patch sets TVT#1 in bypass mode when capi mode is enabled, to prevent as_notify messages from being dropped.

Debugging/Testing improvements

- core/stack: backtrace unwind basic OPAL call details

  Put OPAL callers' r1 into the stack back chain, and then use that to unwind back to the OPAL entry frame (as opposed to boot entry, which has a 0 back chain).

  From there, dump the OPAL call token and the caller's r1. A backtrace looks like this:

```
CPU 0000 Backtrace:
 S: 0000000031c03ba0 R: 000000003001a548   ._abort+0x4c
 S: 0000000031c03c20 R: 000000003001baac   .opal_run_pollers+0x3c
 S: 0000000031c03ca0 R: 000000003001bcbc   .opal_poll_events+0xc4
 S: 0000000031c03d20 R: 00000000300051dc   opal_entry+0x12c
 --- OPAL call entry token: 0xa caller R1: 0xc0000000006d3b90 ---
```

  This is pretty basic for the moment, but it does give you the bottom of the Linux stack. It will allow some interesting improvements in future.

  First, with the eframe, all the call's parameters can be printed out as well. The ___backtrace / ___print_backtrace API needs to be reworked in order to support this, but it's otherwise very simple (see opal_trace_entry()).

  Second, it will allow Linux's stack to be passed back to Linux via a debugging opal call. This will allow Linux's BUG() or xmon to also print the Linux back trace in case of a NMI or MCE or watchdog lockup that hits in OPAL.

- asm/head: implement quiescing without stack or clobbering regs

  Quiescing currently is implmeented in C in opal_entry before the opal call handler is called. This works well enough for simple cases like fast reset when one CPU wants all others out of the way.

  Linux would like to use it to prevent an sreset IPI from interrupting firmware, which could lead to deadlocks when crash dumping or entering the debugger. Linux interrupts do not recover well when returning back to general OPAL code, due to r13 not being restored. OPAL also can't be re-entered, which may happen e.g., from the debugger.

  So move the quiesce hold/reject to entry code, beore the stack or r1 or r13 registers are switched. OPAL can be interrupted and returned to or re-entered during this period.

  This does not completely solve all such problems. OPAL will be interrupted with sreset if the quiesce times out, and it can be interrupted by MCEs as well. These still have the issues above.

- core/opal: Allow poller re-entry if OPAL was re-entered

  If an NMI interrupts the middle of running pollers and the OS invokes pollers again (e.g., for console output), the poller re-entrancy check will prevent it from running and spam the console.

  That check was designed to catch a poller calling opal_run_pollers, OPAL re-entrancy is something different and is detected elsewhere. Avoid the poller recursion check if OPAL has been re-entered. This is a best-effort attempt to cope with errors.

- core/opal: Emergency stack for re-entry

  This detects OPAL being re-entered by the OS, and switches to an emergency stack if it was. This protects the firmware's main stack from re-entrancy and allows the OS to use NMI facilities for crash / debug functionality.

  Further nested re-entry will destroy the previous emergency stack and prevent returning, but those should be rare cases.

  This stack is sized at 16kB, which doubles the size of CPU stacks, so as not to introduce a regression in primary stack size. The 16kB stack originally had a 4kB machine check stack at the top, which was removed by 80eee1946 ("opal: Remove machine check interrupt patching in OPAL."). So it is possible the size could be tightened again, but that would require further analysis.

- hdat_to_dt: hash_prop the same on all platforms Fixes this unit test on ppc64le hosts.

- mambo: Add persistent memory disk support

  This adds support to for mapping disks images using persistent memory. Disks can be added by setting this ENV variable:

  PMEM_DISK="/mydisks/disk1.img,/mydisks/disk2.img"

  These will show up in Linux as /dev/pmem0 and /dev/pmem1.

  This uses a new feature in mambo "mysim memory mmap .." which is only available since mambo commit 0131f0fc08 (from 24/4/2018).

  This also needs the of_pmem.c driver in Linux which is only available since v4.17. It works with powernv_defconfig + CONFIG_OF_PMEM.

- external/mambo: Add di command to decode instructions

  By default you get 16 instructions but you can specify the number you want. i.e.

```
systemsim % di 0x100 4
0x0000000000000100: Enc:0xA64BB17D : mtspr   HSPRG1,r13
0x0000000000000104: Enc:0xA64AB07D : mfspr   r13,HSPRG0
0x0000000000000108: Enc:0xF0092DF9 : std     r9,0x9F0(r13)
0x000000000000010C: Enc:0xA6E2207D : mfspr   r9,PPR
```

Using di since it's what xmon uses.

- mambo/mambo_utils.tcl: Inject an MCE at a specified address

  Currently we don't support injecting an MCE on a specific address. This is useful for testing functionality like memcpy_mcsafe() (see https://patchwork.ozlabs.org/cover/893339/)

  The core of the functionality is a routine called inject_mce_ue_on_addr, which takes an addr argument and injects an MCE (load/store with UE) when the specified address is accessed by code. This functionality can easily be enhanced to cover instruction UE's as well.

  A sample use case to create an MCE on stack access would be

  ```
  set addr [mysim display gpr 1]
  inject_mce_ue_on_addr $addr
  ```

  This would cause an mce on any r1 or r1 based access

- external/mambo: improve helper for machine checks

  Improve workarounds for stop injection, because mambo often will trigger on 0x104/204 when injecting sreset/mces.

  This also adds a workaround to skip injecting on reservations to avoid infinite loops when doing inject_mce_step.

- travis: Enable ppc64le builds

  At least on the IBM Travis Enterprise instance, we can now do ppc64le builds!

  We can only build a subset of our matrix due to availability of ppc64le distros. The Dockerfiles need some tweaking to only attempt to install (x86_64 only) Mambo binaries, as well as the build scripts.

- external: Add "lpc" tool

  This is a little front-end to the lpc debugfs files to access the LPC bus from userspace on the host.

- core/test/run-trace: fix on ppc64el

5.1.104 skiboot-6.0-rc2

skiboot v6.0-rc2 was released on Wednesday May 9th 2018. It is the second release candidate of skiboot 6.0, which will become the new stable release of skiboot following the 5.11 release, first released April 6th 2018.

Skiboot 6.0 will mark the basis for op-build v2.0 and will be required for POWER9 systems.

skiboot v6.0-rc2 contains all bug fixes as of *skiboot-5.11*, *skiboot-5.10.5*, and *skiboot-5.4.9* (the currently maintained stable releases). Once 6.0 is released, we do *not* expect any further stable releases in the 5.10.x series, nor in the 5.11.x series.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

The current plan is to cut the final 6.0 in early May (maybe in a day or two after this -rc if things look okay), with skiboot 6.0 being for all POWER8 and POWER9 platforms in op-build v2.0.

Over skiboot-6.0-rc1, we have the following changes:

- Update default stop-state-disable mask to cut only stop11

  Stability improvements in microcode for stop4/stop5 are available in upstream hcode images. Stop4 and stop5 can be safely enabled by default.

  Use ~0xE0000000 to cut all but stop0,1,2 in case there are any issues with stop4/5.

  example:

```
nvram -p ibm,skiboot --update-config opal-stop-state-disable-mask=0x1FFFFFFF
```

Note: that DD2.1 chips that have a frequency <1867Mhz possible *need* to run a hcode image *different* than the default in op-build (set *BR2_HCODE_LATEST_VERSION=y* in your config)

- ibm,firmware-versions: add hcode to device tree

  op-build commit 736a08b996e292a449c4996edb264011dfe56a40 added hcode to the VERSION partition, let's parse it out and let the user know.

- ipmi: Add BMC firmware version to device tree

  BMC Get device ID command gives BMC firmware version details. Lets add this to device tree. User space tools will use this information to display BMC version details.

- mambo: Enable XER CA32 and OV32 bits on P9

  POWER9 adds 32 bit carry and overflow bits to the XER, but we need to set the relevant CTRL1 bit to enable them.

- Makefile: Fix building natively on ppc64le

  When on ppc64le and CROSS is not set by the environment, make assumes ppc64 and sets a default CROSS. Check for ppc64le as well, so that 'make' works out of the box on ppc64le.

- p9dsu: timeout for variant detection, default to 2uess

- core/direct-controls: improve p9_stop_thread error handling

  p9_stop_thread should fail the operation if it finds the thread was already quiescd. This implies something else is doing direct controls on the thread (e.g., pdbg) or there is some exceptional condition we don't know how to deal with. Proceeding here would cause things to trample on each other, for example the hard lockup watchdog trying to send a sreset to the core while it is stopped for debugging with pdbg will end in tears.

  If p9_stop_thread times out waiting for the thread to quiesce, do not hit it with a core_start direct control, because we don't know what state things are in and doing more things at this point is worse than doing nothing. There is no good recipe described in the workbook to de-assert the core_stop control if it fails to quiesce the thread. After timing out here, the thread may eventually quiesce and get stuck, but that's simpler to debug than undefied behaviour.

- core/direct-controls: fix p9_cont_thread for stopped/inactive threads

  Firstly, p9_cont_thread should check that the thread actually was quiesced before it tries to resume it. Anything could happen if we try this from an arbitrary thread state.

  Then when resuming a quiesced thread that is inactive or stopped (in a stop idle state), we must not send a core_start direct control, clear_maint must be used in these cases.

- occ: Use major version number while checking the pstate table format

  The minor version increments of the pstate table are backward compatible. The minor version is changed when the pstate table remains same and the existing reserved bytes are used for pointing new data. So use only major version number while parsing the pstate table. This will allow old skiboot to parse the pstate table and handle minor version updates.

- hmi: Clear unknown debug trigger

  On some systems, seeing hangs like this when Linux starts:

```
[ 170.027252763,5] OCC: All Chip Rdy after 0 ms
[ 170.062930145,5] INIT: Starting kernel at 0x20011000, fdt at 0x30ae0530 366247␣
↪bytes)
[ 171.238270428,5] OPAL: Switch to little-endian OS
```

If you look at the in memory skiboot console (or do *nvram -p ibm,skiboot –update-config log-level-driver=7*) we see the console get spammed with:

```
[ 5209.109790675,7] HMI: Received HMI interrupt: HMER = 0x0000400000000000
[ 5209.109792716,7] HMI: Received HMI interrupt: HMER = 0x0000400000000000
[ 5209.109794695,7] HMI: Received HMI interrupt: HMER = 0x0000400000000000
[ 5209.109796689,7] HMI: Received HMI interrupt: HMER = 0x0000400000000000
```

We're taking the debug trigger (bit 17) early on, before the hmi_debug_trigger function in the kernel is set up.

This clears the HMI in Skiboot and reports to the kernel instead of bringing down the machine.

- core/hmi: assign flags=0 in case nothing set by handle_hmi_exception

  Theoretically we could have returned junk to the OS in this parameter.

- SLW: Fix mambo boot to use stop states

  After commit 35c66b8ce5a2 ("SLW: Move MAMBO simulator checks to slw_init"), mambo boot no longer calls add_cpu_idle_state_properties() and as such we never enable stop states.

  After adding the call back, we get more testing coverage as well as faster mambo SMT boots.

- phb4: Hardware init updates

  CFG Write Request Timeout was incorrectly set to informational and not fatal for both non-CAPI and CAPI, so set it to fatal. This was a mistake in the specification. Correcting this fixes a niche bug in escalation (which is necessary on pre-DD2.2) that can cause a checkstop due to a NCU timeout.

  In addition, set the values in the timeout control registers to match. This fixes an extremely rare and unreproducible bug, though the current timings don't make sense since they're higher than the NCU timeout (16) which will checkstop the machine anyway.

- SLW: quieten 'Configuring self-restore' for DARN,NCU_SPEC_BAR and HRMOR
- Experimental support for building with Clang
- Improvements to testing and Travis CI

5.1.105 skiboot-6.0.1

skiboot 6.0.1 was released on Wednesday May 16th, 2018. It replaces *skiboot-6.0* as the current stable release in the 6.0.x series.

It is recommended that 6.0.1 be used instead of any previous 6.0.x version due to the bug fixes and debugging enhancements in it.

Over *skiboot-6.0*, we have two bug fixes:

- OpenBMC: use 0x3a as OEM command for partial add esel.

  This fixes the bug where skiboot would never send an eSEL to the BMC.

- Add location code to NPU2 HMI logging

  The current HMI error message does not specifiy where the HMI error occured.

  The original error message was

```
NPU: FIR#0 FIR 0x0080100000000000 mask 0x009a48180f01ffff
```

  The enhanced error message is

```
NPU2: [Loc: UOPWR.0000000-Node0-Proc0] P:0 FIR#0 FIR 0x0000100000000000 mask␣
↪0x009a48180f03ffff
```

5.1.106 skiboot-6.0.10

skiboot 6.0.10 was released on Wednesday October 31st, 2018. It replaces *skiboot-6.0.9* as the current stable release in the 6.0.x series.

It is recommended that 6.0.10 be used instead of any previous 6.0.x version due to the bug fixes it contains.

The bug fixes are:

- Recognise signed VERSION partition
- hdata/i2c: Skip unknown device type

  Do not add unknown I2C devices to device tree.
- hdata/i2c: Make SPD workaround more workaroundy

  We have a hack in the I2C device parser to fix up entries generated by hostboot for the DIMM SPD devices. For some reason they get reported as 128Kbit EEPROMs which is bad since those have a different I2C interface to an actual SPD device.

  Oddly enough, the FSP also gets this wrong in a slightly different way. In the FSP case they are reported as a at24c04 (4Kbit) EEPROM, which also has a different I2C interface.

  To fix both these problems for any eeprom we find on that bus to have the compatible string of "spd".
- hdata/i2c: Add whitelisting for Host I2C devices

  Many of the devices that we get information about through HDAT are for use by firmware rather than the host operating system. This patch adds a boolean flag to hdat_i2c_info structure that indicates whether devices with a given purpose should be reserved for use inside of OPAL (or some other firmware component, such as the OCC).
- Add fast-reboot property to /ibm,opal DT node

  this means that if it's permanently disabled on boot, the test suite can pick that up and not try a fast reboot test.
- libflash: Add ipmi-hiomap (currently for Witherspoon only)

  ipmi-hiomap implements the PNOR access control protocol formerly known as "the mbox protocol" but uses IPMI instead of the AST LPC mailbox as a transport. As there is no-longer any mailbox involved in this alternate implementation the old protocol name is quite misleading, and so it has been renamed to "the hiomap protoocol" (Host I/O Mapping protocol). The same commands and events are used though this client-side implementation assumes v2 of the protocol is supported by the BMC.
- AMI BMC: use 0x3a as OEM command

  The 0x3a OEM command is for IBM commands, while 0x32 was for AMI ones. Sometime in the P8 timeframe, AMI BMCs were changed to listen for our commands on either 0x32 or 0x3a. Since 0x3a is the direction forward, we'll use that, as P9 machines with AMI BMCs probably also want these to work, and let's not bet that 0x32 will continue to be okay.
- astbmc: Set romulus BMC type to OpenBMC
- Fixes to bulid with GCC8
- phb4/capp: Use link width to allocate STQ engines to CAPP

  Update phb4_init_capp_regs() to allocates STQ Engines to CAPP/PEC2 based on link width instead of always assuming it to x8.

Also re-factor the function slightly to evaluate the link-width only once and cache it so that it can also be used to allocate DMA read engines.

- phb4/capp: Update the expected Eye-catcher for CAPP ucode lid

  Currently on a FSP based P9 system load_capp_code() expects CAPP ucode lid header to have eye-catcher magic of 'CAPPPSLL'. However skiboot currently supports CAPP ucode only lids that have a eye-catcher magic of 'CAPPLIDH'. This prevents skiboot from loading the ucode with this error message:

  ```
  CAPP: ucode header invalid
  ```

  We fix this issue by updating load_capp_ucode() to use the eye-catcher value of 'CAPPLIDH' instead of 'CAPPPSLL'.

5.1.107 skiboot-6.0.11

skiboot 6.0.11 was released on Friday November 2nd, 2018. It replaces *skiboot-6.0.10* as the current stable release in the 6.0.x series.

It is recommended that 6.0.11 be used instead of any previous 6.0.x version due to the bug fixes it contains.

The bug fixes are:

- phb4/capp: Only reset FIR bits that cause capp machine check

  During CAPP recovery do_capp_recovery_scoms() will reset the CAPP Fir register just after CAPP recovery is completed. This has an unintentional side effect of preventing PRD from analyzing and reporting this error. If PRD tries to read the CAPP FIR after opal has already reset it, then it logs a critical error complaining "No active error bits found".

  To prevent this from happening we update do_capp_recovery_scoms() to only reset fir bits that cause CAPP machine check (local xstop). This is done by reading the CAPP Fir Action0/1 & Mask registers and generating a mask which is then written on CAPP_FIR_CLEAR register.

- phb4: Check for RX errors after link training

  Some PHB4 PHYs can get stuck in a bad state where they are constantly retraining the link. This happens transparently to skiboot and Linux but will causes PCIe to be slow. Resetting the PHB4 clears the problem.

  We can detect this case by looking at the RX errors count where we check for link stability. This patch does this by modifying the link optimal code to check for RX errors. If errors are occurring we retrain the link irrespective of the chip rev or card.

  Normally when this problem occurs, the RX error count is maxed out at 255. When there is no problem, the count is 0. We chose 8 as the max rx errors value to give us some margin for a few errors. There is also a knob that can be used to set the error threshold for when we should retrain the link. i.e.

  ```
  nvram -p ibm,skiboot --update-config phb-rx-err-max=8
  ```

- core/flash: Log return code when ffs_init() fails
- libflash/ipmi-hiomap: Use error codes rather than abort()
- libflash/ipmi-hiomap: Restore window state on window/protocol reset
- libflash/ipmi-hiomap: Improve event handling
- p9dsu: Describe platform BMC register configuration

  Provide the p9dsu-specific BMC configuration values required for the host kernel to drive the VGA display correctly.

- p9dsu: Add HIOMAP-over-IPMI support
- libflash/ipmi-hiomap: Cleanup allocation on init failure

5.1.108 skiboot-6.0.12

skiboot 6.0.12 was released on Monday November 12th, 2018. It replaces *skiboot-6.0.11* as the current stable release in the 6.0.x series.

It is recommended that 6.0.12 be used instead of any previous 6.0.x version due to the bug fixes it contains.

The bug fixes are:

- hiomap: quieten warning on failing to move a window

  This isn't *necessarily* an error that we should complain loudly about. If, for example, the BMC enforces the Read Only flag on a FFS partition, opening a write window *should* fail, and we do indeed test this in op-test.

  Thus we deal with the error in a well known path: returning an error code and then it's eventually a userspace problem.

- libflash/ipmi-hiomap: Respect daemon presence and flash control

5.1.109 skiboot-6.0.13

skiboot 6.0.13 was released on Wednesday November 14th, 2018. It replaces *skiboot-6.0.12* as the current stable release in the 6.0.x series.

This release includes one pflash change. This release does not modify skiboot itself, so there is no reason to upgrade to this version if you're on 6.0.12 already. This release is made exclusively so OpenBMC can ship an updated pflash from a tagged release.

The pflash change is:

- pflash: Add –skip option for reading

  Add a –skip=N option to pflash to skip N number of bytes when reading. This would allow users to print the VERSION partition without the STB header by specifying the –skip=4096 argument, and it's a more generic solution rather than making pflash depend on secure/trusted boot code.

5.1.110 skiboot-6.0.14

skiboot 6.0.14 was released on Tuesday November 27th, 2018. It replaces *skiboot-6.0.13* as the current stable release in the 6.0.x series.

It is recommended that 6.0.14 be used instead of any previous 6.0.x version due to the bug fixes it contains.

Bug fixes included in this release are:

- libflash: Don't merge ECC-protected ranges

  Libflash currently merges contiguous ECC-protected ranges, but doesn't check that the ECC bytes at the end of the first and start of the second range actually match sanely. More importantly, if blocklevel_read() is called with a position at the start of a partition that is contained somewhere within a region that has been merged it will update the position assuming ECC wasn't being accounted for. This results in the position being somewhere well after the actual start of the partition which is incorrect.

  For now, remove the code merging ranges. This means more ranges must be held and checked however it prevents incorrectly reading ECC-correct regions like below:

```
[  174.334119453,7] FLASH: CAPP partition has ECC
[  174.437349574,3] ECC: uncorrectable error: ffffffffffffffff ff
[  174.437426306,3] FLASH: failed to read the first 0x1000 from CAPP partition,␣
↪rc 14
[  174.439919343,3] CAPP: Error loading ucode lid. index=201d1
```

- ipmi: Reduce ipmi_queue_msg_sync() polling loop time to 10ms.

  On a plain boot with hiomap, this reduces the time spent in OPAL by ~170ms on p9dsu. This is due to hiomap (currently) using synchronous IPMI messages.

  It will also *significantly* reduce latency on runtime flash operations with hiomap, as we'll spend typically 10-20ms in OPAL rather than 100-200ms. It's not an ideal solution to that, but it's a quick and obvious win for jitter.

- opal-prd: Fix opal-prd crash

  Crash log without this patch:

```
opal-prd[2864]: unhandled signal 11 at 0000000000029320 nip 00000 00102012830 lr␣
↪0000000102016890 code 1
```

5.1.111 skiboot-6.0.15

skiboot 6.0.15 was released on Monday December 17th, 2018. It replaces *skiboot-6.0.14* as the current stable release in the 6.0.x series.

It is recommended that 6.0.15 be used instead of any previous 6.0.x version due to the bug fixes it contains.

Bug fixes included in this release are:

- i2c: Fix i2c request hang during opal init if timers are not checked

  If an i2c request cannot go through the first time, because the bus is found in error and need a reset or it's locked by the OCC for example, the underlying i2c implementation is using timers to manage the request. However during opal init, opal pollers may not be called, it depends in the context in which the i2c request is made. If the pollers are not called, the timers are not checked and we can end up with an i2c request which will not move foward and skiboot hangs.

  Fix it by explicitly checking the timers if we are waiting for an i2c request to complete and it seems to be taking a while.

- opal-prd: hservice: Enable hservice->wakeup() in BMC

  This patch enables HBRT to use HYP special wakeup register in openBMC which until now was only used in FSP based machines.

  This patch also adds a capability check for opal-prd so that HBRT can decide if the host special wakeup register can be used.

- npu2: Advertise correct TCE page size

  The P9 NPU workbook says that only 4K/64K/16M/256M page size are supported and in fact npu2_map_pe_dma_window() supports just these but in absence of the "ibm,supported-tce-sizes" property Linux assumes the default P9 PHB4 page sizes - 4K/64K/2M/1G - so when Linux tries 2M/1G TCEs, we get lots of "Unexpected TCE size" from npu2_tce_kill().

  This advertises TCE page sizes so Linux could handle it correctly, i.e. fall back to 4K/64K TCEs.

5.1.112 skiboot-6.0.16

skiboot 6.0.16 was released on Tuesday February 5th, 2019. It replaces *skiboot-6.0.15* as the current stable release in the 6.0.x series.

It is recommended that 6.0.16 be used instead of any previous 6.0.x version due to the bug fixes it contains.

Bug fixes included in this release are:

- p9dsu: Fix p9dsu default variant

  Add the default when no riser_id is returned from the ipmi query.

  This addresses: https://github.com/open-power/boston-openpower/issues/1369

  Allow a little more time for BMC reply and cleanup some label strings.

- p9dsu: Fix p9dsu slot tables

  Set the attributes on the slot tables to account for builtin or pluggable etypes, this will allow pci enumeration to calculate subordinate buses.

  Update some slot label strings.

  Add WIO Slot5 which is standard on the ESS config.

- phb4: Generate checkstop on AIB ECC corr/uncorr for DD2.0 parts

  On DD2.0 parts, PCIe ECC protection is not warranted in the response data path. Thus, for these parts, we need to flag any ECC errors detected from the adjacent AIB RX Data path so the part can be replaced.

  This patch configures the FIRs so that we escalate these AIB ECC errors to a checkstop so the parts can be replaced.

- core/lock: Stop drop_my_locks() from always causing abort

  Fix an erroneous failure in an error path that looked like this:

  ```
  LOCK ERROR: Releasing lock we don't hold depth @0x30493d20 (state:␣
  ↪0x0000000000000001)
  [13836.000173140,0] Aborting!
  CPU 0000 Backtrace:
   S: 0000000031c03930 R: 000000003001d840   ._abort+0x60
   S: 0000000031c039c0 R: 000000003001a0c4   .lock_error+0x64
   S: 0000000031c03a50 R: 0000000030019c70   .unlock+0x54
   S: 0000000031c03af0 R: 000000003001a040   .drop_my_locks+0xf4
  ```

5.1.113 skiboot-6.0.17

skiboot 6.0.17 was released on Wednesday February 20th, 2019. It replaces *skiboot-6.0.16* as the current stable release in the 6.0.x series.

It is recommended that 6.0.17 be used instead of any previous 6.0.x version due to the bug fixes it contains.

Bug fixes included in this release are:

- core/opal: Print PIR value in exit path, which is useful for debugging.
- core/ipmi: Improve error message
- hdata: Fix dtc warnings

  Fix dtc warnings related to mcbist node

```
Warning (reg_format): "reg" property in /xscom@623fc00000000/mcbist@1 has invalid␣
↪length (4 bytes) (#address-cells == 1, #size-cells == 1)
Warning (reg_format): "reg" property in /xscom@623fc00000000/mcbist@2 has invalid␣
↪length (4 bytes) (#address-cells == 1, #size-cells == 1)
Warning (reg_format): "reg" property in /xscom@603fc00000000/mcbist@1 has invalid␣
↪length (4 bytes) (#address-cells == 1, #size-cells == 1)
Warning (reg_format): "reg" property in /xscom@603fc00000000/mcbist@2 has invalid␣
↪length (4 bytes) (#address-cells == 1, #size-cells == 1)
```

Ideally we should add proper xscom range here… but we are not getting that information in HDAT today. Lets fix warning until we get proper data in HDAT.

- hdata/test: workaround dtc bugs

  In dtc v1.4.5 to at least v1.4.7 there have been a few bugs introduced that change the layout of what's produced in the dts. In order to be immune from them, we should use the (provided) dtdiff utility, but we also need to run the dts we're diffing against through a dtb cycle in order to ensure we get the same format as what the hdat_to_dt to dts conversion will.

  This fixes a bunch of unit test failures on the version of dtc shipped with recent Linux distros such as Fedora 29.

- firmware-versions: Add test case for parsing VERSION

  Also make it possible to use with afl-lop/afl-fuzz just to help make *sure* we're all good.

  Additionally, if we hit a entry in VERSION that is larger than our buffer size, we skip over it gracefully rather than overwriting the stack. This is only a problem if VERSION isn't trusted, which as of 4b8cc05a94513816d43fb8bd6178896b430af08f it is verified as part of Secure Boot.

- core/cpu: HID update race

  If the per-core HID register is updated concurrently by multiple threads, updates can get lost. This has been observed during fast reboot where the HILE bit does not get cleared on all cores, which can cause machine check exception interrupts to crash.

  Fix this by only updating HID on thread0.

- cpufeatures: Always advertise POWER8NVL as DD2

  Despite the major version of PVR being 1 (0x004c0100) for POWER8NVL, these chips are functionally equalent to P8/P8E DD2 levels.

  This advertises POWER8NVL as DD2. As the result, skiboot adds ibm,powerpc-cpu-features/processor-control-facility for such CPUs and the linux kernel can use hypervisor doorbell messages to wake secondary threads; otherwise "KVM: CPU %d seems to be stuck" would appear because of missing LPCR_PECEDH.

5.1.114 skiboot-6.0.18

skiboot 6.0.18 was released on Wednesday March 6th, 2019. It replaces *skiboot-6.0.17* as the current stable release in the 6.0.x series.

It is recommended that 6.0.18 be used instead of any previous 6.0.x version due to the bug fixes it contains.

Over *skiboot-6.0.17* we have several bug fixes, including important ones for powercap, ipmi-hiomap and BMC communication driver.

powercap

- powercap: occ: Fix the powercapping range allowed for user

OCC provides two limits for minimum powercap. One being hard powercap minimum which is guaranteed by OCC and the other one is a soft powercap minimum which is lesser than hard-min and may or may not be asserted due to various power-thermal reasons. So to allow the users to access the entire powercap range, this patch exports soft powercap minimum as the "powercap-min" DT property. And it also adds a new DT property called "powercap-hard-min" to export the hard-min powercap limit.

IPMI-HIOMAP

- ipmi-hiomap test case enhancements/fixes.
- libflash/ipmi-hiomap: Enforce message size for empty response

  The protocol defines the response to the associated messages as empty except for the command ID and sequence fields. If the BMC is returning extra data consider the message malformed.
- libflash/ipmi-hiomap: Remove unused close handling

  Issuing a HIOMAP_C_CLOSE is not required by the protocol specification, rather a close can be implicit in a subsequent CREATE_{READ,WRITE}_WINDOW request. The implicit close provides an opportunity to reduce LPC traffic and the implementation takes up that optimisation, so remove the case from the IPMI callback handler.
- libflash/ipmi-hiomap: Overhaul event handling

  Reworking the event handling was inspired by a bug report by Vasant where the host would get wedged on multiple flash access attempts in the face of a persistent error state on the BMC-side. The cause of this bug was the early-exit based on ctx->update, which erronously assumed that all events had been completely handled in prior calls to ipmi_hiomap_handle_events(). This is not true if e.g. HIOMAP_E_DAEMON_READY is clear in the prior calls.

  Regardless, there were other correctness and efficiency problems with the handling strategy:

  - Ack-able event state was not restored in the face of errors in the process of re-establishing protocol state
  - It forced needless window restoration with respect to the context in which ipmi_hiomap_handle_events() was called.
  - Tests for HIOMAP_E_DAEMON_READY and HIOMAP_E_FLASH_LOST were redundant with the overhauled error handling introduced in the previous patch

  Fix all of the above issues and add comments to explain the event handling flow.

  Tests for correctness follow later in the series.
- libflash/ipmi-hiomap: Overhaul error handling

  The aim is to improve the robustness with respect to absence of the BMC-side daemon. The current error handling roughly mirrors what was done for the mailbox implementation, but there's room for improvement.

  Errors are split into two classes, those that affect the transport state and those that affect the window validity. From here, we push the transport state error checks right to the bottom of the stack, to ensure the link is known to be in a good state before any message is sent. Window validity tests remain as they were in the hiomap_window_move() and ipmi_hiomap_read() functions. Validity tests are not necessary in the write and erase paths as we will receive an error response from the BMC when performing a dirty or flush on an invalid window.

  Recovery also remains as it was, done on entry to the blocklevel callbacks. If an error state is encountered in the middle of an operation no attempt is made to recover it on the spot, instead the error is returned up the stack and the caller can choose how it wishes to respond.
- libflash/ipmi-hiomap: Fix leak of msg in callback

BMC communication

- core/ipmi: Add ipmi sync messages to top of the list

  In ipmi_queue_msg_sync() path OPAL will wait until it gets response from BMC. If we do not get response ontime we may endup in kernel hardlockups. Hence lets add sync messages to top of the queue. This will reduces the chance of hardlockups.

- hw/bt: Introduce separate list for synchronous messages

  BT send logic always sends top of bt message list to BMC. Once BMC reads the message, it clears the interrupt and bt_idle() becomes true.

  bt_add_ipmi_msg_head() adds message to top of the list. If bt message list is not empty then:

  - if bt_idle() is true then we will endup sending message to BMC before getting response from BMC for inflight message. Looks like on some BMC implementation this results in message timeout.
  - else we endup starting message timer without actually sending message to BMC.. which is not correct.

  This patch introduces separate list to track synchronous messages. bt_add_ipmi_msg_head() will add messages to tail of this new list. We will always process this queue before processing normal queue.

  Finally this patch introduces new variable (inflight_bt_msg) to track inflight message. This will point to current inflight message.

- hw/bt: Fix message retry handler

  In some corner cases (like BMC reboot), bt_send_and_unlock() starts message timer, but won't send message to BMC as driver is not free to send message. bt_expire_old_msg() function enables H2B interrupt without actually sending message.

  This patch fixes above issue.

- ipmi/power: Fix system reboot issue

  Kernel makes reboot/shudown OPAL call for reboot/shutdown. Once kernel gets response from OPAL it runs opal_poll_events() until firmware handles the request.

  On BMC based system, OPAL makes IPMI call (IPMI_CHASSIS_CONTROL) to initiate system reboot/shutdown. At present OPAL queues IPMI messages and return SUCESS to Host. If BMC is not ready to accept command (like BMC reboot), then these message will fail. We have to manually reboot/shutdown the system using BMC interface.

  This patch adds logic to validate message return value. If message failed, then it will resend the message. At some stage BMC will be ready to accept message and handles IPMI message.

- hw/bt: Add backend interface to disable ipmi message retry option

  During boot OPAL makes IPMI_GET_BT_CAPS call to BMC to get BT interface capabilities which includes IPMI message max resend count, message timeout, etc,. Most of the time OPAL gets response from BMC within specified timeout. In some corner cases (like mboxd daemon reset in BMC, BMC reboot, etc) OPAL may not get response within timeout period. In such scenarios, OPAL resends message until max resend count reaches.

  OPAL uses synchronous IPMI message (ipmi_queue_msg_sync()) for few operations like flash read, write, etc. Thread will wait in OPAL until it gets response from BMC. In some corner cases like BMC reboot, thread may wait in OPAL for long time (more than 20 seconds) and results in kernel hardlockup.

  This patch introduces new interface to disable message resend option. We will disable message resend option for synchrous message. This will greatly reduces kernel hardlock up issues.

  This is short term fix. Long term solution is to convert all synchronous messages to asynhrounous one.

PHB3

- hw/phb3/naples: Disable D-states

  Putting "Mellanox Technologies MT27700 Family [ConnectX-4] [15b3:1013]" (more precisely, the second of 2 its PCI functions, no matter in what order) into the D3 state causes EEH with the "PCT timeout" error. This has been noticed on garrison machines only and firestones do not seem to have this issue.

  This disables D-states changing for devices on root buses on Naples by installing a config space access filter (copied from PHB4).

5.1.115 skiboot-6.0.19

skiboot 6.0.19 was released on Tuesday March 19th, 2019. It replaces *skiboot-6.0.18* as the current stable release in the 6.0.x series.

It is recommended that 6.0.19 be used instead of any previous 6.0.x version due to the bug fixes it contains.

Bug fixes included in this release are:

- p9dsu: Undo slot label name changes

  During some code updates the slot labels were updated to reflect the phb layout, however expectations were that the slot labels be aligned with the riser card slots and not the system planar slots.

  [stewart: The tale of how we got here is long and varied and not at all clear. The first ESS systems went out with a skiboot v5.9.8 with additional SuperMicro patches. It was probably a slot table, but who knows, we don't have the code so can't check. It's possible it was all coming in through HDAT instead). The op-build tree (thus the exact patches) shipped on systems that work correct seems to not be around anywhere anymore (if it ever was). It was only in skiboot v6.0 that a slot table made it in, and, of course, only having remote machines in random configs, including possibly with riser cards from Briggs&Stratton rather than the ones destined for this system, doesn't make for verifying this at all. It also doesn't help that *consistently* there is *never* any review on slot tables, and we've had things be wrong in the past. Combine this with not upstream Hostboot patches.]

- p9dsu: Fix slot labels for p9dsu2u

  Update the slot labels for the p9dsu2u tables.

5.1.116 skiboot-6.0.2

skiboot 6.0.2 was released on Friday May 18th, 2018. It replaces *skiboot-6.0.1* as the current stable release in the 6.0.x series.

It is recommended that 6.0.2 be used instead of any previous 6.0.x version.

Over *skiboot-6.0.1*, we one bug fix:

- cpu: Clear PCR SPR in opal_reinit_cpus()

  Currently if Linux boots with a non-zero PCR, things can go bad where some early userspace programs can take illegal instructions. This is being fixed in Linux, but in the mean time, we should cleanup in skiboot also.

  This could exhibit itself as petitboot getting killed with SIGILL and no boot devices showing up, but only in a situation where you've done a kdump from a kernel running a p8 compat guest

5.1.117 skiboot-6.0.20

skiboot 6.0.20 was released on Thursday May 9th, 2019. It replaces *skiboot-6.0.19* as the current stable release in the 6.0.x series.

It is recommended that 6.0.20 be used instead of any previous 6.0.x version due to the bug fixes it contains.

Bug fixes included in this release are:

- core/flash: Retry requests as necessary in flash_load_resource()

  We would like to successfully boot if we have a dependency on the BMC for flash even if the BMC is not current ready to service flash requests. On the assumption that it will become ready, retry for several minutes to cover a BMC reboot cycle and *eventually* rather than *immediately* crash out with:

  ```
  [  269.549748] reboot: Restarting system
  [  390.297462587,5] OPAL: Reboot request...
  [  390.297737995,5] RESET: Initiating fast reboot 1...
  [  391.074707590,5] Clearing unused memory:
  [  391.075198880,5] PCI: Clearing all devices...
  [  391.075201618,7] Clearing region 201ffe000000-201fff800000
  [  391.086235699,5] PCI: Resetting PHBs and training links...
  [  391.254089525,3] FFS: Error 17 reading flash header
  [  391.254159668,3] FLASH: Can't open ffs handle: 17
  [  392.307245135,5] PCI: Probing slots...
  [  392.363723191,5] PCI Summary:
  ...
  [  393.423255262,5] OCC: All Chip Rdy after 0 ms
  [  393.453092828,5] INIT: Starting kernel at 0x20000000, fdt at
  0x30800a88 390645 bytes
  [  393.453202605,0] FATAL: Kernel is zeros, can't execute!
  [  393.453247064,0] Assert fail: core/init.c:593:0
  [  393.453289682,0] Aborting!
  CPU 0040 Backtrace:
   S: 0000000031e03ca0 R: 000000003001af60   ._abort+0x4c
   S: 0000000031e03d20 R: 000000003001afdc   .assert_fail+0x34
   S: 0000000031e03da0 R: 00000000300146d8   .load_and_boot_kernel+0xb30
   S: 0000000031e03e70 R: 0000000030026cf0   .fast_reboot_entry+0x39c
   S: 0000000031e03f00 R: 0000000030002a4c   fast_reset_entry+0x2c
   --- OPAL boot ---
  ```

  The OPAL flash API hooks directly into the blocklevel layer, so there's no delay for e.g. the host kernel, just for asynchronously loaded resources during boot.

- pci/iov: Remove skiboot VF tracking

  This feature was added a few years ago in response to a request to make the MaxPayloadSize (MPS) field of a Virtual Function match the MPS of the Physical Function that hosts it.

  The SR-IOV specification states the the MPS field of the VF is "ResvP". This indicates the VF will use whatever MPS is configured on the PF and that the field should be treated as a reserved field in the config space of the VF. In other words, a SR-IOV spec compliant VF should always return zero in the MPS field. Adding hacks in OPAL to make it non-zero is… misguided at best.

  Additionally, there is a bug in the way pci_device structures are handled by VFs that results in a crash on fast-reboot that occurs if VFs are enabled and then disabled prior to rebooting. This patch fixes the bug by removing the code entirely. This patch has no impact on SR-IOV support on the host operating system.

- hw/xscom: Enable sw xstop by default on p9

This was disabled at some point during bringup to make life easier for the lab folks trying to debug NVLink issues. This hack really should have never made it out into the wild though, so we now have the following situation occuring in the field:

1) A bad happens
2) The host kernel recieves an unrecoverable HMI and calls into OPAL to request a platform reboot.
3) OPAL rejects the reboot attempt and returns to the kernel with OPAL_PARAMETER.
4) Kernel panics and attempts to kexec into a kdump kernel.

A side effect of the HMI seems to be CPUs becoming stuck which results in the initialisation of the kdump kernel taking a extremely long time (6+ hours). It's also been observed that after performing a dump the kdump kernel then crashes itself because OPAL has ended up in a bad state as a side effect of the HMI.

All up, it's not very good so re-enable the software checkstop by default. If people still want to turn it off they can using the nvram override.

- opal/hmi: Initialize the hmi event with old value of TFMR.

  Do this before we fix TFAC errors. Otherwise the event at host console shows no thread error reported in TFMR register.

  Without this patch the console event show TFMR with no thread error: (DEC parity error TFMR[59] injection)

```
[   53.737572] Severe Hypervisor Maintenance interrupt [Recovered]
[   53.737596]  Error detail: Timer facility experienced an error
[   53.737611]  HMER: 0840000000000000
[   53.737621]  TFMR: 3212000870e04000
```

  After this patch it shows old TFMR value on host console:

```
[ 2302.267271] Severe Hypervisor Maintenance interrupt [Recovered]
[ 2302.267305]  Error detail: Timer facility experienced an error
[ 2302.267320]  HMER: 0840000000000000
[ 2302.267330]  TFMR: 3212000870e14010
```

- libflash/ipmi-hiomap: Fix blocks count issue

  We convert data size to block count and pass block count to BMC. If data size is not block aligned then we endup sending block count less than actual data. BMC will write partial data to flash memory.

  Sample log

```
[  594.388458416,7] HIOMAP: Marked flash dirty at 0x42010 for 8
[  594.398756487,7] HIOMAP: Flushed writes
[  594.409596439,7] HIOMAP: Marked flash dirty at 0x42018 for 3970
[  594.419897507,7] HIOMAP: Flushed writes
```

  In this case HIOMAP sent data with block count=0 and hence BMC didn't flush data to flash.

  Lets fix this issue by adjusting block count before sending it to BMC.

- Fix hang in pnv_platform_error_reboot path due to TOD failure.

  On TOD failure, with TB stuck, when linux heads down to pnv_platform_error_reboot() path due to unrecoverable hmi event, the panic cpu gets stuck in OPAL inside ipmi_queue_msg_sync(). At this time, rest all other cpus are in smp_handle_nmi_ipi() waiting for panic cpu to proceed. But with panic cpu stuck inside OPAL, linux never recovers/reboot.

```
p0 c1 t0
NIA : 0x000000003001dd3c <.time_wait+0x64>
CFAR : 0x000000003001dce4 <.time_wait+0xc>
MSR : 0x9000000002803002
LR : 0x000000003002ecf8 <.ipmi_queue_msg_sync+0xec>

STACK: SP NIA
0x0000000031c236e0 0x0000000031c23760 (big-endian)
0x0000000031c23760 0x000000003002ecf8 <.ipmi_queue_msg_sync+0xec>
0x0000000031c237f0 0x00000000300aa5f8 <.hiomap_queue_msg_sync+0x7c>
0x0000000031c23880 0x00000000300aaadc <.hiomap_window_move+0x150>
0x0000000031c23950 0x00000000300ab1d8 <.ipmi_hiomap_write+0xcc>
0x0000000031c23a90 0x00000000300a7b18 <.blocklevel_raw_write+0xbc>
0x0000000031c23b30 0x00000000300a7c34 <.blocklevel_write+0xfc>
0x0000000031c23bf0 0x0000000030030be0 <.flash_nvram_write+0xd4>
0x0000000031c23c90 0x000000003002c128 <.opal_write_nvram+0xd0>
0x0000000031c23d20 0x00000000300051e4 <opal_entry+0x134>
0xc000001fea6e7870 0xc0000000000a9060 <opal_nvram_write+0x80>
0xc000001fea6e78c0 0xc000000000030b84 <nvram_write_os_partition+0x94>
0xc000001fea6e7960 0xc0000000000310b0 <nvram_pstore_write+0xb0>
0xc000001fea6e7990 0xc0000000004792d4 <pstore_dump+0x1d4>
0xc000001fea6e7ad0 0xc00000000018a570 <kmsg_dump+0x140>
0xc000001fea6e7b40 0xc000000000028e5c <panic_flush_kmsg_end+0x2c>
0xc000001fea6e7b60 0xc0000000000a7168 <pnv_platform_error_reboot+0x68>
0xc000001fea6e7bd0 0xc0000000000ac9b8 <hmi_event_handler+0x1d8>
0xc000001fea6e7c80 0xc00000000012d6c8 <process_one_work+0x1b8>
0xc000001fea6e7d20 0xc00000000012da28 <worker_thread+0x88>
0xc000001fea6e7db0 0xc0000000001366f4 <kthread+0x164>
0xc000001fea6e7e20 0xc00000000000b65c <ret_from_kernel_thread+0x5c>
```

This is because, there is a while loop towards the end of ipmi_queue_msg_sync() which keeps looping until "sync_msg" does not match with "msg". It loops over time_wait_ms() until exit condition is met. In normal scenario time_wait_ms() calls run pollers so that ipmi backend gets a chance to check ipmi response and set sync_msg to NULL.

```
while (sync_msg == msg)
        time_wait_ms(10);
```

But in the event when TB is in failed state time_wait_ms()->time_wait_poll() returns immediately without calling pollers and hence we end up looping forever. This patch fixes this hang by calling opal_run_pollers() in TB failed state as well.

- core/ipmi: Print correct netfn value
- core/lock: don't set bust_locks on lock error

  bust_locks is a big hammer that guarantees a mess if it's set while all other threads are not stopped.

  I propose removing this in the lock error paths. In debugging the previous deadlock false positive, none of the error messages printed, and the in-memory console was totally garbled due to lack of locking.

  I think it's generally better for debugging and system integrity to keep locks held when lock errors occur. Lock busting should be used carefully, just to allow messages to be printed out or machine to be restarted, probably when the whole system is single-threaded.

  Skiboot is slowly working toward that being feasible with co-operative debug APIs between firmware and host, but for the time being, difficult lock crashes are better not to corrupt everything by busting locks.

5.1.118 skiboot-6.0.3

skiboot 6.0.3 was released on Wednesday May 23rd, 2018. It replaces *skiboot-6.0.2* as the current stable release in the 6.0.x series.

It is recommended that 6.0.3 be used instead of any previous 6.0.x version.

Over *skiboot-6.0.3*, we have bug fixes related to i2c booting in secure mode, and general functionality with a TPM present. These changes are:

- p8-i2c: Remove force reset

  Force reset was added as an attempt to work around some issues with TPM devices locking up their I2C bus. In that particular case the problem was that the device would hold the SCL line down permanently due to a device firmware bug. The force reset doesn't actually do anything to alleviate the situation here, it just happens to reset the internal master state enough to make the I2C driver appear to work until something tries to access the bus again.

  On P9 systems with secure boot enabled there is the added problem of the "diagostic mode" not being supported on I2C masters A,B,C and D. Diagnostic mode allows the SCL and SDA lines to be driven directly by software. Without this force reset is impossible to implement.

  This patch removes the force reset functionality entirely since:

  a) it doesn't do what it's supposed to, and

  b) it's butt ugly code

  Additionally, turn p8_i2c_reset_engine() into p8_i2c_reset_port(). There's no need to reset every port on a master in response to an error that occurred on a specific port.

- libstb/i2c-driver: Bump max timeout

  We have observed some TPMs clock streching the I2C bus for signifigant amounts of time when processing commands. The same TPMs also have errata that can result in permernantly locking up a bus in response to an I2C transaction they don't understand. Using an excessively long timeout to prevent this in the field.

- Add TPM timeout workaround

  Set the default timeout for any bus containing a TPM to one second. This is needed to work around a bug in the firmware of certain TPMs that will clock strech the I2C port the for up to a second. Additionally, when the TPM is clock streching it responds to a STOP condition on the bus by bricking itself. Clearing this error requires a hard power cycle of the system since the TPM is powered by standby power.

5.1.119 skiboot-6.0.4

skiboot 6.0.4 was released on Monday May 28th, 2018. It replaces *skiboot-6.0.3* as the current stable release in the 6.0.x series.

It is recommended that 6.0.4 be used instead of any previous 6.0.x version.

Over *skiboot-6.0.3*, we have two bug fixes: one helps with performance (especially in HPC environments), and one is an opal-prd fix.

Changes are:

- SLW: Remove stop1_lite and stop2_lite

  stop1_lite has been removed since it adds no additional benefit over stop0_lite. stop2_lite has been removed since currently it adds minimal benefit over stop2. However, the benefit is eclipsed by the time required to ungate the clocks

Moreover, Lite states don't give up the SMT resources, can potentially have a performance impact on sibling threads.

Since current OSs (Linux) aren't smart enough to make good decisions with these stop states, we're (temporarly) removing them from what we expose to the OS, the idea being to bring them back in a new DT representation so that only an OS that knows what to do will do things with them.

- opal-prd: Do not error out on first failure for soft/hard offline.

  The memory errors (CEs and UEs) that are detected as part of background memory scrubbing are reported by PRD asynchronously to opal-prd along with affected memory ranges. hservice_memory_error() converts these ranges into page granularity before hooking up them to soft/hard offline-ing infrastructure.

  But the current implementation of hservice_memory_error() does not hookup all the pages to soft/hard offline-ing if any of the page offline action fails. e.g hard offline can fail for:

  - Pages that are not part of buddy managed pool.
  - Pages that are reserved by kernel using memblock_reserved()
  - Pages that are in use by kernel.

  But for the pages that are in use by user space application, the hard offline marks the page as hwpoison, sends SIGBUS signal to kill the affected application as recovery action and returns success.

  Hence, It is possible that some of the pages in that memory range are in use by application or free. By stopping on first error we loose the opportunity to hwpoison the subsequent pages which may be free or in use by application. This patch fixes this issue.

5.1.120 skiboot-6.0.5

skiboot 6.0.5 was released on Wednesday July 11th, 2018. It replaces *skiboot-6.0.4* as the current stable release in the 6.0.x series.

It is recommended that 6.0.5 be used instead of any previous 6.0.x version.

Over *skiboot-6.0.4* we have several bug fixes, including important ones for NVLINK2 and NX.

PCI/PHB4

- phb4: Delay training till after PERST is deasserted

  This helps some cards train on the second PERST (ie fast-reboot). The reason is not clear why but it helps, so YOLO!

- pci: Fix PCI_DEVICE_ID()

  The vendor ID is 16 bits not 8. This error leaves the top of the vendor ID in the bottom bits of the device ID, which resulted in e.g. a failure to run the PCI quirk for the AST VGA device.

  Fixes: 2b841bf0ef1b (present in v5.7-rc1)

PHB4/CAPI

- phb4/capp: Calculate STQ/DMA read engines based on link-width for PEC

  Presently in CAPI mode the number of STQ/DMA-read engines allocated on PEC2 for CAPP is fixed to 6 and 0-30 respectively irrespective of the PCI link width. These values are only suitable for x8 cards and quickly run out if a x16 card is plugged to a PEC2 attached slot. This usually manifests as CAPP reporting TLBI timeout due to these messages getting stalled due to insufficient STQs.

To fix this we update enable_capi_mode() to check if PEC2 chiplet is in x16 mode and if yes then we allocate 4/0-47 STQ/DMA-read engines for the CAPP traffic.

- capi: Select the correct IODA table entry for the mbt cache.

  With the current code, the capi mmio window is not correctly configured in the IODA table entry. The first entry (generally the non-prefetchable BAR) is overwrriten. This patch sets the capi window bar at the right place.

Sensors

- occ: sensors: Fix the size of the phandle array 'sensors' in DT

  Fixes: 99505c03f493 (present in v5.10-rc4)

NPU2/NVLINK2

- npu2/hw-procedures: Fence bricks via NTL instead of MISC

  There are a couple of places we can set/unset fence for a brick:

  1. MISC register: NPU2_MISC_FENCE_STATE
  2. NTL register for the brick: NPU2_NTL_MISC_CFG1(ndev)

  Recent testing of ATS in combination with GPU reset has exposed a side effect of using (1); if fence is set for all six bricks, it triggers a sticky nmmu latch which prevents the NPU from getting ATR responses. This manifests as a hang in the tests.

  We have npu2_dev_fence_brick() which uses (1), and only two calls to it. Replace the call which sets fence with a write to (2). Remove the corresponding unset call entirely. It's unneeded because the procedures already do a progression from full fence to half to idle using (2).

- opal/hmi: Display correct chip id while printing NPU FIRs.

  HMIs for NPU xstops are broadcasted to all chips. All cores on all the chips receive HMI. HMI handler correctly identifies and extracts the NPU FIR details from affected chip, but while printing FIR data it prints chip id and location code details of this_cpu()->chip_id which may not be correct. This patch fixes this issue.

  Fixes: 7bcbc78c (present in v6.0.1)

VPD

- vpd: Add vendor property to processor node

  Processor FRU vpd doesn't contain vendor detail. We have to parse module VPD to get vendor detail.

- vpd: Sanitize VPD data

  On OpenPower system, VPD keyword size tells us the maximum size of the data. But they fill trailing end with space (0x20) instead of NULL. Also spec doesn't stop user to have space (0x20) within actual data.

  This patch discards trailing spaces before populating device tree.

NX/VAS for POWER9

- NX: Add NX coprocessor init opal call

  The read offset (4:11) in Receive FIFO control register is incremented by FIFO size whenever CRB read by NX. But the index in RxFIFO has to match with the corresponding entry in FIFO maintained by VAS in kernel. VAS entry is reset to 0 when opening the receive window during driver initialization. So when NX842 is reloaded or in kexec boot, possibility of mismatch between RxFIFO control register and VAS entries in kernel. It could cause CRB failure / timeout from NX.

This patch adds nx_coproc_init opal call for kernel to initialize readOffset (4:11) and Queued (15:23) in RxFIFO control register.

Fixes: 3b3c5962f432 (present in v5.8-rc1)

5.1.121 skiboot-6.0.6

skiboot 6.0.6 was released on Thursday July 19th, 2018. It replaces *skiboot-6.0.5* as the current stable release in the 6.0.x series.

It is recommended that 6.0.5 be used instead of any previous 6.0.x version, especially in the case where NVLINK2 GPUs and/or Mellanox CX5 adapters are being used.

Over *skiboot-6.0.5* we have several important performance related bug fixes and one stability bug fix:

- phb4/CAPI: Reallocate PEC2 DMA-Read engines to improve GPU-Direct bandwidth

  We reallocate additional 16/8 DMA-Read engines allocated to stack0/1 on PEC2 respectively. This is needed to improve bandwidth available to the Mellanox CX5 adapter when trying to read GPU memory (GPU-Direct).

  If kernel cxl driver indicates a request to allocate maximum possible DMA read engines when calling enable_capi_mode() and card is attached to PEC2/stack0 slot then we assume its a Mellanox CX5 adapter. We then allocate additional 16/8 extra DMA read engines to stack0 and stack1 respectively on PEC2. This is done by populating the XPEC_PCI_PRDSTKOVR and XPEC_NEST_READ_STACK_OVERRIDE as suggested by the h/w team.

- phb4: Disable nodal scoped DMA accesses when PB pump mode is enabled

  By default when a PCIe device issues a read request via the PHB it is first issued with nodal scope. When accessing GPU memory the NPU does not know at the time of response if the requested memory page is off node or not. Therefore every read of GPU memory by a PHB is retried with larger scope which introduces bandwidth and latency issues.

  On smaller boxes which have pump mode enabled nodal and group scoped reads are treated the same and both types of request are broadcast to one chip. Therefore we can avoid the retry by disabling nodal scope on the PHB for these boxes. On larger boxes nodal (single chip) and group (multiple chip) scoped reads are treated differently. Therefore we avoid disabling nodal scope on large boxes which have pump mode disabled to avoid all PHB requests being broadcast to multiple chips.

- npu2/hw-procedures: Enable parity and credit overflow checks

  Enable these error checking features by setting the appropriate bits in our one-off initialization of each "NTL Misc Config 2" register.

  The exception is NDL RX parity checking, which should be disabled during the link training procedures.

5.1.122 skiboot-6.0.7

skiboot 6.0.7 was released on Friday August 3rd, 2018. It replaces *skiboot-6.0.6* as the current stable release in the 6.0.x series.

It is recommended that 6.0.7 be used instead of any previous 6.0.x version due to it containing a workaround for hardware errata in the XIVE interrupt controller (present on POWER9 systems).

The bug fix is:

- xive: Disable block tracker

  Due to some HW errata, the block tracking facility (performance optimisation for large systems) should be disabled on Nimbus chips. Disable it unconditionally for now.

5.1.123 skiboot-6.0.8

skiboot 6.0.8 was released on Thursday August 16th, 2018. It replaces *skiboot-6.0.7* as the current stable release in the 6.0.x series.

It is recommended that 6.0.8 be used instead of any previous 6.0.x version due to the bug fixes it contains.

The bug fixes are:

- i2c: Ensure ordering between i2c_request_send() and completion

  i2c_request_send loops waiting for a flag "uc.done" set by the completion routine, and then look for a result code also set by that same completion.

  There is no synchronization, the completion can happen on another processor, so we need to order the stores to uc and the reads from uc so that uc.done is stored last and tested first using memory barriers.

- i2c: Fix multiple-enqueue of the same request on NACK

  i2c_request_send() will retry the request if the error is a NAK, however it forgets to clear the "ud.done" flag. It will thus loop again and try to re-enqueue the same request causing internal request list corruption.

- phb4: Disable 32-bit MSI in capi mode

  If a capi device does a DMA write targeting an address lower than 4GB, it does so through a 32-bit operation, per the PCI spec. In capi mode, the first TVE entry is configured in bypass mode, so the address is valid. But with any (bad) luck, the address could be 0xFFFFxxxx, thus looking like a 32-bit MSI.

  We currently enable both 32-bit and 64-bit MSIs, so the PHB will interpret the DMA write as a MSI, which very likely results in an EEH (MSI with a bad payload size).

  We can fix it by disabling 32-bit MSI when switching the PHB to capi mode. Capi devices are 64-bit.

- capp: Fix the capp recovery timeout comparison

  The current capp recovery timeout control loop in do_capp_recovery_scoms() uses a wrong comparison for return value of tb_compare(). This may cause do_capp_recovery_scoms() to report an timeout earlier than the 168ms stipulated time.

  The patch fixes this by updating the loop timeout control branch in do_capp_recovery_scoms() to use the correct enum tb_cmpval.

- phb4/capp: Update DMA read engines set in APC_FSM_READ_MASK based on link-width

  Commit 47c09cdfe7a3("phb4/capp: Calculate STQ/DMA read engines based on link-width for PEC") update the CAPP init sequence by calculating the needed STQ/DMA-read engines based on link width and populating it in XPEC_NEST_CAPP_CNTL register. This however needs to be synchronized with the value set in CAPP APC FSM Read Machine Mask Register.

  Hence this patch update phb4_init_capp_regs() to calculate the link width of the stack on PEC2 and populate the same values as previously populated in PEC CAPP_CNTL register.

- core/cpu: Call memset with proper cpu_thread offset

5.1.124 skiboot-6.0.9

skiboot 6.0.9 was released on Friday October 12th, 2018. It replaces *skiboot-6.0.8* as the current stable release in the 6.0.x series.

It is recommended that 6.0.9 be used instead of any previous 6.0.x version due to the bug fixes it contains.

The bug fixes are:

- opal/hmi: Ignore debug trigger inject core FIR.

  Core FIR[60] is a side effect of the work around for the CI Vector Load issue in DD2.1. Usually this gets delivered as HMI with HMER[17] where Linux already ignores it. But it looks like in some cases we may happen to see CORE_FIR[60] while we are already in Malfunction Alert HMI (HMER[0]) due to other reasons e.g. CAPI recovery or NPU xstop. If that happens then just ignore it instead of crashing kernel as not recoverable.

- opal/hmi: Handle early HMIs on thread0 when secondaries are still in OPAL.

  When primary thread receives a CORE level HMI for timer facility errors while secondaries are still in OPAL, thread 0 ends up in rendez-vous waiting for secondaries to get into hmi handling. This is because OPAL runs with MSR(EE=0) and hence HMIs are delayed on secondary threads until they are given to Linux OS. Fix this by adding a check for secondary state and force them in hmi handling by queuing job on secondary threads.

  I have tested this by injecting HDEC parity error very early during Linux kernel boot. Recovery works fine for non-TB errors. But if TB is bad at this very eary stage we already doomed.

  Without this patch we see:

  ```
  [  285.046347408,7] OPAL: Start CPU 0x0843 (PIR 0x0843) -> 0x000000000000a83c
  [  285.051160609,7] OPAL: Start CPU 0x0844 (PIR 0x0844) -> 0x000000000000a83c
  [  285.055359021,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
  [  285.055361439,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:0:␣
  ↪TFMR(2e12002870e14000) Timer Facility Error
  [  286.232183823,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
  ↪thread 1 (sptr=0000ccc1)
  [  287.409002056,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
  ↪thread 2 (sptr=0000ccc1)
  [  289.073820164,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
  ↪thread 3 (sptr=0000ccc1)
  [  290.250638683,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
  ↪thread 1 (sptr=0000ccc2)
  [  291.427456821,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
  ↪thread 2 (sptr=0000ccc2)
  [  293.092274807,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
  ↪thread 3 (sptr=0000ccc2)
  [  294.269092904,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
  ↪thread 1 (sptr=0000ccc3)
  [  295.445910944,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
  ↪thread 2 (sptr=0000ccc3)
  [  297.110728970,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
  ↪thread 3 (sptr=0000ccc3)
  ```

  After this patch:

  ```
  [  259.401719351,7] OPAL: Start CPU 0x0841 (PIR 0x0841) -> 0x000000000000a83c
  [  259.406259572,7] OPAL: Start CPU 0x0842 (PIR 0x0842) -> 0x000000000000a83c
  [  259.410615534,7] OPAL: Start CPU 0x0843 (PIR 0x0843) -> 0x000000000000a83c
  [  259.415444519,7] OPAL: Start CPU 0x0844 (PIR 0x0844) -> 0x000000000000a83c
  [  259.419641401,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
  [  259.419644124,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:0:␣
  ↪TFMR(2e12002870e04000) Timer Facility Error
  ```

(continues on next page)

(continued from previous page)

```
[  259.419650678,7] HMI: Sending hmi job to thread 1
[  259.419652744,7] HMI: Sending hmi job to thread 2
[  259.419653051,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
[  259.419654725,7] HMI: Sending hmi job to thread 3
[  259.419654916,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
[  259.419658025,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
[  259.419658406,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:2:␣
↪TFMR(2e12002870e04000) Timer Facility Error
[  259.419663095,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:3:␣
↪TFMR(2e12002870e04000) Timer Facility Error
[  259.419655234,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:1:␣
↪TFMR(2e12002870e04000) Timer Facility Error
[  259.425109779,7] OPAL: Start CPU 0x0845 (PIR 0x0845) -> 0x000000000000a83c
[  259.429870681,7] OPAL: Start CPU 0x0846 (PIR 0x0846) -> 0x000000000000a83c
[  259.434549250,7] OPAL: Start CPU 0x0847 (PIR 0x0847) -> 0x000000000000a83c
```

- hw/bt.c: quieten all the noisy BT/IPMI messages
- npu2: Use correct kill type for TCE invalidation

  kill_type is enum of OPAL_PCI_TCE_KILL_PAGES, OPAL_PCI_TCE_KILL_PE, OPAL_PCI_TCE_KILL_ALL and phb4_tce_kill() gets it right but npu2_tce_kill() uses OPAL_PCI_TCE_KILL which is an OPAL API token.
- hw/npu2-opencapi: Fix setting of supported OpenCAPI templates

  In opal_npu_tl_set(), we made a typo that means the OPAL_NPU_TL_SET call may not clear the enable bits for templates that were previously enabled but are now disabled.

  Fix the typo so we clear NPU2_OTL_CONFIG1_TX_TEMP2_EN as well as TEMP{1,3}_EN.
- phb4: Workaround PHB errata with CFG write UR/CA errors

  If the PHB encounters a UR or CA status on a CFG write, it will incorrectly freeze the wrong PE. Instead of using the PE# specified in the CONFIG_ADDRESS register, it will use the PE# of whatever MMIO occurred last.

  Work around this disabling freeze on such errors
- phb4: Handle allocation errors in phb4_eeh_dump_regs()

  If the zalloc fails (and it can be a rather large allocation), we will overwite memory at 0 instead of failing.
- phb4: Don't try to access non-existent PEST entries

  In a POWER9 chip, some PHB4s have 256 PEs, some have 512.

  Currently, the diagnostics code retrieves 512 unconditionally, which is wrong and causes us to incorrectly report bogus values for the "high" PEs on the small PHBs.

  Use the actual number of implemented PEs instead
- phb4: Don't probe a PHB if its garded

  Presently phb4_probe_stack() causes an exception while trying to probe a PHB if its garded. This causes skiboot to go into a reboot loop with following exception log:

```
***********************************************
Fatal MCE at 000000003006ecd4   .probe_phb4+0x570
CFAR : 00000000300b98a0
<snip>
Aborting!
```

(continues on next page)

(continued from previous page)

```
CPU 0018 Backtrace:
S: 0000000031cc37e0 R: 000000003001a51c   ._abort+0x4c
S: 0000000031cc3860 R: 0000000030028170   .exception_entry+0x180
S: 0000000031cc3a40 R: 0000000000001f10 *
S: 0000000031cc3c20 R: 000000003006ecb0   .probe_phb4+0x54c
S: 0000000031cc3e30 R: 0000000030014ca4   .main_cpu_entry+0x5b0
S: 0000000031cc3f00 R: 0000000030002700   boot_entry+0x1b8
```

This is caused as phb4_probe_stack() will ignore all xscom read/write errors to enable PHB Bars and then tries to perform an mmio to read PHB Version registers that cause the fatal MCE.

We fix this by ignoring the PHB probe if the first xscom_write() to populate the PHB Bar register fails, which indicates that there is something wrong with the PHB.

5.1.125 skiboot-6.1

skiboot v6.1 was released on Wednesday July 11th 2018. It is the first release of skiboot 6.1, which is the new stable release of skiboot following the 6.0 release, first released May 11th 2018.

Skiboot 6.1 is the basis for op-build v2.1 and contains all bug fixes as of *skiboot-6.0.5*, and *skiboot-5.4.9* (the currently maintained stable releases). We expect further stable releases in the 6.0.x and 5.4.x series, while we do not expect to do any stable releases of 6.1.x.

This final 6.1 release follows a single release candidate release, as this cycle we have been rather quiet, with mainly cleanup and bug fix patches going in.

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

Over skiboot-6.0, we have the following changes:

General changes and bug fixes

Since *skiboot-6.1-rc1*:

- slw: Fix trivial typo in debug message
- vpd: Add vendor property to processor node

  Processor FRU vpd doesn't contain vendor detail. We have to parse module VPD to get vendor detail.
- vpd: Sanitize VPD data

  On OpenPower system, VPD keyword size tells us the maximum size of the data. But they fill trailing end with space (0x20) instead of NULL. Also spec doesn't stop user to have space (0x20) within actual data.

  This patch discards trailing spaces before populating device tree.
- core: always flush console before stopping

  This catches a few cases (e.g., fast reboot failure messages) that don't always make it to the console before the machine is rebooted.
- core/cpu: parallelise global CPU register setting jobs

  On a 176 thread system, before:

  ```
  [  122.319923233,5] OPAL: Switch to big-endian OS
  [  126.317897467,5] OPAL: Switch to little-endian OS
  ```

  after:

```
[  212.439299889,5] OPAL: Switch to big-endian OS
[  212.469323643,5] OPAL: Switch to little-endian OS
```

- init, occ: Initialise OCC earlier on BMC systems

  We need to use the OCC to obtain presence data for the SXM2 slots on Witherspoon systems. This is needed to determine device type for NVLink GPUs and OpenCAPI devices which can be plugged into the same slot. Support for this will be implemented in a future patch.

  Currently, OCC initialisation is done just before handing over to Linux, which is well after NPU probe. On FSP systems, OCC boot starts very late, so we wait until the last possible moment to initialise the skiboot side in order to give it the maximum time to boot. On BMC systems, OCC boot starts earlier, so there aren't any issues in moving it earlier in the skiboot init sequence.

  When running on a BMC machine, call occ_pstates_init() as early as possible in the init sequence. On FSP machines, continue to call it from its current location.

Since *skiboot-6.0*:

- GCC8 build fixes
- Add prepare_hbrt_update to hbrt interfaces

  Add placeholder support for prepare_hbrt_update call into hostboot runtime (opal-prd) code. This interface is only called as part of a concurrent code update on a FSP based system.

- cpu: Clear PCR SPR in opal_reinit_cpus()

  Currently if Linux boots with a non-zero PCR, things can go bad where some early userspace programs can take illegal instructions. This is being fixed in Linux, but in the mean time, we should cleanup in skiboot also.

- pci: Fix PCI_DEVICE_ID()

  The vendor ID is 16 bits not 8. This error leaves the top of the vendor ID in the bottom bits of the device ID, which resulted in e.g. a failure to run the PCI quirk for the AST VGA device.

- Quieten console output on boot

  We print out a whole bunch of things on boot, most of which aren't interesting, so we should *not* print them instead.

  Printing things like what CPUs we found and what PCI devices we found *are* useful, so continue to do that. But we don't need to splat out a bunch of things that are always going to be true.

- core/console: fix deadlock when printing with console lock held

  Some debugging options will print while the console lock is held, which is why the console lock is taken as a recursive lock. However console_write calls __flush_console, which will drop and re-take the lock non-recursively in some cases.

  Just set con_need_flush and return from __flush_console if we are holding the console lock already.

  This stack usage message (taken with this patch applied) could lead to a deadlock without this:

```
CPU 0000 lowest stack mark 11768 bytes left pc=300cb808 token=0
CPU 0000 Backtrace:
S: 0000000031c03370 R: 00000000300cb808   .list_check_node+0x1c
S: 0000000031c03410 R: 00000000300cb910   .list_check+0x38
S: 0000000031c034b0 R: 00000000300190ac   .try_lock_caller+0xb8
S: 0000000031c03540 R: 00000000300192e0   .lock_caller+0x80
S: 0000000031c03600 R: 0000000030012c70   .__flush_console+0x134
S: 0000000031c036d0 R: 00000000300130cc   .console_write+0x68
S: 0000000031c03780 R: 00000000300347bc   .vprlog+0xc8
```

(continues on next page)

(continued from previous page)

```
S: 0000000031c03970 R: 0000000030034844   ._prlog+0x50
S: 0000000031c03a00 R: 00000000300364a4   .log_simple_error+0x74
S: 0000000031c03b90 R: 000000003004ab48   .occ_pstates_init+0x184
S: 0000000031c03d50 R: 000000003001480c   .load_and_boot_kernel+0x38c
S: 0000000031c03e30 R: 000000003001571c   .main_cpu_entry+0x62c
S: 0000000031c03f00 R: 0000000030002700   boot_entry+0x1c0
```

- opal-prd: Do not error out on first failure for soft/hard offline.

  The memory errors (CEs and UEs) that are detected as part of background memory scrubbing are reported by PRD asynchronously to opal-prd along with affected memory ranges. hservice_memory_error() converts these ranges into page granularity before hooking up them to soft/hard offline-ing infrastructure.

  But the current implementation of hservice_memory_error() does not hookup all the pages to soft/hard offline-ing if any of the page offline action fails. e.g hard offline can fail for:

  - Pages that are not part of buddy managed pool.
  - Pages that are reserved by kernel using memblock_reserved()
  - Pages that are in use by kernel.

  But for the pages that are in use by user space application, the hard offline marks the page as hwpoison, sends SIGBUS signal to kill the affected application as recovery action and returns success.

  Hence, It is possible that some of the pages in that memory range are in use by application or free. By stopping on first error we loose the opportunity to hwpoison the subsequent pages which may be free or in use by application. This patch fixes this issue.

- libflash/blocklevel_write: Fix missing error handling

  Caught by scan-build, we seem to trap the errors in rc, but not take any recovery action during blocklevel_write.

I2C

- p8-i2c: fix wrong request status when a reset is needed

  If the bus is found in error state when starting a new request, the engine is reset and we enter recovery. However, once complete, the reset operation shows a status of complete in the status register. So any badly-timed called to check_status() will think the current top request is complete, even though it hasn't run yet.

  So don't update any request status while we are in recovery, as nothing useful for the request is supposed to happen in that state.

- p8-i2c: Remove force reset

  Force reset was added as an attempt to work around some issues with TPM devices locking up their I2C bus. In that particular case the problem was that the device would hold the SCL line down permanently due to a device firmware bug. The force reset doesn't actually do anything to alleviate the situation here, it just happens to reset the internal master state enough to make the I2C driver appear to work until something tries to access the bus again.

  On P9 systems with secure boot enabled there is the added problem of the "diagostic mode" not being supported on I2C masters A,B,C and D. Diagnostic mode allows the SCL and SDA lines to be driven directly by software. Without this force reset is impossible to implement.

  This patch removes the force reset functionality entirely since:

  a) it doesn't do what it's supposed to, and
  b) it's butt ugly code

Additionally, turn p8_i2c_reset_engine() into p8_i2c_reset_port(). There's no need to reset every port on a master in response to an error that occurred on a specific port.

- libstb/i2c-driver: Bump max timeout

  We have observed some TPMs clock streching the I2C bus for signifigant amounts of time when processing commands. The same TPMs also have errata that can result in permernantly locking up a bus in response to an I2C transaction they don't understand. Using an excessively long timeout to prevent this in the field.

- hdata: Add TPM timeout workaround

  Set the default timeout for any bus containing a TPM to one second. This is needed to work around a bug in the firmware of certain TPMs that will clock strech the I2C port the for up to a second. Additionally, when the TPM is clock streching it responds to a STOP condition on the bus by bricking itself. Clearing this error requires a hard power cycle of the system since the TPM is powered by standby power.

- p8-i2c: Allow a per-port default timeout

  Add support for setting a default timeout for the I2C port to the device-tree. This is consumed by skiboot.

IPMI Watchdog

- ipmi-watchdog: Support handling re-initialization

  Watchdog resets can return an error code from the BMC indicating that the BMC watchdog was not initialized. Currently we abort skiboot due to a missing error handler. This patch implements handling re-initialization for the watchdog, automatically saving the last watchdog set values and re-issuing them if needed.

- ipmi-watchdog: The stop action should disable reset

  Otherwise it is possible for the reset timer to elapse and trigger the watchdog to wake back up. This doesn't affect the behavior of the system since we are providing a NONE action to the BMC. However we would like to avoid the action from taking place if possible.

- ipmi-watchdog: Add a flag to determine if we are still ticking

  This makes it easier for future changes to ensure that the watchdog stops ticking and doesn't requeue itself for execution in the background. This way it is safe for resets to be performed after the ticks are assumed to be stopped and it won't start the timer again.

- ipmi-watchdog: (prepare for) not disabling at shutdown

  The op-build linux kernel has been configured to support the ipmi watchdog. This driver will always handle the watchdog by either leaving it enabled if configured, or by disabling it during module load if no configuration is provided. This increases the coverage of the watchdog during the boot process. The watchdog should no longer be disabled at any point during skiboot execution.

  We're not enabling this by default yet as people can (and do, at least in development) mix and match old BOOTKERNEL with new skiboot and we don't want to break that too obviously.

- ipmi-watchdog: Don't reset the watchdog twice

  There is no clarification for why this change was needed, but presumably this is due to a buggy BMC implementation where the Watchdog Set command was processed concurrently or after the initial Watchdog Reset. This inversion would cause the watchdog to stop since the DONT_STOP bit was not set. Since we are now using the DONT_STOP bit during initialization, the watchdog should not be stopped even if an inversion occurs.

- ipmi-watchdog: Make it possible to set DONT_STOP

  The IPMI standard supports setting a DONT_STOP bit during an Watchdog Set operation. Most of the time we don't want to stop the Watchdog when updating the settings so we should be using this bit. This patch makes it possible for callers of set_wdt to prevent the watchdog from being stopped. This only changes the behavior of the watchdog during the initial settings update when initializing skiboot. The watchdog is no longer disabled and then immediately re-enabled.

- ipmi-watchdog: WD_POWER_CYCLE_ACTION -> WD_RESET_ACTION

  The IPMI specification denotes that action 0x1 is Host Reset and 0x3 is Host Power Cycle. Use the correct name for Reset in our watchdog code.

POWER8 platforms

- astbmc: Enable mbox depending on scratch reg

  P8 boxes can opt in for mbox pnor support if they set the scratch register bit to indicate it is supported.

Simulator platforms

Since *skiboot-6.1-rc1*:

- pmem: volatile bindings for the poorly enabled

  PMEM_DISK bindings were added, but they rely on a rather recent mmap feature. This patch steals from those bindings to add volatile bindings. I've used these bindings with PMEM_VOLATILE to launch an instance with the publicly available systemsim-p9. The bindings are volatile and one should not expect any data to be saved/retrieved.

Since *skiboot-6.0*:

- plat/qemu: add PNOR support

  To access the PNOR, OPAL/skiboot drives the BMC SPI controller using the iLPC2AHB device of the BMC SuperIO controller and accesses the flash contents using the LPC FW address space on which the PNOR is remapped.

  The QEMU PowerNV machine now integrates such models (SuperIO controller, iLPC2AHB device) and also a pseudo Aspeed SoC AHB memory space populated with the SPI controller registers (same model as for ARM). The AHB window giving access to the contents of the BMC SPI controller flash modules is mapped on the LPC FW address space.

  The change should be compatible for machine without PNOR support.

- external/mambo: Add support for readline if it exists

  Add support for tclreadline package if it is present. This patch loads the package and uses it when the simulation stops for any reason.

FSP based platforms

- Disable fast reboot on FSP IPL side change

  If FSP changes next IPL side, then disable fast reboot.

  sample output:

```
[  620.196442259,5] FSP: Got sysparam update, param ID 0xf0000007
[  620.196444501,5] CUPD: FW IPL side changed. Disable fast reboot
[  620.196445389,5] CUPD: Next IPL side : perm
```

- fsp/console: Always establish OPAL console API backend

  Currently we only call set_opal_console() to establish the backend used by the OPAL console API if we find at least one FSP serial port in HDAT.

  On systems where there is none (IPMI only), we fail to set it, causing the console code to try to use the dummy console causing an assertion failure during boot due to clashing on the device-tree node names.

  So always set it if an FSP is present

AST BMC based platforms

- AMI BMC: use 0x3a as OEM command

  The 0x3a OEM command is for IBM commands, while 0x32 was for AMI ones. Sometime in the P8 timeframe, AMI BMCs were changed to listen for our commands on either 0x32 or 0x3a. Since 0x3a is the direction forward, we'll use that, as P9 machines with AMI BMCs probably also want these to work, and let's not bet that 0x32 will continue to be okay.

- astbmc: Set romulus BMC type to OpenBMC
- platform/astbmc: Do not delete compatible property

  P9 onwards OPAL is building device tree for BMC based system using HDAT. We are populating bmc/compatible node with bmc version. Hence do not delete this property.

Utilities

- external/xscom-utils: Add python library for xscom access

  Patch adds a simple python library module for xscom access. It directly manipulate the '/access' file for scom read and write from debugfs 'scom' directory.

  Example on how to generate a getscom using this module:

  ```
  from adu_scoms import *
  getscom = GetSCom()
  getscom.parse_args()
  getscom.run_command()
  ```

  Sample output for above getscom.py:

  ```
  # ./getscom.py -l
  Chip ID  | Rev   | Chip type
  ---------|-------|-----------
  00000008 | DD2.0 | P9 (Nimbus) processor
  00000000 | DD2.0 | P9 (Nimbus) processor
  ```

- ffspart: Don't require user to create blank partitions manually

  Add '–allow-empty' which allows the filename for a given partition to be blank. If set ffspart will set that part of the PNOR file 'blank' and set ECC bits if required. Without this option behaviour is unchanged and ffspart will return an error if it can not find the partition file.

- pflash: Use correct prefix when installing

  pflash uses lowercase prefix when running make install in it's direcetory, but uppercase PREFIX when running it in shared. Use lowercase everywhere.

  With this the OpenBMC bitbake recipie can drop an out of tree patch it's been carrying for years.

POWER9

Since *skiboot-6.1-rc1*:

- occ: sensors: Fix the size of the phandle array 'sensors' in DT

  Fixes: 99505c03f493 (present in v5.10-rc4)

- phb4: Delay training till after PERST is deasserted

  This helps some cards train on the second PERST (ie fast-reboot). The reason is not clear why but it helps, so YOLO!

Since *skiboot-6.0*:

- occ-sensor: Avoid using uninitialised struct cpu_thread

  When adding the sensors in occ_sensors_init, if the type is not OCC_SENSOR_LOC_CORE, then the loop to find 'c' will not be executed. Then c->pir is used for both of the the add_sensor_node calls below.

  This provides a default value of 0 instead.

- NX: Add NX coprocessor init opal call

  The read offset (4:11) in Receive FIFO control register is incremented by FIFO size whenever CRB read by NX. But the index in RxFIFO has to match with the corresponding entry in FIFO maintained by VAS in kernel. VAS entry is reset to 0 when opening the receive window during driver initialization. So when NX842 is reloaded or in kexec boot, possibility of mismatch between RxFIFO control register and VAS entries in kernel. It could cause CRB failure / timeout from NX.

  This patch adds nx_coproc_init opal call for kernel to initialize readOffset (4:11) and Queued (15:23) in RxFIFO control register.

- SLW: Remove stop1_lite and stop2_lite

  stop1_lite has been removed since it adds no additional benefit over stop0_lite. stop2_lite has been removed since currently it adds minimal benefit over stop2. However, the benefit is eclipsed by the time required to ungate the clocks

  Moreover, Lite states don't give up the SMT resources, can potentially have a performance impact on sibling threads.

  Since current OSs (Linux) aren't smart enough to make good decisions with these stop states, we're (temporarly) removing them from what we expose to the OS, the idea being to bring them back in a new DT representation so that only an OS that knows what to do will do things with them.

- cpu: Use STOP1 on POWER9 for idle/sleep inside OPAL

  The current code requests STOP3, which means it gets STOP2 in practice.

  STOP2 has proven to occasionally be unreliable depending on FW version and chip revision, it also requires a functional CME, so instead, let's use STOP1. The difference is rather minimum for something that is only used a few seconds during boot.

NPU2 (NVLink2 and OpenCAPI)

Since *skiboot-6.1-rc1*:

- capi: Select the correct IODA table entry for the mbt cache.

  With the current code, the capi mmio window is not correctly configured in the IODA table entry. The first entry (generally the non-prefetchable BAR) is overwrriten. This patch sets the capi window bar at the right place.

- npu2/hw-procedures: Fence bricks via NTL instead of MISC

  There are a couple of places we can set/unset fence for a brick:

  1. MISC register: NPU2_MISC_FENCE_STATE
  2. NTL register for the brick: NPU2_NTL_MISC_CFG1(ndev)

  Recent testing of ATS in combination with GPU reset has exposed a side effect of using (1); if fence is set for all six bricks, it triggers a sticky nmmu latch which prevents the NPU from getting ATR responses. This manifests as a hang in the tests.

  We have npu2_dev_fence_brick() which uses (1), and only two calls to it. Replace the call which sets fence with a write to (2). Remove the corresponding unset call entirely. It's unneeded because the procedures already do a progression from full fence to half to idle using (2).

- phb4/capp: Calculate STQ/DMA read engines based on link-width for PEC

  Presently in CAPI mode the number of STQ/DMA-read engines allocated on PEC2 for CAPP is fixed to 6 and 0-30 respectively irrespective of the PCI link width. These values are only suitable for x8 cards and quickly run out if a x16 card is plugged to a PEC2 attached slot. This usually manifests as CAPP reporting TLBI timeout due to these messages getting stalled due to insufficient STQs.

  To fix this we update enable_capi_mode() to check if PEC2 chiplet is in x16 mode and if yes then we allocate 4/0-47 STQ/DMA-read engines for the CAPP traffic.

  Fixes: 37ea3cfdc852 (present in v5.7-rc1)

- npu2: Use same compatible string for NVLink and OpenCAPI link nodes in device tree

  Currently, we distinguish between NPU links for NVLink devices and OpenCAPI devices through the use of two different compatible strings - ibm,npu-link and ibm,npu-link-opencapi.

  As we move towards supporting configurations with both NVLink and OpenCAPI devices behind a single NPU, we need to detect the device type as part of presence detection, which can't happen until well after the point where the HDAT or platform code has created the NPU device tree nodes. Changing a node's compatible string after it's been created is a bit ugly, so instead we should move the device type to a new property which we can add to the node later on.

  Get rid of the ibm,npu-link-opencapi compatible string, add a new ibm,npu-link-type property, and a helper function to check the link type. Add an "unknown" device type in preparation for later patches to detect device type dynamically.

  These device tree bindings are entirely internal to skiboot and are not consumed directly by Linux, so this shouldn't break anything (other than internal BML lab environments).

- occ: Add support for GPU presence detection

  On the Witherspoon platform, we need to distinguish between NVLink GPUs and OpenCAPI accelerators. In order to do this, we first need to find out whether the SXM2 socket is populated.

  On Witherspoon, the SXM2 socket's presence detection pin is only visible via I2C from the APSS, and thus can only be exposed to the host via the OCC. The OCC, per OCC Firmware Interface Specification for POWER9 version 0.22, now exposes this to skiboot through a field in the dynamic data shared memory.

  Add the necessary dynamic data changes required to read the version and GPU presence fields. Add a function, occ_get_gpu_presence(), that can be used to check GPU presence.

  If the OCC isn't reporting presence (old OCC firmware, or some other reason), we default to assuming there is a device present and wait until link training to fail.

  This will be used in later patches to fix up the NPU2 probe path for OpenCAPI support on Witherspoon.

- hw/npu2, core/hmi: Use NPU instead of NPU2 as log message prefix

  The NPU2{DBG,INF,ERR} macros use "NPU%d" as a prefix to identify messages relating to a particular NPU.

  It's slightly confusing to have per-NPU messages prefixed with "NPU0" or "NPU1" and NPU-generic messages prefixed with "NPU2". On some future system we could potentially have a NPU #2 in which case it'd be really confusing.

  Use NPU rather than NPU2 for NPU-generic log messages. There's no risk of confusion with the original npu.c code since that's only for P8.

Since *skiboot-6.0*:

- npu2: Reset NVLinks on hot reset

  This effectively fences GPU RAM on GPU reset so the host system does not have to crash every time we stop a KVM guest with a GPU passed through.

- npu2-opencapi: reduce number of retries to train the link

  We've been reliably training the opencapi link on the first attempt for quite a while. Furthermore, if it doesn't train on the first attempt, retries haven't been that useful. So let's reduce the number of attempts we do to train the link.

  2 retries = 3 attempts to train.

  Each (failed) training sequence costs about 3 seconds.

- opal/hmi: Display correct chip id while printing NPU FIRs.

  HMIs for NPU xstops are broadcasted to all chips. All cores on all the chips receive HMI. HMI handler correctly identifies and extracts the NPU FIR details from affected chip, but while printing FIR data it prints chip id and location code details of this_cpu()->chip_id which may not be correct. This patch fixes this issue.

- npu2-opencapi: Fix link state to report link down

  The PHB callback 'get_link_state' is always reporting the link width, irrespective of the link status and even when the link is down. It is causing too much work (and failures) when the PHB is probed during pci init. The fix is to look at the link status first and report the link as down when appropriate.

- npu2-opencapi: Cleanup traces printed during link training

  Now that links may train in parallel, traces shown during training can be all mixed up. So add a prefix to all the traces to clearly identify the chip and link the trace refers to:

  ```
  OCAPI[<chip id>:<link id>]: this is a very useful message
  ```

  The lower-level hardware procedures (npu2-hw-procedures.c) also print traces which would need work. But that code is being reworked to be better integrated with opencapi and nvidia, so leave it alone for now.

- npu2-opencapi: Train links on fundamental reset

  Reorder our link training steps so that they are executed on fundamental reset instead of during the initial setup. Skiboot always call a fundamental reset on all the PHBs during pci init.

  It is done through a state machine, similarly to what is done for 'real' PHBs.

  This is the first step for a longer term goal to be able to trigger an adapter reset from linux. We'll need the reset callbacks of the PHB to be defined. We have to handle the various delays differently, since a linux thread shouldn't stay stuck waiting in opal for too long.

- npu2-opencapi: Rework adapter reset

  Rework a bit the code to reset the opencapi adapter:

  - make clearer which i2c pin is resetting which device

  - break the reset operation in smaller chunks. This is really to prepare for a future patch.

  No functional changes.

- npu2-opencapi: Use presence detection

  Presence detection is not part of the opencapi specification. So each platform may choose to implement it the way it wants.

  All current platforms implement it through an i2c device where we can query a pin to know if a device is connected or not. ZZ and Zaius have a similar design and even use the same i2c information and pin numbers. However, presence detection on older ZZ planar (older than v4) doesn't work, so we don't activate it for now, until our lab systems are upgraded and it's better tested.

  Presence detection on witherspoon is still being worked on. It's shaping up to be quite different, so we may have to revisit the topic in a later patch.

Testing and CI

Since *skiboot-6.1-rc1*:

- test/qemu: start building qemu again, and use our built qemu for tests

  We need to use QEMU_BIN rather than QEMU as the makefiles define QEMU already.

- opal-ci: qemu: Use the powernv-3.0 branch

  This is based off the current development version of Qemu, and importantly it contains the patch that allows skiboot and Linux to clear the PCR that we require to boot.

5.1.126 skiboot-6.1-rc1

skiboot v6.1-rc1 was released on Friday June 22nd 2018. It is the first release candidate of skiboot 6.1, which will become the new stable release of skiboot following the 6.0 release, first released May 11th 2018.

Skiboot 6.1 will mark the basis for op-build v2.1.

skiboot v6.1-rc1 contains all bug fixes as of *skiboot-6.0.4*, and *skiboot-5.4.9* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

This release contains a lot of small cleanups and fixes all over the place, which is possibly a sign that we've shipped our big POWER9 GA release and now get to breathe for a moment to look at what we ended up with. Since this is a really small incremental release, there will unlikely be many release candidates.

Over skiboot 6.0, we have the following changes:

General changes and bug fixes

- GCC8 build fixes
- Add prepare_hbrt_update to hbrt interfaces

  Add placeholder support for prepare_hbrt_update call into hostboot runtime (opal-prd) code. This interface is only called as part of a concurrent code update on a FSP based system.

- cpu: Clear PCR SPR in opal_reinit_cpus()

  Currently if Linux boots with a non-zero PCR, things can go bad where some early userspace programs can take illegal instructions. This is being fixed in Linux, but in the mean time, we should cleanup in skiboot also.

- pci: Fix PCI_DEVICE_ID()

  The vendor ID is 16 bits not 8. This error leaves the top of the vendor ID in the bottom bits of the device ID, which resulted in e.g. a failure to run the PCI quirk for the AST VGA device.

- Quieten console output on boot

  We print out a whole bunch of things on boot, most of which aren't interesting, so we should *not* print them instead.

  Printing things like what CPUs we found and what PCI devices we found *are* useful, so continue to do that. But we don't need to splat out a bunch of things that are always going to be true.

- core/console: fix deadlock when printing with console lock held

  Some debugging options will print while the console lock is held, which is why the console lock is taken as a recursive lock. However console_write calls __flush_console, which will drop and re-take the lock non-recursively in some cases.

  Just set con_need_flush and return from __flush_console if we are holding the console lock already.

  This stack usage message (taken with this patch applied) could lead to a deadlock without this:

```
CPU 0000 lowest stack mark 11768 bytes left pc=300cb808 token=0
CPU 0000 Backtrace:
S: 0000000031c03370 R: 00000000300cb808   .list_check_node+0x1c
S: 0000000031c03410 R: 00000000300cb910   .list_check+0x38
S: 0000000031c034b0 R: 00000000300190ac   .try_lock_caller+0xb8
S: 0000000031c03540 R: 00000000300192e0   .lock_caller+0x80
S: 0000000031c03600 R: 0000000030012c70   .__flush_console+0x134
S: 0000000031c036d0 R: 00000000300130cc   .console_write+0x68
S: 0000000031c03780 R: 00000000300347bc   .vprlog+0xc8
S: 0000000031c03970 R: 0000000030034844   ._prlog+0x50
S: 0000000031c03a00 R: 00000000300364a4   .log_simple_error+0x74
S: 0000000031c03b90 R: 000000003004ab48   .occ_pstates_init+0x184
S: 0000000031c03d50 R: 000000003001480c   .load_and_boot_kernel+0x38c
S: 0000000031c03e30 R: 000000003001571c   .main_cpu_entry+0x62c
S: 0000000031c03f00 R: 0000000030002700   boot_entry+0x1c0
```

- opal-prd: Do not error out on first failure for soft/hard offline.

  The memory errors (CEs and UEs) that are detected as part of background memory scrubbing are reported by PRD asynchronously to opal-prd along with affected memory ranges. hservice_memory_error() converts these ranges into page granularity before hooking up them to soft/hard offline-ing infrastructure.

  But the current implementation of hservice_memory_error() does not hookup all the pages to soft/hard offlineing if any of the page offline action fails. e.g hard offline can fail for:

  - Pages that are not part of buddy managed pool.
  - Pages that are reserved by kernel using memblock_reserved()
  - Pages that are in use by kernel.

  But for the pages that are in use by user space application, the hard offline marks the page as hwpoison, sends SIGBUS signal to kill the affected application as recovery action and returns success.

  Hence, It is possible that some of the pages in that memory range are in use by application or free. By stopping on first error we loose the opportunity to hwpoison the subsequent pages which may be free or in use by application. This patch fixes this issue.

- libflash/blocklevel_write: Fix missing error handling

  Caught by scan-build, we seem to trap the errors in rc, but not take any recovery action during blocklevel_write.

I2C

- p8-i2c: fix wrong request status when a reset is needed

  If the bus is found in error state when starting a new request, the engine is reset and we enter recovery. However, once complete, the reset operation shows a status of complete in the status register. So any badly-timed called to check_status() will think the current top request is complete, even though it hasn't run yet.

  So don't update any request status while we are in recovery, as nothing useful for the request is supposed to happen in that state.

- p8-i2c: Remove force reset

  Force reset was added as an attempt to work around some issues with TPM devices locking up their I2C bus. In that particular case the problem was that the device would hold the SCL line down permanently due to a device firmware bug. The force reset doesn't actually do anything to alleviate the situation here, it just happens to reset the internal master state enough to make the I2C driver appear to work until something tries to access the bus again.

  On P9 systems with secure boot enabled there is the added problem of the "diagostic mode" not being supported on I2C masters A,B,C and D. Diagnostic mode allows the SCL and SDA lines to be driven directly by software. Without this force reset is impossible to implement.

  This patch removes the force reset functionality entirely since:

  a) it doesn't do what it's supposed to, and

  b) it's butt ugly code

  Additionally, turn p8_i2c_reset_engine() into p8_i2c_reset_port(). There's no need to reset every port on a master in response to an error that occurred on a specific port.

- libstb/i2c-driver: Bump max timeout

  We have observed some TPMs clock streching the I2C bus for signifigant amounts of time when processing commands. The same TPMs also have errata that can result in permernantly locking up a bus in response to an I2C transaction they don't understand. Using an excessively long timeout to prevent this in the field.

- hdata: Add TPM timeout workaround

  Set the default timeout for any bus containing a TPM to one second. This is needed to work around a bug in the firmware of certain TPMs that will clock strech the I2C port the for up to a second. Additionally, when the TPM is clock streching it responds to a STOP condition on the bus by bricking itself. Clearing this error requires a hard power cycle of the system since the TPM is powered by standby power.

- p8-i2c: Allow a per-port default timeout

  Add support for setting a default timeout for the I2C port to the device-tree. This is consumed by skiboot.

IPMI Watchdog

- ipmi-watchdog: Support handling re-initialization

  Watchdog resets can return an error code from the BMC indicating that the BMC watchdog was not initialized. Currently we abort skiboot due to a missing error handler. This patch implements handling re-initialization for the watchdog, automatically saving the last watchdog set values and re-issuing them if needed.

- ipmi-watchdog: The stop action should disable reset

  Otherwise it is possible for the reset timer to elapse and trigger the watchdog to wake back up. This doesn't affect the behavior of the system since we are providing a NONE action to the BMC. However we would like to avoid the action from taking place if possible.

- ipmi-watchdog: Add a flag to determine if we are still ticking

  This makes it easier for future changes to ensure that the watchdog stops ticking and doesn't requeue itself for execution in the background. This way it is safe for resets to be performed after the ticks are assumed to be stopped and it won't start the timer again.

- ipmi-watchdog: (prepare for) not disabling at shutdown

  The op-build linux kernel has been configured to support the ipmi watchdog. This driver will always handle the watchdog by either leaving it enabled if configured, or by disabling it during module load if no configuration is provided. This increases the coverage of the watchdog during the boot process. The watchdog should no longer be disabled at any point during skiboot execution.

  We're not enabling this by default yet as people can (and do, at least in development) mix and match old BOOTKERNEL with new skiboot and we don't want to break that too obviously.

- ipmi-watchdog: Don't reset the watchdog twice

  There is no clarification for why this change was needed, but presumably this is due to a buggy BMC implementation where the Watchdog Set command was processed concurrently or after the initial Watchdog Reset. This inversion would cause the watchdog to stop since the DONT_STOP bit was not set. Since we are now using the DONT_STOP bit during initialization, the watchdog should not be stopped even if an inversion occurs.

- ipmi-watchdog: Make it possible to set DONT_STOP

  The IPMI standard supports setting a DONT_STOP bit during an Watchdog Set operation. Most of the time we don't want to stop the Watchdog when updating the settings so we should be using this bit. This patch makes it possible for callers of set_wdt to prevent the watchdog from being stopped. This only changes the behavior of the watchdog during the initial settings update when initializing skiboot. The watchdog is no longer disabled and then immediately re-enabled.

- ipmi-watchdog: WD_POWER_CYCLE_ACTION -> WD_RESET_ACTION

  The IPMI specification denotes that action 0x1 is Host Reset and 0x3 is Host Power Cycle. Use the correct name for Reset in our watchdog code.

POWER8 platforms

- astbmc: Enable mbox depending on scratch reg

  P8 boxes can opt in for mbox pnor support if they set the scratch register bit to indicate it is supported.

Simulator platforms

- plat/qemu: add PNOR support

  To access the PNOR, OPAL/skiboot drives the BMC SPI controller using the iLPC2AHB device of the BMC SuperIO controller and accesses the flash contents using the LPC FW address space on which the PNOR is remapped.

  The QEMU PowerNV machine now integrates such models (SuperIO controller, iLPC2AHB device) and also a pseudo Aspeed SoC AHB memory space populated with the SPI controller registers (same model as for ARM). The AHB window giving access to the contents of the BMC SPI controller flash modules is mapped on the LPC FW address space.

  The change should be compatible for machine without PNOR support.

- external/mambo: Add support for readline if it exists

  Add support for tclreadline package if it is present. This patch loads the package and uses it when the simulation stops for any reason.

FSP based platforms

- Disable fast reboot on FSP IPL side change

  If FSP changes next IPL side, then disable fast reboot.

  sample output:

```
[  620.196442259,5] FSP: Got sysparam update, param ID 0xf0000007
[  620.196444501,5] CUPD: FW IPL side changed. Disable fast reboot
[  620.196445389,5] CUPD: Next IPL side : perm
```

- fsp/console: Always establish OPAL console API backend

  Currently we only call set_opal_console() to establish the backend used by the OPAL console API if we find at least one FSP serial port in HDAT.

  On systems where there is none (IPMI only), we fail to set it, causing the console code to try to use the dummy console causing an assertion failure during boot due to clashing on the device-tree node names.

  So always set it if an FSP is present

AST BMC based platforms

- AMI BMC: use 0x3a as OEM command

  The 0x3a OEM command is for IBM commands, while 0x32 was for AMI ones. Sometime in the P8 timeframe, AMI BMCs were changed to listen for our commands on either 0x32 or 0x3a. Since 0x3a is the direction forward, we'll use that, as P9 machines with AMI BMCs probably also want these to work, and let's not bet that 0x32 will continue to be okay.

- astbmc: Set romulus BMC type to OpenBMC
- platform/astbmc: Do not delete compatible property

  P9 onwards OPAL is building device tree for BMC based system using HDAT. We are populating bmc/compatible node with bmc version. Hence do not delete this property.

Utilities

- external/xscom-utils: Add python library for xscom access

  Patch adds a simple python library module for xscom access. It directly manipulate the '/access' file for scom read and write from debugfs 'scom' directory.

  Example on how to generate a getscom using this module:

```
from adu_scoms import *
getscom = GetSCom()
getscom.parse_args()
getscom.run_command()
```

  Sample output for above getscom.py:

```
# ./getscom.py -l
Chip ID  | Rev   | Chip type
---------|-------|-----------
00000008 | DD2.0 | P9 (Nimbus) processor
00000000 | DD2.0 | P9 (Nimbus) processor
```

- ffspart: Don't require user to create blank partitions manually

  Add '–allow-empty' which allows the filename for a given partition to be blank. If set ffspart will set that part of the PNOR file 'blank' and set ECC bits if required. Without this option behaviour is unchanged and ffspart will return an error if it can not find the partition file.

- pflash: Use correct prefix when installing

  pflash uses lowercase prefix when running make install in it's direcetory, but uppercase PREFIX when running it in shared. Use lowercase everywhere.

  With this the OpenBMC bitbake recipie can drop an out of tree patch it's been carrying for years.

POWER9

- occ-sensor: Avoid using uninitialised struct cpu_thread

  When adding the sensors in occ_sensors_init, if the type is not OCC_SENSOR_LOC_CORE, then the loop to find 'c' will not be executed. Then c->pir is used for both of the the add_sensor_node calls below.

  This provides a default value of 0 instead.

- NX: Add NX coprocessor init opal call

  The read offset (4:11) in Receive FIFO control register is incremented by FIFO size whenever CRB read by NX. But the index in RxFIFO has to match with the corresponding entry in FIFO maintained by VAS in kernel. VAS entry is reset to 0 when opening the receive window during driver initialization. So when NX842 is reloaded or in kexec boot, possibility of mismatch between RxFIFO control register and VAS entries in kernel. It could cause CRB failure / timeout from NX.

  This patch adds nx_coproc_init opal call for kernel to initialize readOffset (4:11) and Queued (15:23) in RxFIFO control register.

- SLW: Remove stop1_lite and stop2_lite

  stop1_lite has been removed since it adds no additional benefit over stop0_lite. stop2_lite has been removed since currently it adds minimal benefit over stop2. However, the benefit is eclipsed by the time required to ungate the clocks

  Moreover, Lite states don't give up the SMT resources, can potentially have a performance impact on sibling threads.

  Since current OSs (Linux) aren't smart enough to make good decisions with these stop states, we're (temporarly) removing them from what we expose to the OS, the idea being to bring them back in a new DT representation so that only an OS that knows what to do will do things with them.

- cpu: Use STOP1 on POWER9 for idle/sleep inside OPAL

  The current code requests STOP3, which means it gets STOP2 in practice.

  STOP2 has proven to occasionally be unreliable depending on FW version and chip revision, it also requires a functional CME, so instead, let's use STOP1. The difference is rather minimum for something that is only used a few seconds during boot.

NPU2 (NVLink2 and OpenCAPI)

- npu2: Reset NVLinks on hot reset

  This effectively fences GPU RAM on GPU reset so the host system does not have to crash every time we stop a KVM guest with a GPU passed through.

- npu2-opencapi: reduce number of retries to train the link

  We've been reliably training the opencapi link on the first attempt for quite a while. Furthermore, if it doesn't train on the first attempt, retries haven't been that useful. So let's reduce the number of attempts we do to train the link.

  2 retries = 3 attempts to train.

  Each (failed) training sequence costs about 3 seconds.

- opal/hmi: Display correct chip id while printing NPU FIRs.

  HMIs for NPU xstops are broadcasted to all chips. All cores on all the chips receive HMI. HMI handler correctly identifies and extracts the NPU FIR details from affected chip, but while printing FIR data it prints chip id and location code details of this_cpu()->chip_id which may not be correct. This patch fixes this issue.

- npu2-opencapi: Fix link state to report link down

  The PHB callback 'get_link_state' is always reporting the link width, irrespective of the link status and even when the link is down. It is causing too much work (and failures) when the PHB is probed during pci init. The fix is to look at the link status first and report the link as down when appropriate.

- npu2-opencapi: Cleanup traces printed during link training

  Now that links may train in parallel, traces shown during training can be all mixed up. So add a prefix to all the traces to clearly identify the chip and link the trace refers to:

  ```
  OCAPI[<chip id>:<link id>]: this is a very useful message
  ```

  The lower-level hardware procedures (npu2-hw-procedures.c) also print traces which would need work. But that code is being reworked to be better integrated with opencapi and nvidia, so leave it alone for now.

- npu2-opencapi: Train links on fundamental reset

  Reorder our link training steps so that they are executed on fundamental reset instead of during the initial setup. Skiboot always call a fundamental reset on all the PHBs during pci init.

  It is done through a state machine, similarly to what is done for 'real' PHBs.

  This is the first step for a longer term goal to be able to trigger an adapter reset from linux. We'll need the reset callbacks of the PHB to be defined. We have to handle the various delays differently, since a linux thread shouldn't stay stuck waiting in opal for too long.

- npu2-opencapi: Rework adapter reset

  Rework a bit the code to reset the opencapi adapter:

  - make clearer which i2c pin is resetting which device
  - break the reset operation in smaller chunks. This is really to prepare for a future patch.

  No functional changes.

- npu2-opencapi: Use presence detection

  Presence detection is not part of the opencapi specification. So each platform may choose to implement it the way it wants.

  All current platforms implement it through an i2c device where we can query a pin to know if a device is connected or not. ZZ and Zaius have a similar design and even use the same i2c information and pin numbers. However, presence detection on older ZZ planar (older than v4) doesn't work, so we don't activate it for now, until our lab systems are upgraded and it's better tested.

  Presence detection on witherspoon is still being worked on. It's shaping up to be quite different, so we may have to revisit the topic in a later patch.

5.1.127 skiboot-6.2

skiboot v6.2 was released on Friday December 14th 2018. It is the first release of skiboot 6.2, which becomes the new stable release of skiboot following the 6.1 release, first released July 11th 2018.

Skiboot 6.2 will mark the basis for op-build v2.2.

skiboot v6.2 contains all bug fixes as of *skiboot-6.0.14*, and *skiboot-5.4.10* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

This release has been a longer cycle than typical for a variety of reasons. It also contains a lot of cleanup work and minor bug fixes (much like skiboot 6.1 did).

Over skiboot 6.1, we have the following changes:

General

Since v6.2-rc2:

- i2c: Fix i2c request hang during opal init if timers are not checked

  If an i2c request cannot go through the first time, because the bus is found in error and need a reset or it's locked by the OCC for example, the underlying i2c implementation is using timers to manage the request. However during opal init, opal pollers may not be called, it depends in the context in which the i2c request is made. If the pollers are not called, the timers are not checked and we can end up with an i2c request which will not move foward and skiboot hangs.

  Fix it by explicitly checking the timers if we are waiting for an i2c request to complete and it seems to be taking a while.

Since v6.1:

- cpu: Quieten OS endian switch messages

  Users see these when loading an OS from Petitboot:

  ```
  [  119.486794100,5] OPAL: Switch to big-endian OS
  [  120.022302604,5] OPAL: Switch to little-endian OS
  ```

  Which is expected and doesn't provide any information the user can act on. Switch them to PR_INFO so they still appear in the log, but not on the serial console.

- Recognise signed VERSION partition

  A few things need to change to support a signed VERSION partition:

  - A signed VERSION partition will be 4K + SECURE_BOOT_HEADERS_SIZE (4K).
  - The VERSION partition needs to be loaded after secure/trusted boot is set up, and therefore after nvram_init().
  - Added to the trustedboot resources array.

  This also moves the ipmi_dt_add_bmc_info() call to after flash_dt_add_fw_version() since it adds info to ibm,firmware-versions.

- Run pollers in time_wait() when not booting

  This only bit us hard with hiomap in one scenario.

  Our OPAL API has been OPAL_POLL_EVENTS may be needed to make forward progress on ongoing operations, and the internal to skiboot API has been that time_wait() of a suitable time will run pollers (on at least one CPU) to help ensure forward progress can be made.

In a perfect world, interrupts are used but they may: a) be disabled, or b) the thing we're doing can't use interrupts because computers are generally terrible.

Back in 3db397ea5892a (circa 2015), we changed skiboot so that we'd run pollers only on the boot CPU, and not if we held any locks. This was to reduce the chance of programming code that could deadlock, as well as to ensure that we didn't just thrash all the cachelines for running pollers all over a large system during boot, or hard spin on the same locks on all secondary CPUs.

The problem arises if the OS we're booting makes an OPAL call early on, with interrupts disabled, that requires a poller to run to make forward progress. An example of this would be OPAL_WRITE_NVRAM early in Linux boot (where Linux sets up the partitions it wants) - something that occurs iff we've had to reformat NVRAM this boot (i.e. first boot or corrupted NVRAM).

The hiomap implementation should arguably *not* rely on synchronous IPMI messages, but this is a future improvement (as was for mbox before it). The mbox-flash code solved this problem by spinning on check_timers().

More generically though, the approach of running the pollers when no longer booting means we behave more in line with what the API is meant to be, rather than have this odd case of "time_wait() for a condition that could also be tripped by an interrupt works fine unless the OS is up and running but hasn't set interrupts up yet".

- ipmi: Reduce ipmi_queue_msg_sync() polling loop time to 10ms

  On a plain boot, this reduces the time spent in OPAL by ~170ms on p9dsu. This is due to hiomap (currently) using synchronous IPMI messages.

  It will also *significantly* reduce latency on runtime flash operations for hiomap, as we'll spend typically 10-20ms in OPAL rather than 100-200ms. It's not an ideal solution to that, but it's a quick and obvious win for jitter.

- core/device: NULL pointer dereference fix
- core/flash: NULL pointer dereference fixes
- core/cpu: Call memset with proper cpu_thread offset
- libflash: Add ipmi-hiomap, and prefer it for PNOR access

  ipmi-hiomap implements the PNOR access control protocol formerly known as "the mbox protocol" but uses IPMI instead of the AST LPC mailbox as a transport. As there is no-longer any mailbox involved in this alternate implementation the old protocol name is quite misleading, and so it has been renamed to "the hiomap protoocol" (Host I/O Mapping protocol). The same commands and events are used though this client-side implementation assumes v2 of the protocol is supported by the BMC.

  The code is a heavily-reworked copy of the mbox-flash source and is introduced this way to allow for the mbox implementation's eventual removal.

  mbox-flash should in theory be renamed to mbox-hiomap for consistency, but as it is on life-support effective immediately we may as well just remove it entirely when the time is right.

- opal/hmi: Handle early HMIs on thread0 when secondaries are still in OPAL.

  When primary thread receives a CORE level HMI for timer facility errors while secondaries are still in OPAL, thread 0 ends up in rendez-vous waiting for secondaries to get into hmi handling. This is because OPAL runs with MSR(EE=0) and hence HMIs are delayed on secondary threads until they are given to Linux OS. Fix this by adding a check for secondary state and force them in hmi handling by queuing job on secondary threads.

  I have tested this by injecting HDEC parity error very early during Linux kernel boot. Recovery works fine for non-TB errors. But if TB is bad at this very eary stage we already doomed.

  Without this patch we see:

```
[  285.046347408,7] OPAL: Start CPU 0x0843 (PIR 0x0843) -> 0x000000000000a83c
[  285.051160609,7] OPAL: Start CPU 0x0844 (PIR 0x0844) -> 0x000000000000a83c
[  285.055359021,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
```

(continues on next page)

(continued from previous page)

```
[  285.055361439,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:0:␣
↪TFMR(2e12002870e14000) Timer Facility Error
[  286.232183823,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 1 (sptr=0000ccc1)
[  287.409002056,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 2 (sptr=0000ccc1)
[  289.073820164,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 3 (sptr=0000ccc1)
[  290.250638683,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 1 (sptr=0000ccc2)
[  291.427456821,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 2 (sptr=0000ccc2)
[  293.092274807,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 3 (sptr=0000ccc2)
[  294.269092904,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 1 (sptr=0000ccc3)
[  295.445910944,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 2 (sptr=0000ccc3)
[  297.110728970,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 3 (sptr=0000ccc3)
```

After this patch:

```
[  259.401719351,7] OPAL: Start CPU 0x0841 (PIR 0x0841) -> 0x000000000000a83c
[  259.406259572,7] OPAL: Start CPU 0x0842 (PIR 0x0842) -> 0x000000000000a83c
[  259.410615534,7] OPAL: Start CPU 0x0843 (PIR 0x0843) -> 0x000000000000a83c
[  259.415444519,7] OPAL: Start CPU 0x0844 (PIR 0x0844) -> 0x000000000000a83c
[  259.419641401,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
[  259.419644124,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:0:␣
↪TFMR(2e12002870e04000) Timer Facility Error
[  259.419650678,7] HMI: Sending hmi job to thread 1
[  259.419652744,7] HMI: Sending hmi job to thread 2
[  259.419653051,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
[  259.419654725,7] HMI: Sending hmi job to thread 3
[  259.419654916,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
[  259.419658025,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
[  259.419658406,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:2:␣
↪TFMR(2e12002870e04000) Timer Facility Error
[  259.419663095,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:3:␣
↪TFMR(2e12002870e04000) Timer Facility Error
[  259.419655234,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:1:␣
↪TFMR(2e12002870e04000) Timer Facility Error
[  259.425109779,7] OPAL: Start CPU 0x0845 (PIR 0x0845) -> 0x000000000000a83c
[  259.429870681,7] OPAL: Start CPU 0x0846 (PIR 0x0846) -> 0x000000000000a83c
[  259.434549250,7] OPAL: Start CPU 0x0847 (PIR 0x0847) -> 0x000000000000a83c
```

- core/cpu: Fix memory allocation for job array

  fixes: 7a3f307e core/cpu: parallelise global CPU register setting jobs

  This bug would result in boot-hang on some configurations due to cpu_wait_job() endlessly waiting for the last bogus jobs[cpu->pir] pointer.

- i2c: Fix multiple-enqueue of the same request on NACK

  i2c_request_send() will retry the request if the error is a NAK, however it forgets to clear the “ud.done” flag. It will thus loop again and try to re-enqueue the same request causing internal request list corruption.

- i2c: Ensure ordering between i2c_request_send() and completion i2c_request_send loops waiting for a flag "uc.done" set by the completion routine, and then look for a result code also set by that same completion.

There is no synchronization, the completion can happen on another processor, so we need to order the stores to uc and the reads from uc so that uc.done is stored last and tested first using memory barriers.

- pci: Clarify power down logic

  Currently pci_scan_bus() unconditionally calls pci_slot_set_power_state() when it's finished scanning a bus. This is one of those things that makes you go "WHAT?" when you first see it and frankly the skiboot PCI code could do with less of that.

Fast Reboot

- fast-reboot: parallel memory clearing

  Arbitrarily pick 16GB as the unit of parallelism, and split up clearing memory into jobs and schedule them node-local to the memory (or on node 0 if we can't work that out because it's the memory up to SKIBOOT_BASE)

  This seems to cut at least ~40% time from memory zeroing on fast-reboot on a 256GB Boston system.

  For many systems, scanning PCI takes about as much time as zeroing all of RAM, so we may as well do them at the same time and cut a few seconds off the total fast reboot time.

- fast-reboot: verify firmware "romem" checksum

  This takes a checksum of skiboot memory after boot that should be unchanged during OS operation, and verifies it before allowing a fast reboot.

  This is not read-only memory from skiboot's point of view, beause it includes things like the opal branch table that gets populated during boot.

  This helps to improve the integrity of firmware against host and runtime firmware memory scribble bugs.

- core/fast-reboot: print the fast reboot disable reason

  Once things start to go wrong, disable_fast_reboot can be called a number of times, so make the first reason sticky, and also print it to the console at disable time. This helps with making sense of fast reboot disables.

- Add fast-reboot property to /ibm,opal DT node

  this means that if it's permanently disabled on boot, the test suite can pick that up and not try a fast reboot test.

Utilities

Since v6.2-rc2:

- opal-prd: hservice: Enable hservice->wakeup() in BMC

  This patch enables HBRT to use HYP special wakeup register in openBMC which until now was only used in FSP based machines.

  This patch also adds a capability check for opal-prd so that HBRT can decide if the host special wakeup register can be used.

- ffspart: Support flashing already ECC protected images

  We do this by assuming filenames with '.ecc' in them are already ECC protected.

  This solves a practical problem in transitioning op-build to use ffspart for pnor assembly rather than three perl scripts and a lot of XML.

  We also update the ffspart tests to take into account ECC requirements.

- ffspart: Increase MAX_LINE to above PATH_MAX

  Otherwise we saw failures in CI and the ~221 character paths Jankins likes to have.

- libflash/file: greatly increase perf of file_erase()

  Do 4096 byte chunks not 8 byte chunks. A ffspart invocation constructing a 64MB PNOR goes from a couple of seconds to ~0.1seconds with this patch.

Since v6.2-rc1: - libflash: Don't merge ECC-protected ranges

  Libflash currently merges contiguous ECC-protected ranges, but doesn't check that the ECC bytes at the end of the first and start of the second range actually match sanely. More importantly, if blocklevel_read() is called with a position at the start of a partition that is contained somewhere within a region that has been merged it will update the position assuming ECC wasn't being accounted for. This results in the position being somewhere well after the actual start of the partition which is incorrect.

  For now, remove the code merging ranges. This means more ranges must be held and checked however it prevents incorrectly reading ECC-correct regions like below:

```
[  174.334119453,7] FLASH: CAPP partition has ECC
[  174.437349574,3] ECC: uncorrectable error: ffffffffffffffff ff
[  174.437426306,3] FLASH: failed to read the first 0x1000 from CAPP␣
↪partition, rc 14
[  174.439919343,3] CAPP: Error loading ucode lid. index=201d1
```

- libflash: Restore blocklevel tests

  This fell out in f58be46 "libflash/test: Rewrite Makefile.check to improve scalability". Add it back in as test-blocklevel.

Since v6.1:

- pflash: Add –skip option for reading

  Add a –skip=N option to pflash to skip N number of bytes when reading. This would allow users to print the VERSION partition without the STB header by specifying the –skip=4096 argument, and it's a more generic solution rather than making pflash depend on secure/trusted boot code.

- xscom-utils: Rework getsram

  Allow specifying a file on the command line to read OCC SRAM data into. If no file is specified then we print it to stdout as text. This is a bit inconsistent, but it retains compatibility with the existing tool.

- xscom-utils/getsram: Make it work on P9

  The XSCOM base address of the OCC control registers changed slightly between P8 and P9. Fix this up and add a bit of PVR checking so we look in the right place.

- opal-prd: Fix opal-prd crash

  Presently callback function from HBRT uses r11 to point to target function pointer. r12 is garbage. This works fine when we compile with "-no-pie" option (as we don't use r12 to calculate TOC).

  As per ABIv2 : "r12 : Function entry address at global entry point"

  With "-pie" compilation option, we have to set r12 to point to global function entry point. So that we can calculate TOC properly.

  Crash log without this patch:

```
opal-prd[2864]: unhandled signal 11 at 0000000000029320 nip 00000 00102012830 lr␣
↪0000000102016890 code 1
```

Development and Debugging

Since v6.1-rc1: - Warn on long OPAL calls

Measure entry/exit time for OPAL calls and warn appropriately if the calls take too long (>100ms gets us a DEBUG log, > 1000ms gets us a warning).

Since v6.1:

- core/lock: Use try_lock_caller() in lock_caller() to capture owner

  Otherwise we can get reports of core/lock.c owning the lock, which is not helpful when tracking down ownership issues.

- core/flash: Emit a warning if Skiboot version doesn't match

  This means you'll get a warning that you've modified skiboot separately to the rest of the PNOR image, which can be useful in determining what firmware is actually running on a machine.

- gcov: link in ctors* as newer GCC doesn't group them all

  It seems that newer toolchains get us multiple ctors sections to link in rather than just one. If we discard them (as we were doing), then we don't have a working gcov build (and we get the "doesn't look sane" warning on boot).

- core/flash: Log return code when ffs_init() fails

  Knowing the return code is at least better than not knowing the return code.

- gcov: Fix building with GCC8
- travis/ci: rework Dockerfiles to produce build artifacts

  ubuntu-latest was also missing clang, as ubuntu-latest is closer to ubuntu 18.04 than 16.04

- cpu: add cpu_queue_job_on_node()

  Add a job scheduling API which will run the job on the requested chip_id (or return failure).

- opal-ci: Build old dtc version for fedora 28

  There are patches that will go into dtc to fix the issues we hit, but for the moment let's just build and use a slightly older version.

- mem_region: Merge similar allocations when dumping

  Currently we print one line for each allocation done at runtime when dumping the memory allocations. We do a few thousand allocations at boot so this can result in a huge amount of text being printed which is a) slow to print, and b) Can result in the log buffer overflowing which destroys otherwise useful information.

  This patch adds a de-duplication to this memory allocation dump by merging "similar" allocations (same location, same size) into one.

  Unfortunately, the algorithm used to do the de-duplication is quadratic, but considering we only dump the allocations in the event of a fatal error I think this is acceptable. I also did some benchmarking and found that on a ZZ it takes ~3ms to do a dump with 12k allocations. On a Zaius it's slightly longer at about ~10ms for 10k allocs. However, the difference there was due to the output being written to the UART.

  This patch also bumps the log level to PR_NOTICE. PR_INFO messages are suppressed at the default log level, which probably isn't something you want considering we only dump the allocations when we run out of skiboot heap space.

- core/lock: fix timeout warning causing a deadlock false positive

If a lock waiter exceeds the warning timeout, it prints a message while still registered as requesting the lock. Printing the message can take locks, so if one is held when the owner of the original lock tries to print a message, it will get a false positive deadlock detection, which brings down the system.

This can easily be hit when there is a lot of HMI activity from a KVM guest, where the timebase was not returned to host timebase before calling the HMI handler.

- hw/p8-i2c: Print the set error bits

  This is purely to save me from having to look it up every time someone gets an I2C error.

- init: Fix starting stripped kernel

  Currently if we try to run a raw/stripped binary kernel (ie. without the elf header) we crash with:

```
[    0.008757768,5] INIT: Waiting for kernel...
[    0.008762937,5] INIT: platform wait for kernel load failed
[    0.008768171,5] INIT: Assuming kernel at 0x20000000
[    0.008779241,3] INIT: ELF header not found. Assuming raw binary.
[    0.017047348,5] INIT: Starting kernel at 0x0, fdt at 0x3044b230 14339 bytes
[    0.017054251,0] FATAL: Kernel is zeros, can't execute!
[    0.017059054,0] Assert fail: core/init.c:590:0
[    0.017065371,0] Aborting!
```

  This is because we haven't set kernel_entry correctly in this path. This fixes it.

- cpu: Better output when waiting for a very long job

  Instead of printing at the end if the job took more than 1s, print in the loop every 30s along with a backtrace. This will give us some output if the job is deadlocked.

- lock: Fix interactions between lock dependency checker and stack checker

  The lock dependency checker does a few nasty things that can cause re-entrancy deadlocks in conjunction with the stack checker or in fact other debug tests.

  A lot of it revolves around taking a new lock (dl_lock) as part of the locking process.

  This tries to fix it by making sure we do not hit the stack checker while holding dl_lock.

  We achieve that in part by directly using the low-level __try_lock and manually unlocking on the dl_lock, and making some functions "nomcount".

  In addition, we mark the dl_lock as being in the console path to avoid deadlocks with the UART driver.

  We move the enabling of the deadlock checker to a separate config option from DEBUG_LOCKS as well, in case we chose to disable it by default later on.

- xscom-utils/adu_scoms.py: run 2to3 over it
- clang: -Wno-error=ignored-attributes

CI, testing, and utilities

Since v6.1-rc2:

- opal-ci: Drop fedora27, add fedora29
- ci: Bump Qemu version

  This moves the qemu version to qemu-powernv-for-skiboot-7 which is based on upstream's 3.1.0, and supports a Power9 machine.

  It also includes a fix for the skiboot XSCOM errors:

```
XSCOM: read error gcid=0x0 pcb_addr=0x1020013 stat=0x0
```

There is no modelling of the xscom behaviour but the reads/writes now succeed which is enough for skiboot to not error out.

- test: Update qemu arguments to use bmc simulator

  THe qemu skiboot platform as of 8340a9642bba ("plat/qemu: use the common OpenPOWER routines to initialize") uses the common aspeed BMC setup routines. This means a BT interface is always set up, and if the corresponding Qemu model is not present the timeout is 30 seconds.

  It looks like this every time an IPMI message is sent:

  ```
  BT: seq 0x9e netfn 0x06 cmd 0x31: Maximum queue length exceeded
  BT: seq 0x9d netfn 0x06 cmd 0x31: Removed from queue
  BT: seq 0x9f netfn 0x06 cmd 0x31: Maximum queue length exceeded
  BT: seq 0x9e netfn 0x06 cmd 0x31: Removed from queue
  BT: seq 0xa0 netfn 0x06 cmd 0x31: Maximum queue length exceeded
  BT: seq 0x9f netfn 0x06 cmd 0x31: Removed from queue
  ```

  Avoid this by adding the bmc simulator model to the Qemu powernv machine.

- ci: Add opal-utils to Debian unstable

  This puts a 'pflash' in the users PATH, allowing more test coverage of ffspart.

- ci: Drop P8 mambo from Debian unstable

  Debian Unstable has removed OpenSSL 1.0.0 from the repository so mambo no longer runs:

  ```
  /opt/ibm/systemsim-p8/bin/systemsim-pegasus: error while loading shared
  libraries: libcrypto.so.1.0.0: cannot open shared object file: No such
  file or directory
  ```

  By removing it from the container these tests will be automatically skipped.

  Tracked in https://github.com/open-power/op-build/issues/2519

- ci: Add dtc dependencies for rawhide

  Both F28 and Rawhide build their own dtc version. Rawhide was missing the required build deps.

- ci: Update Debian unstable packages

  This syncs Debian unstable with Ubuntu 18.04 in order to get the clang package. It also adds qemu to the Debian install, which makes sense Debian also has 2.12.

- ci: Use Ubuntu latest config for Debian unstable

  Debian unstable has the same GCOV issue with 8.2 as Ubuntu latest so it makes sense to share configurations there.

- ci: Disable GCOV builds in ubuntu-latest

  They are known to be broken with GCC 8.2: https://github.com/open-power/skiboot/issues/206

- ci: Update gcov comment in Fedora 28

- plat/qemu: fix platform initialization when the BT device is not present

  A QEMU PowerNV machine does not necessarily have a BT device. It needs to be defined on the command line with :

```
-device ipmi-bmc-sim,id=bmc0 -device isa-ipmi-bt,bmc=bmc0,irq=10
```

When the QEMU platform is initialized by skiboot, we need to check that such a device is present and if not, skip the AST initialization.

Since v6.1-rc1:

- travis: Coverity fixed their SSL cert
- opal-ci: Use ubuntu:rolling for Ubuntu latest image
- ffspart: Add test for eraseblock size
- ffspart: Add toc test
- hdata/test: workaround dtc bugs

  In dtc v1.4.5 to at least v1.4.7 there have been a few bugs introduced that change the layout of what's produced in the dts. In order to be immune from them, we should use the (provided) dtdiff utility, but we also need to run the dts we're diffing against through a dtb cycle in order to ensure we get the same format as what the hdat_to_dt to dts conversion will.

  This fixes a bunch of unit test failures on the version of dtc shipped with recent Linux distros such as Fedora 29.

Mambo Platform

- mambo: Merge PMEM_DISK and PMEM_VOLATILE code

  PMEM_VOLATILE and PMEM_DISK can't be used together and are basically copies of the same code.

  This merges the two and allows them used together. Same API is kept.

- hw/chiptod: test QUIRK_NO_CHIPTOD in opal_resync_timebase

  This allows some test coverage of deep stop states in Linux with Mambo.

- core/mem_region: mambo reserve kernel payload areas

  Mambo image payloads get overwritten by the OS and by fast reboot memory clearing because they have no region defined. Add them, which allows fast reboot to work.

Qemu platform

Since v6.2-rc2: - plat/qemu: use the common OpenPOWER routines to initialize

> Back in 2016, we did not have a large support of the PowerNV devices under QEMU and we were using our own custom ones. This has changed and we can now use all the common init routines of the OpenPOWER platforms.

Since v6.1:

- nx: Don't abort on missing NX when using a QEMU machine

  These don't have an NX node (and probably never will) as they don't provide any coprocessor. However, the DARN instruction works so this abort is unnecessary.

POWER8 Platforms

- SBE-p8: Do all sbe timer update with xscom lock held

  Without this, on some P8 platforms, we could (falsely) think the SBE timer had stalled getting the dreaded "timer stuck" message.

  The code was doing the mftb() to set the start of the timeout period while *not* holding the lock, so the 1ms timeout started sometime when somebody else had the xscom lock.

  The simple solution is to just do the whole routine holding the xscom lock, so do it that way.

Vesnin Platform

- platforms/astbmc/vesnin: Send list of PCI devices to BMC through IPMI

  Implements sending a list of installed PCI devices through IPMI protocol. Each PCI device description is sent as a standalone IPMI message. A list of devices can be gathered from separate messages using the session identifier. The session Id is an incremental counter that is updated at the start of synchronization session.

POWER9 Platforms

- STOP API: API conditionally supports 255 SCOM restore entries for each quad.
- hdata/i2c: Skip unknown device type

  Do not add unknown I2C devices to device tree.
- hdata/i2c: Add whitelisting for Host I2C devices

  Many of the devices that we get information about through HDAT are for use by firmware rather than the host operating system. This patch adds a boolean flag to hdat_i2c_info structure that indicates whether devices with a given purpose should be reserved for use inside of OPAL (or some other firmware component, such as the OCC).
- hdata/iohub: Fix Cumulus Hub ID number
- opal/hmi: Wakeup the cpu before reading core_fir

  When stop state 5 is enabled, reading the core_fir during an HMI can result in a xscom read error with xscom_read() returning an OPAL_XSCOM_PARTIAL_GOOD error code and core_fir value of all FFs. At present this return error code is not handled in decode_core_fir() hence the invalid core_fir value is sent to the kernel where it interprets it as a FATAL hmi causing a system check-stop.

  This can be prevented by forcing the core to wake-up using before reading the core_fir. Hence this patch wraps the call to read_core_fir() within calls to dctl_set_special_wakeup() and dctl_clear_special_wakeup().
- xive: Disable block tracker

  Due to some HW errata, the block tracking facility (performance optimisation for large systems) should be disabled on Nimbus chips. Disable it unconditionally for now.
- opal/hmi: Ignore debug trigger inject core FIR.

  Core FIR[60] is a side effect of the work around for the CI Vector Load issue in DD2.1. Usually this gets delivered as HMI with HMER[17] where Linux already ignores it. But it looks like in some cases we may happen to see CORE_FIR[60] while we are already in Malfunction Alert HMI (HMER[0]) due to other reasons e.g. CAPI recovery or NPU xstop. If that happens then just ignore it instead of crashing kernel as not recoverable.

- hdata: Make sure reserved node name starts with "ibm, "

  HDAT does not provide consistent label format for reserved memory label. Few starts with "ibm," while few other starts with component name.

- hdata: Fix dtc warnings

  Fix dtc warnings related to mcbist node.

```
Warning (reg_format): "reg" property in /xscom@623fc00000000/mcbist@1 has invalid␣
↪length (4 bytes) (#address-cells == 1, #size-cells == 1)
Warning (reg_format): "reg" property in /xscom@623fc00000000/mcbist@2 has invalid␣
↪length (4 bytes) (#address-cells == 1, #size-cells == 1)
Warning (reg_format): "reg" property in /xscom@603fc00000000/mcbist@1 has invalid␣
↪length (4 bytes) (#address-cells == 1, #size-cells == 1)
Warning (reg_format): "reg" property in /xscom@603fc00000000/mcbist@2 has invalid␣
↪length (4 bytes) (#address-cells == 1, #size-cells == 1)
```

  Ideally we should add proper xscom range here… but we are not getting that information in HDAT today. Lets fix warning until we get proper data in HDAT.

PHB4

- phb4: Generate checkstop on AIB ECC corr/uncorr for DD2.0 parts

  On DD2.0 parts, PCIe ECC protection is not warranted in the response data path. Thus, for these parts, we need to flag any ECC errors detected from the adjacent AIB RX Data path so the part can be replaced.

  This patch configures the FIRs so that we escalate these AIB ECC errors to a checkstop so the parts can be replaced.

- phb4: Reset pfir and nfir if new errors reported during ETU reset

  During fast-reboot new PEC errors can be latched even after ETU-Reset is asserted. This will result in values of variables nfir_cache and pfir_cache to be out of sync.

  During step-2 of CRESET nfir_cache and pfir_cache values are used to bring the PHB out of reset state. However if these variables are out as noted above of date the nfir/pfir registers are never reset completely and ETU still remains frozen.

  Hence this patch updates step-2 of phb4_creset to re-read the values of nfir/pfir registers to check if any new errors were reported after ETU-reset was asserted, report these new errors and reset the nfir/pfir registers. This should bring the ETU out of reset successfully.

- phb4: Disable nodal scoped DMA accesses when PB pump mode is enabled

  By default when a PCIe device issues a read request via the PHB it is first issued with nodal scope. When accessing GPU memory the NPU does not know at the time of response if the requested memory page is off node or not. Therefore every read of GPU memory by a PHB is retried with larger scope which introduces bandwidth and latency issues.

  On smaller boxes which have pump mode enabled nodal and group scoped reads are treated the same and both types of request are broadcast to one chip. Therefore we can avoid the retry by disabling nodal scope on the PHB for these boxes. On larger boxes nodal (single chip) and group (multiple chip) scoped reads are treated differently. Therefore we avoid disabling nodal scope on large boxes which have pump mode disabled to avoid all PHB requests being broadcast to multiple chips.

- phb4/capp: Only reset FIR bits that cause capp machine check

  During CAPP recovery do_capp_recovery_scoms() will reset the CAPP Fir register just after CAPP recovery is completed. This has an unintentional side effect of preventing PRD from analyzing and reporting this error.

If PRD tries to read the CAPP FIR after opal has already reset it, then it logs a critical error complaining "No active error bits found".

To prevent this from happening we update do_capp_recovery_scoms() to only reset fir bits that cause CAPP machine check (local xstop). This is done by reading the CAPP Fir Action0/1 & Mask registers and generating a mask which is then written on CAPP_FIR_CLEAR register.

- phb4: Check for RX errors after link training

  Some PHB4 PHYs can get stuck in a bad state where they are constantly retraining the link. This happens transparently to skiboot and Linux but will causes PCIe to be slow. Resetting the PHB4 clears the problem.

  We can detect this case by looking at the RX errors count where we check for link stability. This patch does this by modifying the link optimal code to check for RX errors. If errors are occurring we retrain the link irrespective of the chip rev or card.

  Normally when this problem occurs, the RX error count is maxed out at 255. When there is no problem, the count is 0. We chose 8 as the max rx errors value to give us some margin for a few errors. There is also a knob that can be used to set the error threshold for when we should retrain the link. ie

```
nvram -p ibm,skiboot --update-config phb-rx-err-max=8
```

- hw/phb4: Add a helper to dump the PELT-V

  The "Partitionable Endpoint Lookup Table (Vector)" is used by the PHB when processing EEH events. The PELT-V defines which PEs should be additionally frozen in the event of an error being flagged on a given PE. Knowing the state of the PELT-V is sometimes useful for debugging PHB issues so this patch adds a helper to dump it.

- hw/phb4: Print the PEs in the EEH dump in hex

  Linux always displays the PE number in hexidecimal while skiboot displays the PEST index (PE number) in decimal. This makes correlating errors between Skiboot and Linux more annoying than it should be so this patch makes Skiboot print the PEST number in hex.

- phb4: Reallocate PEC2 DMA-Read engines to improve GPU-Direct bandwidth

  We reallocate additional 16/8 DMA-Read engines allocated to stack0/1 on PEC2 respectively. This is needed to improve bandwidth available to the Mellanox CX5 adapter when trying to read GPU memory (GPU-Direct).

  If kernel cxl driver indicates a request to allocate maximum possible DMA read engines when calling enable_capi_mode() and card is attached to PEC2/stack0 slot then we assume its a Mellanox CX5 adapter. We then allocate additional 16/8 extra DMA read engines to stack0 and stack1 respectively on PEC2. This is done by populating the XPEC_PCI_PRDSTKOVR and XPEC_NEST_READ_STACK_OVERRIDE as suggested by the h/w team.

- phb4: Enable PHB MMIO-0/1 Bars only when mmio window exists

  Presently phb4_probe_stack() will always enable PHB MMIO0/1 windows even if they doesn't exist in phy_map. Hence we do some minor shuffling in the phb4_probe_stack() so that MMIO-0/1 Bars are only enabled if there corresponding MMIO window exists in the phy_map. In case phy_map for an mmio window is '0' we set the corresponding BAR register to '0'.

- hw/phb4: Use local_alloc for phb4 structures

  Struct phb4 is fairly heavyweight at 283664 bytes. On systems with 6x PHBs per socket this results in using 3.2MB of heap space the PHB structures alone. This is a fairly large chunk of our 12MB heap and on systems with particularly large PCIe topologies, or additional PHBs we can fail to boot because we cannot allocate space for the FDT blob.

  This patch switches to using local_alloc() for the PHB structures so they don't consume too large a portion of our 12MB heap space.

- phb4: Fix typo in disable lane eq code

  In this commit

  ```
  commit 737c0ba3d72b8aab05a765a9fc111a48faac0f75
  Author: Michael Neuling <mikey@neuling.org>
  Date:   Thu Feb 22 10:52:18 2018 +1100
  phb4: Disable lane eq when retrying some nvidia GEN3 devices
  ```

  We made a typo and set PH2 twice. This fixes it.

  It worked previously as if only phase 2 (PH2) is set it, skips phase 2 and phase 3 (PH3).

- phb4: Don’t probe a PHB if its garded

  Presently phb4_probe_stack() causes an exception while trying to probe a PHB if its garded. This causes skiboot to go into a reboot loop with following exception log:

  ```
   ***********************************************
   Fatal MCE at 000000003006ecd4   .probe_phb4+0x570
   CFAR : 00000000300b98a0
   <snip>
   Aborting!
  CPU 0018 Backtrace:
   S: 0000000031cc37e0 R: 000000003001a51c   ._abort+0x4c
   S: 0000000031cc3860 R: 0000000030028170   .exception_entry+0x180
   S: 0000000031cc3a40 R: 0000000000001f10 *
   S: 0000000031cc3c20 R: 000000003006ecb0   .probe_phb4+0x54c
   S: 0000000031cc3e30 R: 0000000030014ca4   .main_cpu_entry+0x5b0
   S: 0000000031cc3f00 R: 0000000030002700   boot_entry+0x1b8
  ```

  This is caused as phb4_probe_stack() will ignore all xscom read/write errors to enable PHB Bars and then tries to perform an mmio to read PHB Version registers that cause the fatal MCE.

  We fix this by ignoring the PHB probe if the first xscom_write() to populate the PHB Bar register fails, which indicates that there is something wrong with the PHB.

- phb4: Workaround PHB errata with CFG write UR/CA errors

  If the PHB encounters a UR or CA status on a CFG write, it will incorrectly freeze the wrong PE. Instead of using the PE# specified in the CONFIG_ADDRESS register, it will use the PE# of whatever MMIO occurred last.

  Work around this disabling freeze on such errors

- phb4: Handle allocation errors in phb4_eeh_dump_regs()

  If the zalloc fails (and it can be a rather large allocation), we will overwite memory at 0 instead of failing.

- phb4: Don’t try to access non-existent PEST entries

  In a POWER9 chip, some PHB4s have 256 PEs, some have 512.

  Currently, the diagnostics code retrieves 512 unconditionally, which is wrong and causes us to incorrectly report bogus values for the “high” PEs on the small PHBs.

  Use the actual number of implemented PEs instead

CAPI2

- phb4/capp: Use link width to allocate STQ engines to CAPP

Update phb4_init_capp_regs() to allocates STQ Engines to CAPP/PEC2 based on link width instead of always assuming it to x8.

Also re-factor the function slightly to evaluate the link-width only once and cache it so that it can also be used to allocate DMA read engines.

- phb4/capp: Update DMA read engines set in APC_FSM_READ_MASK based on link-width

  Commit 47c09cdfe7a3("phb4/capp: Calculate STQ/DMA read engines based on link-width for PEC") update the CAPP init sequence by calculating the needed STQ/DMA-read engines based on link width and populating it in XPEC_NEST_CAPP_CNTL register. This however needs to be synchronized with the value set in CAPP APC FSM Read Machine Mask Register.

  Hence this patch update phb4_init_capp_regs() to calculate the link width of the stack on PEC2 and populate the same values as previously populated in PEC CAPP_CNTL register.

- capp: Fix the capp recovery timeout comparison

  The current capp recovery timeout control loop in do_capp_recovery_scoms() uses a wrong comparison for return value of tb_compare(). This may cause do_capp_recovery_scoms() to report an timeout earlier than the 168ms stipulated time.

  The patch fixes this by updating the loop timeout control branch in do_capp_recovery_scoms() to use the correct enum tb_cmpval.

- phb4: Disable 32-bit MSI in capi mode

  If a capi device does a DMA write targeting an address lower than 4GB, it does so through a 32-bit operation, per the PCI spec. In capi mode, the first TVE entry is configured in bypass mode, so the address is valid. But with any (bad) luck, the address could be 0xFFFFxxxx, thus looking like a 32-bit MSI.

  We currently enable both 32-bit and 64-bit MSIs, so the PHB will interpret the DMA write as a MSI, which very likely results in an EEH (MSI with a bad payload size).

  We can fix it by disabling 32-bit MSI when switching the PHB to capi mode. Capi devices are 64-bit.

NVLINK2

Since v6.2-rc2: - Add purging CPU L2 and L3 caches into NPU hreset.

If a GPU is passed through to a guest and the guest unexpectedly terminates, there can be cache lines in CPUs that belong to the GPU. So purge the caches as part of the reset sequence. L1 is write through, so doesn't need to be purged.

The sequence to purge the L2 and L3 caches from the hw team:

L2 purge: 1. initiate purge

```
putspy pu.ex EXP.L2.L2MISC.L2CERRS.PRD_PURGE_CMD_TYPE L2CAC_FLUSH -all
putspy pu.ex EXP.L2.L2MISC.L2CERRS.PRD_PURGE_CMD_TRIGGER ON -all
```

2. check this is off in all caches to know purge completed

   ```
   getspy pu.ex EXP.L2.L2MISC.L2CERRS.PRD_PURGE_CMD_REG_BUSY -all
   ```

3.
   ```
   putspy pu.ex EXP.L2.L2MISC.L2CERRS.PRD_PURGE_CMD_TRIGGER OFF -all
   ```

L3 purge: 1. Start the purge:

```
putspy pu.ex EXP.L3.L3_MISC.L3CERRS.L3_PRD_PURGE_TTYPE FULL_PURGE -all
putspy pu.ex EXP.L3.L3_MISC.L3CERRS.L3_PRD_PURGE_REQ ON -all
```

2. Ensure that the purge has completed by checking the status bit:

   ```
   getspy pu.ex EXP.L3.L3_MISC.L3CERRS.L3_PRD_PURGE_REQ -all
   ```

   You should see it say OFF if it's done:

   ```
   p9n.ex k0:n0:s0:p00:c0
   EXP.L3.L3_MISC.L3CERRS.L3_PRD_PURGE_REQ
   OFF
   ```

- npu2: Return sensible PCI error when not frozen

  The current kernel calls OPAL_PCI_EEH_FREEZE_STATUS with an uninitialized @pci_error_type parameter and then analyzes it even if the OPAL call returned OPAL_SUCCESS. This is results in unexpected EEH events and NPU freezes.

  This initializes @pci_error_type and @severity to known safe values.

- npu2: Advertise correct TCE page size

  The P9 NPU workbook says that only 4K/64K/16M/256M page size are supported and in fact npu2_map_pe_dma_window() supports just these but in absence of the "ibm,supported-tce-sizes" property Linux assumes the default P9 PHB4 page sizes - 4K/64K/2M/1G - so when Linux tries 2M/1G TCEs, we get lots of "Unexpected TCE size" from npu2_tce_kill().

  This advertises TCE page sizes so Linux could handle it correctly, i.e. fall back to 4K/64K TCEs.

Since v6.1:

- npu2: Add support for relaxed-ordering mode

  Some device drivers support out of order access to GPU memory. This does not affect the CPU view of memory but it does affect the GPU view of memory. It should only be enabled if the GPU driver has requested it.

  Add OPAL APIs allowing the driver to query relaxed ordering state or request it to be set for a device. Current hardware only allows relaxed ordering to be enabled per PCIe root port. So the code here doesn't enable relaxed ordering until it has been explicitly requested for every device on the port.

- Add the other 7 ATSD registers to the device tree.
- npu2/hw-procedures: Don't open code NPU2_NTL_MISC_CFG2_BRICK_ENABLE

  Name this bit properly. There's a lot more cleanup like this to be done, but I'm catching this one now as part of some related changes.

- npu2/hw-procedures: Enable parity and credit overflow checks

  Enable these error checking features by setting the appropriate bits in our one-off initialization of each "NTL Misc Config 2" register.

  The exception is NDL RX parity checking, which should be disabled during the link training procedures.

- npu2: Use correct kill type for TCE invalidation

  kill_type is enum of OPAL_PCI_TCE_KILL_PAGES, OPAL_PCI_TCE_KILL_PE, OPAL_PCI_TCE_KILL_ALL and phb4_tce_kill() gets it right but npu2_tce_kill() uses OPAL_PCI_TCE_KILL which is an OPAL API token.

  This fixes an obvious mistype.

OpenCAPI

Since v6.2-rc1:

- npu2-opencapi: Log extra information on link training failure
- npu2-opencapi: Detect if link trained in degraded mode

Since v6.1:

- Support OpenCAPI on Witherspoon platform
- npu2-opencapi: Enable presence detection on ZZ

  Presence detection for opencapi adapters was broken for ZZ planars v3 and below. All ZZ systems currently used in the lab have had their planar upgraded, so we can now remove the override we had to force presence and activate presence detection. Which should improve boot time.

  Considering the state of opal support on ZZ, this is really only for lab usage on BML. The opencapi enablement team has okay'd the change. In the unlikely case somebody tries opencapi on an old ZZ, the presence detection through i2c will show that no adapter is present and skiboot won't try to access or train the link.

- npu2-opencapi: Don't send commands to NPU when link is down

  Even if an opencapi link is down, we currently always try to issue a config read operation when probing for PCI devices, because of the default scan map used for an opencapi PHB. The config operation fails, as expected, but it can also raise a FIR bit and trigger an HMI.

  For opencapi, there's no root device like for a "normal" PCI PHB, so there's no reason to do the config operation. To fix it, we keep the scan map blank by default, and only add a device once the link is trained.

- opal/hmi: Catch NPU2 HMIs for opencapi

  HMIs for NPU2 are filtered with the 'compatible' string of the PHB, so add opencapi to the mix.

- occ: Wait if OCC GPU presence status not immediately available

  It takes a few seconds for the OCC to set everything up in order to read GPU presence. At present, we try to kick off OCC initialisation as early as possible to maximise the time it has to read GPU presence.

  Unfortunately sometimes that's not enough, so add a loop in occ_get_gpu_presence() so that on the first time we try to get GPU presence we keep trying for up to 2 seconds. Experimentally this seems to be adequate.

- hw/npu2-hw-procedures: Enable RX auto recal on OpenCAPI links

  The RX_RC_ENABLE_AUTO_RECAL flag is required on OpenCAPI but not NVLink.

  Traditionally, Hostboot sets this value according to the machine type. However, now that Witherspoon supports both NVLink and OpenCAPI, it can't tell whether or not a link is OpenCAPI.

  So instead, set it in skiboot, where it will only be triggered after we've done device detection and found an OpenCAPI device.

- hw/npu2-opencapi: Fix setting of supported OpenCAPI templates

  In opal_npu_tl_set(), we made a typo that means the OPAL_NPU_TL_SET call may not clear the enable bits for templates that were previously enabled but are now disabled.

  Fix the typo so we clear NPU2_OTL_CONFIG1_TX_TEMP2_EN as well as TEMP{1,3}_EN.

Barreleye G2 and Zaius platforms

- zaius: Add a slot table

- zaius: Add slots for the Barreleye G2 HDD rack

  The Barreleye G2 is distinct from the Zaius in that it features a 24 Bay NVMe/SATA HDD rack. To provide meaningful slot names for each NVMe device we need to define a slot table for the NVMe capable HDD bays.

  Unfortunately this isn't straightforward because the PCIe path to the NVMe devices isn't fixed. The PCIe topology is something like: P9 -> HBA card -> 9797 switch -> 20x NVMe HDD slots

  The 9797 switch is partitioned into two (or four) virtual switches which allow multiple HBA cards to be used (e.g. one per socket). As a result the exact BDFN of the ports will vary depending on how the system is configured.

  That said, the virtual switch configuration of the 9797 does not change the device and function numbers of the switch downports. This means that we can define a single slot table that maps switch ports to the NVMe bay names.

  Unfortunately we still need to guess which bus to use this table on, so we assume that any switch downport we find with the PEX9797 VDID is part of the 9797 that supports the HDD rack.

FSP based platforms (firenze and ZZ)

Since v6.2-rc1: - platform/firenze: Fix branch-to-null crash

> When the bus alloc and free methods were removed we missed a case in the Firenze platform slot code that relied on the the bus-specific method to the bus pointer in the request structure. This results in a branch-to-null during boot and a crash. This patch fixes it by initialising it manually here.

Since v6.1:

- phb4/capp: Update the expected Eye-catcher for CAPP ucode lid

  Currently on a FSP based P9 system load_capp_code() expects CAPP ucode lid header to have eye-catcher magic of 'CAPPPSLL'. However skiboot currently supports CAPP ucode only lids that have a eye-catcher magic of 'CAPPLIDH'. This prevents skiboot from loading the ucode with this error message:

  ```
  CAPP: ucode header invalid
  ```

  We fix this issue by updating load_capp_ucode() to use the eye-catcher value of 'CAPPLIDH' instead of 'CAPPPSLL'.

- FSP: Improve Reset/Reload log message

  Below message is confusing. Lets make it clear.

  FSP sends "R/R complete notification" whenever there is a dump. We use *flag* to identify whether its its R/R completion -OR- just new dump notification.

  ```
  [  483.406351956,6] FSP: SP says Reset/Reload complete
  [  483.406354278,5] DUMP: FipS dump available. ID = 0x1a00001f [size: 6367640␣
  ↪bytes]
  [  483.406355968,7]   A Reset/Reload was NOT done
  ```

Witherspoon platform

- platforms/astbmc/witherspoon: Implement OpenCAPI support

  OpenCAPI on Witherspoon is slightly more involved than on Zaius and ZZ, due to the OpenCAPI links using the SXM2 connectors that are used for NVLink GPUs.

This patch adds the regular OpenCAPI platform information, and also a Witherspoon-specific presence detection callback that uses the previously added OCC GPU presence detection to figure out the device types plugged into each SXM2 socket.

The SXM2 connectors are capable of carrying 2 OpenCAPI links, and future OpenCAPI devices are expected to make use of this. However, we don't yet support ganged links and the various implications that has for handling things like device reset, so for now, we only enable 1 brick per device.

Contributors

The v6.2 release of skiboot contains 240 changesets from 28 developers, working for 2 employers. A total of 9146 lines were added, and 2610 removed (delta 6536).

Developers with the most changesets

Developer	#	%
Stewart Smith	58	(24.2%)
Andrew Jeffery	30	(12.5%)
Oliver O'Halloran	27	(11.2%)
Joel Stanley	17	(7.1%)
Vaibhav Jain	14	(5.8%)
Benjamin Herrenschmidt	12	(5.0%)
Frederic Barrat	11	(4.6%)
Nicholas Piggin	11	(4.6%)
Andrew Donnellan	10	(4.2%)
Vasant Hegde	9	(3.8%)
Reza Arbab	8	(3.3%)
Samuel Mendoza-Jonas	5	(2.1%)
Alexey Kardashevskiy	4	(1.7%)
Michael Neuling	4	(1.7%)
Prem Shanker Jha	3	(1.2%)
Cédric Le Goater	2	(0.8%)
Rashmica Gupta	2	(0.8%)
Mahesh J Salgaonkar	2	(0.8%)
Alistair Popple	2	(0.8%)
Shilpasri G Bhat	1	(0.4%)
Adriana Kobylak	1	(0.4%)
Madhavan Srinivasan	1	(0.4%)
Artem Senichev	1	(0.4%)
Russell Currey	1	(0.4%)
Vaidyanathan Srinivasan	1	(0.4%)
Cyril Bur	1	(0.4%)
Jeremy Kerr	1	(0.4%)
Michael Ellerman	1	(0.4%)

Developers with the most changed lines

Developer	#	%
Andrew Jeffery	2861	(29.3%)
Stewart Smith	1891	(19.4%)
Prem Shanker Jha	1046	(10.7%)
Andrew Donnellan	799	(8.2%)
Oliver O'Halloran	649	(6.6%)
Reza Arbab	441	(4.5%)
Nicholas Piggin	412	(4.2%)
Vaibhav Jain	278	(2.8%)
Cédric Le Goater	250	(2.6%)
Frederic Barrat	168	(1.7%)
Rashmica Gupta	161	(1.6%)
Joel Stanley	152	(1.6%)
Benjamin Herrenschmidt	138	(1.4%)
Artem Senichev	101	(1.0%)
Samuel Mendoza-Jonas	83	(0.9%)
Michael Neuling	82	(0.8%)
Michael Ellerman	61	(0.6%)
Mahesh J Salgaonkar	50	(0.5%)
Vasant Hegde	44	(0.5%)
Alexey Kardashevskiy	32	(0.3%)
Adriana Kobylak	29	(0.3%)
Alistair Popple	18	(0.2%)
Shilpasri G Bhat	4	(0.0%)
Madhavan Srinivasan	3	(0.0%)
Cyril Bur	3	(0.0%)
Jeremy Kerr	3	(0.0%)
Russell Currey	2	(0.0%)
Vaidyanathan Srinivasan	2	(0.0%)

Developers with the most lines removed

Developer	#	%
Cédric Le Goater	205	(7.9%)
Samuel Mendoza-Jonas	8	(0.3%)
Shilpasri G Bhat	1	(0.0%)

Developers with the most signoffs

Developer	#	%
Stewart Smith	182	(95.3%)
Alistair Popple	3	(1.6%)
Akshay Adiga	2	(1.0%)
Christophe Lombard	1	(0.5%)
Ryan Grimm	1	(0.5%)
Michael Neuling	1	(0.5%)
Mahesh J Salgaonkar	1	(0.5%)
Total	191	

Developers with the most reviews

Developer	#	%
Andrew Donnellan	15	(19.7%)
Frederic Barrat	11	(14.5%)
Oliver O'Halloran	9	(11.8%)
Alistair Popple	8	(10.5%)
Vasant Hegde	5	(6.6%)
Samuel Mendoza-Jonas	4	(5.3%)
Christophe Lombard	3	(3.9%)
Gregory S. Still	3	(3.9%)
Mahesh J Salgaonkar	2	(2.6%)
RANGANATHPRASAD G. BRAHMASAMUDRA	2	(2.6%)
Jennifer A. Stofer	2	(2.6%)
AMIT J. TENDOLKAR	2	(2.6%)
Christian R. Geddes	2	(2.6%)
Cédric Le Goater	1	(1.3%)
Shilpasri G Bhat	1	(1.3%)
Daniel M. Crowell	1	(1.3%)
Alexey Kardashevskiy	1	(1.3%)
Joel Stanley	1	(1.3%)
Vaibhav Jain	1	(1.3%)
Nicholas Piggin	1	(1.3%)
Andrew Jeffery	1	(1.3%)
Total	76	

Developers with the most test credits

Developer	#	%
Jenkins Server	3	(12.0%)
Cronus HW CI	3	(12.0%)
Hostboot CI	3	(12.0%)
Jenkins OP Build CI	3	(12.0%)
FSP CI Jenkins	3	(12.0%)
Jenkins OP HW	3	(12.0%)
Vasant Hegde	2	(8.0%)
Andrew Donnellan	1	(4.0%)
Oliver O'Halloran	1	(4.0%)
Andrew Jeffery	1	(4.0%)
HWSV CI	1	(4.0%)
Artem Senichev	1	(4.0%)
Total	25	

Developers who gave the most tested-by credits

Developer	#	%
Prem Shanker Jha	19	(76.0%)
Frederic Barrat	2	(8.0%)
Andrew Jeffery	1	(4.0%)
Vaibhav Jain	1	(4.0%)
Stewart Smith	1	(4.0%)
Benjamin Herrenschmidt	1	(4.0%)

Developers with the most report credits

Developer	#	%
Vasant Hegde	2	(25.0%)
Frederic Barrat	1	(12.5%)
Dawn Sylvia	1	(12.5%)
Meng Li	1	(12.5%)
Tyler Seredynski	1	(12.5%)
Pridhiviraj Paidipeddi	1	(12.5%)
Stephanie Swanson	1	(12.5%)

Developers who gave the most report credits

Developer	#	%
Stewart Smith	2	(25.0%)
Vaidyanathan Srinivasan	2	(25.0%)
Vasant Hegde	1	(12.5%)
Vaibhav Jain	1	(12.5%)
Andrew Donnellan	1	(12.5%)
Michael Neuling	1	(12.5%)

Employers with the most hackers

Developer	#	%
IBM	27	(96.4%)
YADRO	1	(3.6%)

5.1.128 skiboot-6.2-rc1

skiboot v6.2-rc1 was released on Monday November 19th 2018. It is the first release candidate of skiboot 6.2, which will become the new stable release of skiboot following the 6.1 release, first released July 11th 2018.

Skiboot 6.2 will mark the basis for op-build v2.2.

skiboot v6.2-rc1 contains all bug fixes as of *skiboot-6.0.13*, and *skiboot-5.4.10* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

This release has been a longer cycle than typical for a variety of reasons. It also contains a lot of cleanup work and minor bug fixes (much like skiboot 6.1 did).

Over skiboot 6.1, we have the following changes:

General

- cpu: Quieten OS endian switch messages

  Users see these when loading an OS from Petitboot:

  ```
  [  119.486794100,5] OPAL: Switch to big-endian OS
  [  120.022302604,5] OPAL: Switch to little-endian OS
  ```

  Which is expected and doesn't provide any information the user can act on. Switch them to PR_INFO so they still appear in the log, but not on the serial console.

- Recognise signed VERSION partition

  A few things need to change to support a signed VERSION partition:

  - A signed VERSION partition will be 4K + SECURE_BOOT_HEADERS_SIZE (4K).
  - The VERSION partition needs to be loaded after secure/trusted boot is set up, and therefore after nvram_init().
  - Added to the trustedboot resources array.

This also moves the ipmi_dt_add_bmc_info() call to after flash_dt_add_fw_version() since it adds info to ibm,firmware-versions.

- Run pollers in time_wait() when not booting

  This only bit us hard with hiomap in one scenario.

  Our OPAL API has been OPAL_POLL_EVENTS may be needed to make forward progress on ongoing operations, and the internal to skiboot API has been that time_wait() of a suitable time will run pollers (on at least one CPU) to help ensure forward progress can be made.

  In a perfect world, interrupts are used but they may: a) be disabled, or b) the thing we're doing can't use interrupts because computers are generally terrible.

  Back in 3db397ea5892a (circa 2015), we changed skiboot so that we'd run pollers only on the boot CPU, and not if we held any locks. This was to reduce the chance of programming code that could deadlock, as well as to ensure that we didn't just thrash all the cachelines for running pollers all over a large system during boot, or hard spin on the same locks on all secondary CPUs.

  The problem arises if the OS we're booting makes an OPAL call early on, with interrupts disabled, that requires a poller to run to make forward progress. An example of this would be OPAL_WRITE_NVRAM early in Linux boot (where Linux sets up the partitions it wants) - something that occurs iff we've had to reformat NVRAM this boot (i.e. first boot or corrupted NVRAM).

  The hiomap implementation should arguably *not* rely on synchronous IPMI messages, but this is a future improvement (as was for mbox before it). The mbox-flash code solved this problem by spinning on check_timers().

  More generically though, the approach of running the pollers when no longer booting means we behave more in line with what the API is meant to be, rather than have this odd case of "time_wait() for a condition that could also be tripped by an interrupt works fine unless the OS is up and running but hasn't set interrupts up yet".

- ipmi: Reduce ipmi_queue_msg_sync() polling loop time to 10ms

  On a plain boot, this reduces the time spent in OPAL by ~170ms on p9dsu. This is due to hiomap (currently) using synchronous IPMI messages.

  It will also *significantly* reduce latency on runtime flash operations for hiomap, as we'll spend typically 10-20ms in OPAL rather than 100-200ms. It's not an ideal solution to that, but it's a quick and obvious win for jitter.

- core/device: NULL pointer dereference fix
- core/flash: NULL pointer dereference fixes
- core/cpu: Call memset with proper cpu_thread offset
- libflash: Add ipmi-hiomap, and prefer it for PNOR access

  ipmi-hiomap implements the PNOR access control protocol formerly known as "the mbox protocol" but uses IPMI instead of the AST LPC mailbox as a transport. As there is no-longer any mailbox involved in this alternate implementation the old protocol name is quite misleading, and so it has been renamed to "the hiomap protoocol" (Host I/O Mapping protocol). The same commands and events are used though this client-side implementation assumes v2 of the protocol is supported by the BMC.

  The code is a heavily-reworked copy of the mbox-flash source and is introduced this way to allow for the mbox implementation's eventual removal.

  mbox-flash should in theory be renamed to mbox-hiomap for consistency, but as it is on life-support effective immediately we may as well just remove it entirely when the time is right.

- opal/hmi: Handle early HMIs on thread0 when secondaries are still in OPAL.

  When primary thread receives a CORE level HMI for timer facility errors while secondaries are still in OPAL, thread 0 ends up in rendez-vous waiting for secondaries to get into hmi handling. This is because OPAL runs with MSR(EE=0) and hence HMIs are delayed on secondary threads until they are given to Linux OS. Fix this by adding a check for secondary state and force them in hmi handling by queuing job on secondary threads.

I have tested this by injecting HDEC parity error very early during Linux kernel boot. Recovery works fine for non-TB errors. But if TB is bad at this very eary stage we already doomed.

Without this patch we see:

```
[  285.046347408,7] OPAL: Start CPU 0x0843 (PIR 0x0843) -> 0x000000000000a83c
[  285.051160609,7] OPAL: Start CPU 0x0844 (PIR 0x0844) -> 0x000000000000a83c
[  285.055359021,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
[  285.055361439,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:0:␣
↪TFMR(2e12002870e14000) Timer Facility Error
[  286.232183823,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 1 (sptr=0000ccc1)
[  287.409002056,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 2 (sptr=0000ccc1)
[  289.073820164,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 3 (sptr=0000ccc1)
[  290.250638683,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 1 (sptr=0000ccc2)
[  291.427456821,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 2 (sptr=0000ccc2)
[  293.092274807,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 3 (sptr=0000ccc2)
[  294.269092904,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 1 (sptr=0000ccc3)
[  295.445910944,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 2 (sptr=0000ccc3)
[  297.110728970,3] HMI: Rendez-vous stage 1 timeout, CPU 0x844 waiting for␣
↪thread 3 (sptr=0000ccc3)
```

After this patch:

```
[  259.401719351,7] OPAL: Start CPU 0x0841 (PIR 0x0841) -> 0x000000000000a83c
[  259.406259572,7] OPAL: Start CPU 0x0842 (PIR 0x0842) -> 0x000000000000a83c
[  259.410615534,7] OPAL: Start CPU 0x0843 (PIR 0x0843) -> 0x000000000000a83c
[  259.415444519,7] OPAL: Start CPU 0x0844 (PIR 0x0844) -> 0x000000000000a83c
[  259.419641401,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
[  259.419644124,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:0:␣
↪TFMR(2e12002870e04000) Timer Facility Error
[  259.419650678,7] HMI: Sending hmi job to thread 1
[  259.419652744,7] HMI: Sending hmi job to thread 2
[  259.419653051,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
[  259.419654725,7] HMI: Sending hmi job to thread 3
[  259.419654916,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
[  259.419658025,7] HMI: Received HMI interrupt: HMER = 0x0840000000000000
[  259.419658406,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:2:␣
↪TFMR(2e12002870e04000) Timer Facility Error
[  259.419663095,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:3:␣
↪TFMR(2e12002870e04000) Timer Facility Error
[  259.419655234,7] HMI: [Loc: U78D3.ND1.WZS004A-P1-C48]: P:8 C:17 T:1:␣
↪TFMR(2e12002870e04000) Timer Facility Error
[  259.425109779,7] OPAL: Start CPU 0x0845 (PIR 0x0845) -> 0x000000000000a83c
[  259.429870681,7] OPAL: Start CPU 0x0846 (PIR 0x0846) -> 0x000000000000a83c
[  259.434549250,7] OPAL: Start CPU 0x0847 (PIR 0x0847) -> 0x000000000000a83c
```

- core/cpu: Fix memory allocation for job array

  fixes: 7a3f307e core/cpu: parallelise global CPU register setting jobs

This bug would result in boot-hang on some configurations due to cpu_wait_job() endlessly waiting for the last bogus jobs[cpu->pir] pointer.

- i2c: Fix multiple-enqueue of the same request on NACK

  i2c_request_send() will retry the request if the error is a NAK, however it forgets to clear the "ud.done" flag. It will thus loop again and try to re-enqueue the same request causing internal request list corruption.

- i2c: Ensure ordering between i2c_request_send() and completion

  i2c_request_send loops waiting for a flag "uc.done" set by the completion routine, and then look for a result code also set by that same completion.

  There is no synchronization, the completion can happen on another processor, so we need to order the stores to uc and the reads from uc so that uc.done is stored last and tested first using memory barriers.

- pci: Clarify power down logic

  Currently pci_scan_bus() unconditionally calls pci_slot_set_power_state() when it's finished scanning a bus. This is one of those things that makes you go "WHAT?" when you first see it and frankly the skiboot PCI code could do with less of that.

Fast Reboot

- fast-reboot: parallel memory clearing

  Arbitrarily pick 16GB as the unit of parallelism, and split up clearing memory into jobs and schedule them node-local to the memory (or on node 0 if we can't work that out because it's the memory up to SKIBOOT_BASE)

  This seems to cut at least ~40% time from memory zeroing on fast-reboot on a 256GB Boston system.

  For many systems, scanning PCI takes about as much time as zeroing all of RAM, so we may as well do them at the same time and cut a few seconds off the total fast reboot time.

- fast-reboot: verify firmware "romem" checksum

  This takes a checksum of skiboot memory after boot that should be unchanged during OS operation, and verifies it before allowing a fast reboot.

  This is not read-only memory from skiboot's point of view, beause it includes things like the opal branch table that gets populated during boot.

  This helps to improve the integrity of firmware against host and runtime firmware memory scribble bugs.

- core/fast-reboot: print the fast reboot disable reason

  Once things start to go wrong, disable_fast_reboot can be called a number of times, so make the first reason sticky, and also print it to the console at disable time. This helps with making sense of fast reboot disables.

- Add fast-reboot property to /ibm,opal DT node

  this means that if it's permanently disabled on boot, the test suite can pick that up and not try a fast reboot test.

Utilities

- pflash: Add –skip option for reading

  Add a –skip=N option to pflash to skip N number of bytes when reading. This would allow users to print the VERSION partition without the STB header by specifying the –skip=4096 argument, and it's a more generic solution rather than making pflash depend on secure/trusted boot code.

- xscom-utils: Rework getsram

  Allow specifying a file on the command line to read OCC SRAM data into. If no file is specified then we print it to stdout as text. This is a bit inconsistent, but it retains compatibility with the existing tool.

- xscom-utils/getsram: Make it work on P9

  The XSCOM base address of the OCC control registers changed slightly between P8 and P9. Fix this up and add a bit of PVR checking so we look in the right place.

- opal-prd: Fix opal-prd crash

  Presently callback function from HBRT uses r11 to point to target function pointer. r12 is garbage. This works fine when we compile with "-no-pie" option (as we don't use r12 to calculate TOC).

  As per ABIv2 : "r12 : Function entry address at global entry point"

  With "-pie" compilation option, we have to set r12 to point to global function entry point. So that we can calculate TOC properly.

  Crash log without this patch:

  ```
  opal-prd[2864]: unhandled signal 11 at 0000000000029320 nip 00000 00102012830 lr␣
  ↪0000000102016890 code 1
  ```

Development and Debugging

- core/lock: Use try_lock_caller() in lock_caller() to capture owner

  Otherwise we can get reports of core/lock.c owning the lock, which is not helpful when tracking down ownership issues.

- core/flash: Emit a warning if Skiboot version doesn't match

  This means you'll get a warning that you've modified skiboot separately to the rest of the PNOR image, which can be useful in determining what firmware is actually running on a machine.

- gcov: link in ctors* as newer GCC doesn't group them all

  It seems that newer toolchains get us multiple ctors sections to link in rather than just one. If we discard them (as we were doing), then we don't have a working gcov build (and we get the "doesn't look sane" warning on boot).

- core/flash: Log return code when ffs_init() fails

  Knowing the return code is at least better than not knowing the return code.

- gcov: Fix building with GCC8

- travis/ci: rework Dockerfiles to produce build artifacts

  ubuntu-latest was also missing clang, as ubuntu-latest is closer to ubuntu 18.04 than 16.04

- cpu: add cpu_queue_job_on_node()

  Add a job scheduling API which will run the job on the requested chip_id (or return failure).

- opal-ci: Build old dtc version for fedora 28

  There are patches that will go into dtc to fix the issues we hit, but for the moment let's just build and use a slightly older version.

- mem_region: Merge similar allocations when dumping

  Currently we print one line for each allocation done at runtime when dumping the memory allocations. We do a few thousand allocations at boot so this can result in a huge amount of text being printed which is a) slow to print, and b) Can result in the log buffer overflowing which destroys otherwise useful information.

  This patch adds a de-duplication to this memory allocation dump by merging "similar" allocations (same location, same size) into one.

  Unfortunately, the algorithm used to do the de-duplication is quadratic, but considering we only dump the allocations in the event of a fatal error I think this is acceptable. I also did some benchmarking and found that on a ZZ it takes ~3ms to do a dump with 12k allocations. On a Zaius it's slightly longer at about ~10ms for 10k allocs. However, the difference there was due to the output being written to the UART.

  This patch also bumps the log level to PR_NOTICE. PR_INFO messages are suppressed at the default log level, which probably isn't something you want considering we only dump the allocations when we run out of skiboot heap space.

- core/lock: fix timeout warning causing a deadlock false positive

  If a lock waiter exceeds the warning timeout, it prints a message while still registered as requesting the lock. Printing the message can take locks, so if one is held when the owner of the original lock tries to print a message, it will get a false positive deadlock detection, which brings down the system.

  This can easily be hit when there is a lot of HMI activity from a KVM guest, where the timebase was not returned to host timebase before calling the HMI handler.

- hw/p8-i2c: Print the set error bits

  This is purely to save me from having to look it up every time someone gets an I2C error.

- init: Fix starting stripped kernel

  Currently if we try to run a raw/stripped binary kernel (ie. without the elf header) we crash with:

  ```
  [    0.008757768,5] INIT: Waiting for kernel...
  [    0.008762937,5] INIT: platform wait for kernel load failed
  [    0.008768171,5] INIT: Assuming kernel at 0x20000000
  [    0.008779241,3] INIT: ELF header not found. Assuming raw binary.
  [    0.017047348,5] INIT: Starting kernel at 0x0, fdt at 0x3044b230 14339 bytes
  [    0.017054251,0] FATAL: Kernel is zeros, can't execute!
  [    0.017059054,0] Assert fail: core/init.c:590:0
  [    0.017065371,0] Aborting!
  ```

  This is because we haven't set kernel_entry correctly in this path. This fixes it.

- cpu: Better output when waiting for a very long job

  Instead of printing at the end if the job took more than 1s, print in the loop every 30s along with a backtrace. This will give us some output if the job is deadlocked.

- lock: Fix interactions between lock dependency checker and stack checker

  The lock dependency checker does a few nasty things that can cause re-entrancy deadlocks in conjunction with the stack checker or in fact other debug tests.

  A lot of it revolves around taking a new lock (dl_lock) as part of the locking process.

  This tries to fix it by making sure we do not hit the stack checker while holding dl_lock.

  We achieve that in part by directly using the low-level __try_lock and manually unlocking on the dl_lock, and making some functions "nomcount".

  In addition, we mark the dl_lock as being in the console path to avoid deadlocks with the UART driver.

We move the enabling of the deadlock checker to a separate config option from DEBUG_LOCKS as well, in case we chose to disable it by default later on.

- xscom-utils/adu_scoms.py: run 2to3 over it
- clang: -Wno-error=ignored-attributes

Mambo Platform

- mambo: Merge PMEM_DISK and PMEM_VOLATILE code

  PMEM_VOLATILE and PMEM_DISK can't be used together and are basically copies of the same code.

  This merges the two and allows them used together. Same API is kept.
- hw/chiptod: test QUIRK_NO_CHIPTOD in opal_resync_timebase

  This allows some test coverage of deep stop states in Linux with Mambo.
- core/mem_region: mambo reserve kernel payload areas

  Mambo image payloads get overwritten by the OS and by fast reboot memory clearing because they have no region defined. Add them, which allows fast reboot to work.

Qemu platform

- nx: Don't abort on missing NX when using a QEMU machine

  These don't have an NX node (and probably never will) as they don't provide any coprocessor. However, the DARN instruction works so this abort is unnecessary.

POWER8 Platforms

- SBE-p8: Do all sbe timer update with xscom lock held

  Without this, on some P8 platforms, we could (falsely) think the SBE timer had stalled getting the dreaded "timer stuck" message.

  The code was doing the mftb() to set the start of the timeout period while *not* holding the lock, so the 1ms timeout started sometime when somebody else had the xscom lock.

  The simple solution is to just do the whole routine holding the xscom lock, so do it that way.

Vesnin Platform

- platforms/astbmc/vesnin: Send list of PCI devices to BMC through IPMI

  Implements sending a list of installed PCI devices through IPMI protocol. Each PCI device description is sent as a standalone IPMI message. A list of devices can be gathered from separate messages using the session identifier. The session Id is an incremental counter that is updated at the start of synchronization session.

POWER9 Platforms

- STOP API: API conditionally supports 255 SCOM restore entries for each quad.

- hdata/i2c: Skip unknown device type

  Do not add unknown I2C devices to device tree.

- hdata/i2c: Add whitelisting for Host I2C devices

  Many of the devices that we get information about through HDAT are for use by firmware rather than the host operating system. This patch adds a boolean flag to hdat_i2c_info structure that indicates whether devices with a given purpose should be reserved for use inside of OPAL (or some other firmware component, such as the OCC).

- hdata/iohub: Fix Cumulus Hub ID number
- opal/hmi: Wakeup the cpu before reading core_fir

  When stop state 5 is enabled, reading the core_fir during an HMI can result in a xscom read error with xscom_read() returning an OPAL_XSCOM_PARTIAL_GOOD error code and core_fir value of all FFs. At present this return error code is not handled in decode_core_fir() hence the invalid core_fir value is sent to the kernel where it interprets it as a FATAL hmi causing a system check-stop.

  This can be prevented by forcing the core to wake-up using before reading the core_fir. Hence this patch wraps the call to read_core_fir() within calls to dctl_set_special_wakeup() and dctl_clear_special_wakeup().

- xive: Disable block tracker

  Due to some HW errata, the block tracking facility (performance optimisation for large systems) should be disabled on Nimbus chips. Disable it unconditionally for now.

- opal/hmi: Ignore debug trigger inject core FIR.

  Core FIR[60] is a side effect of the work around for the CI Vector Load issue in DD2.1. Usually this gets delivered as HMI with HMER[17] where Linux already ignores it. But it looks like in some cases we may happen to see CORE_FIR[60] while we are already in Malfunction Alert HMI (HMER[0]) due to other reasons e.g. CAPI recovery or NPU xstop. If that happens then just ignore it instead of crashing kernel as not recoverable.

- hdata: Make sure reserved node name starts with "ibm, "

  HDAT does not provide consistent label format for reserved memory label. Few starts with "ibm," while few other starts with component name.

- hdata: Fix dtc warnings

  Fix dtc warnings related to mcbist node.

  ```
  Warning (reg_format): "reg" property in /xscom@623fc00000000/mcbist@1 has invalid␣
  ↪length (4 bytes) (#address-cells == 1, #size-cells == 1)
  Warning (reg_format): "reg" property in /xscom@623fc00000000/mcbist@2 has invalid␣
  ↪length (4 bytes) (#address-cells == 1, #size-cells == 1)
  Warning (reg_format): "reg" property in /xscom@603fc00000000/mcbist@1 has invalid␣
  ↪length (4 bytes) (#address-cells == 1, #size-cells == 1)
  Warning (reg_format): "reg" property in /xscom@603fc00000000/mcbist@2 has invalid␣
  ↪length (4 bytes) (#address-cells == 1, #size-cells == 1)
  ```

  Ideally we should add proper xscom range here… but we are not getting that information in HDAT today. Lets fix warning until we get proper data in HDAT.

PHB4

- phb4: Generate checkstop on AIB ECC corr/uncorr for DD2.0 parts

  On DD2.0 parts, PCIe ECC protection is not warranted in the response data path. Thus, for these parts, we need to flag any ECC errors detected from the adjacent AIB RX Data path so the part can be replaced.

This patch configures the FIRs so that we escalate these AIB ECC errors to a checkstop so the parts can be replaced.

- phb4: Reset pfir and nfir if new errors reported during ETU reset

  During fast-reboot new PEC errors can be latched even after ETU-Reset is asserted. This will result in values of variables nfir_cache and pfir_cache to be out of sync.

  During step-2 of CRESET nfir_cache and pfir_cache values are used to bring the PHB out of reset state. However if these variables are out as noted above of date the nfir/pfir registers are never reset completely and ETU still remains frozen.

  Hence this patch updates step-2 of phb4_creset to re-read the values of nfir/pfir registers to check if any new errors were reported after ETU-reset was asserted, report these new errors and reset the nfir/pfir registers. This should bring the ETU out of reset successfully.

- phb4: Disable nodal scoped DMA accesses when PB pump mode is enabled

  By default when a PCIe device issues a read request via the PHB it is first issued with nodal scope. When accessing GPU memory the NPU does not know at the time of response if the requested memory page is off node or not. Therefore every read of GPU memory by a PHB is retried with larger scope which introduces bandwidth and latency issues.

  On smaller boxes which have pump mode enabled nodal and group scoped reads are treated the same and both types of request are broadcast to one chip. Therefore we can avoid the retry by disabling nodal scope on the PHB for these boxes. On larger boxes nodal (single chip) and group (multiple chip) scoped reads are treated differently. Therefore we avoid disabling nodal scope on large boxes which have pump mode disabled to avoid all PHB requests being broadcast to multiple chips.

- phb4/capp: Only reset FIR bits that cause capp machine check

  During CAPP recovery do_capp_recovery_scoms() will reset the CAPP Fir register just after CAPP recovery is completed. This has an unintentional side effect of preventing PRD from analyzing and reporting this error. If PRD tries to read the CAPP FIR after opal has already reset it, then it logs a critical error complaining "No active error bits found".

  To prevent this from happening we update do_capp_recovery_scoms() to only reset fir bits that cause CAPP machine check (local xstop). This is done by reading the CAPP Fir Action0/1 & Mask registers and generating a mask which is then written on CAPP_FIR_CLEAR register.

- phb4: Check for RX errors after link training

  Some PHB4 PHYs can get stuck in a bad state where they are constantly retraining the link. This happens transparently to skiboot and Linux but will causes PCIe to be slow. Resetting the PHB4 clears the problem.

  We can detect this case by looking at the RX errors count where we check for link stability. This patch does this by modifying the link optimal code to check for RX errors. If errors are occurring we retrain the link irrespective of the chip rev or card.

  Normally when this problem occurs, the RX error count is maxed out at 255. When there is no problem, the count is 0. We chose 8 as the max rx errors value to give us some margin for a few errors. There is also a knob that can be used to set the error threshold for when we should retrain the link. ie

  ```
  nvram -p ibm,skiboot --update-config phb-rx-err-max=8
  ```

- hw/phb4: Add a helper to dump the PELT-V

  The "Partitionable Endpoint Lookup Table (Vector)" is used by the PHB when processing EEH events. The PELT-V defines which PEs should be additionally frozen in the event of an error being flagged on a given PE. Knowing the state of the PELT-V is sometimes useful for debugging PHB issues so this patch adds a helper to dump it.

- hw/phb4: Print the PEs in the EEH dump in hex

  Linux always displays the PE number in hexidecimal while skiboot displays the PEST index (PE number) in decimal. This makes correlating errors between Skiboot and Linux more annoying than it should be so this patch makes Skiboot print the PEST number in hex.

- phb4: Reallocate PEC2 DMA-Read engines to improve GPU-Direct bandwidth

  We reallocate additional 16/8 DMA-Read engines allocated to stack0/1 on PEC2 respectively. This is needed to improve bandwidth available to the Mellanox CX5 adapter when trying to read GPU memory (GPU-Direct).

  If kernel cxl driver indicates a request to allocate maximum possible DMA read engines when calling enable_capi_mode() and card is attached to PEC2/stack0 slot then we assume its a Mellanox CX5 adapter. We then allocate additional 16/8 extra DMA read engines to stack0 and stack1 respectively on PEC2. This is done by populating the XPEC_PCI_PRDSTKOVR and XPEC_NEST_READ_STACK_OVERRIDE as suggested by the h/w team.

- phb4: Enable PHB MMIO-0/1 Bars only when mmio window exists

  Presently phb4_probe_stack() will always enable PHB MMIO0/1 windows even if they doesn't exist in phy_map. Hence we do some minor shuffling in the phb4_probe_stack() so that MMIO-0/1 Bars are only enabled if there corresponding MMIO window exists in the phy_map. In case phy_map for an mmio window is '0' we set the corresponding BAR register to '0'.

- hw/phb4: Use local_alloc for phb4 structures

  Struct phb4 is fairly heavyweight at 283664 bytes. On systems with 6x PHBs per socket this results in using 3.2MB of heap space the PHB structures alone. This is a fairly large chunk of our 12MB heap and on systems with particularly large PCIe topologies, or additional PHBs we can fail to boot because we cannot allocate space for the FDT blob.

  This patch switches to using local_alloc() for the PHB structures so they don't consume too large a portion of our 12MB heap space.

- phb4: Fix typo in disable lane eq code

  In this commit

  ```
  commit 737c0ba3d72b8aab05a765a9fc111a48faac0f75
  Author: Michael Neuling <mikey@neuling.org>
  Date:   Thu Feb 22 10:52:18 2018 +1100
  phb4: Disable lane eq when retrying some nvidia GEN3 devices
  ```

  We made a typo and set PH2 twice. This fixes it.

  It worked previously as if only phase 2 (PH2) is set it, skips phase 2 and phase 3 (PH3).

- phb4: Don't probe a PHB if its garded

  Presently phb4_probe_stack() causes an exception while trying to probe a PHB if its garded. This causes skiboot to go into a reboot loop with following exception log:

  ```
   ***********************************************
   Fatal MCE at 000000003006ecd4   .probe_phb4+0x570
   CFAR : 00000000300b98a0
   <snip>
   Aborting!
  CPU 0018 Backtrace:
   S: 0000000031cc37e0 R: 000000003001a51c   ._abort+0x4c
   S: 0000000031cc3860 R: 0000000030028170   .exception_entry+0x180
   S: 0000000031cc3a40 R: 0000000000001f10 *
   S: 0000000031cc3c20 R: 000000003006ecb0   .probe_phb4+0x54c
  ```

(continues on next page)

(continued from previous page)

```
S: 0000000031cc3e30 R: 0000000030014ca4   .main_cpu_entry+0x5b0
S: 0000000031cc3f00 R: 0000000030002700   boot_entry+0x1b8
```

This is caused as phb4_probe_stack() will ignore all xscom read/write errors to enable PHB Bars and then tries to perform an mmio to read PHB Version registers that cause the fatal MCE.

We fix this by ignoring the PHB probe if the first xscom_write() to populate the PHB Bar register fails, which indicates that there is something wrong with the PHB.

- phb4: Workaround PHB errata with CFG write UR/CA errors

  If the PHB encounters a UR or CA status on a CFG write, it will incorrectly freeze the wrong PE. Instead of using the PE# specified in the CONFIG_ADDRESS register, it will use the PE# of whatever MMIO occurred last.

  Work around this disabling freeze on such errors

- phb4: Handle allocation errors in phb4_eeh_dump_regs()

  If the zalloc fails (and it can be a rather large allocation), we will overwite memory at 0 instead of failing.

- phb4: Don't try to access non-existent PEST entries

  In a POWER9 chip, some PHB4s have 256 PEs, some have 512.

  Currently, the diagnostics code retrieves 512 unconditionally, which is wrong and causes us to incorrectly report bogus values for the "high" PEs on the small PHBs.

  Use the actual number of implemented PEs instead

CAPI2

- phb4/capp: Use link width to allocate STQ engines to CAPP

  Update phb4_init_capp_regs() to allocates STQ Engines to CAPP/PEC2 based on link width instead of always assuming it to x8.

  Also re-factor the function slightly to evaluate the link-width only once and cache it so that it can also be used to allocate DMA read engines.

- phb4/capp: Update DMA read engines set in APC_FSM_READ_MASK based on link-width

  Commit 47c09cdfe7a3("phb4/capp: Calculate STQ/DMA read engines based on link-width for PEC") update the CAPP init sequence by calculating the needed STQ/DMA-read engines based on link width and populating it in XPEC_NEST_CAPP_CNTL register. This however needs to be synchronized with the value set in CAPP APC FSM Read Machine Mask Register.

  Hence this patch update phb4_init_capp_regs() to calculate the link width of the stack on PEC2 and populate the same values as previously populated in PEC CAPP_CNTL register.

- capp: Fix the capp recovery timeout comparison

  The current capp recovery timeout control loop in do_capp_recovery_scoms() uses a wrong comparison for return value of tb_compare(). This may cause do_capp_recovery_scoms() to report an timeout earlier than the 168ms stipulated time.

  The patch fixes this by updating the loop timeout control branch in do_capp_recovery_scoms() to use the correct enum tb_cmpval.

- phb4: Disable 32-bit MSI in capi mode

If a capi device does a DMA write targeting an address lower than 4GB, it does so through a 32-bit operation, per the PCI spec. In capi mode, the first TVE entry is configured in bypass mode, so the address is valid. But with any (bad) luck, the address could be 0xFFFFxxxx, thus looking like a 32-bit MSI.

We currently enable both 32-bit and 64-bit MSIs, so the PHB will interpret the DMA write as a MSI, which very likely results in an EEH (MSI with a bad payload size).

We can fix it by disabling 32-bit MSI when switching the PHB to capi mode. Capi devices are 64-bit.

NVLINK2

- npu2: Add support for relaxed-ordering mode

  Some device drivers support out of order access to GPU memory. This does not affect the CPU view of memory but it does affect the GPU view of memory. It should only be enabled if the GPU driver has requested it.

  Add OPAL APIs allowing the driver to query relaxed ordering state or request it to be set for a device. Current hardware only allows relaxed ordering to be enabled per PCIe root port. So the code here doesn't enable relaxed ordering until it has been explicitly requested for every device on the port.

- Add the other 7 ATSD registers to the device tree.
- npu2/hw-procedures: Don't open code NPU2_NTL_MISC_CFG2_BRICK_ENABLE

  Name this bit properly. There's a lot more cleanup like this to be done, but I'm catching this one now as part of some related changes.

- npu2/hw-procedures: Enable parity and credit overflow checks

  Enable these error checking features by setting the appropriate bits in our one-off initialization of each "NTL Misc Config 2" register.

  The exception is NDL RX parity checking, which should be disabled during the link training procedures.

- npu2: Use correct kill type for TCE invalidation

  kill_type is enum of OPAL_PCI_TCE_KILL_PAGES, OPAL_PCI_TCE_KILL_PE, OPAL_PCI_TCE_KILL_ALL and phb4_tce_kill() gets it right but npu2_tce_kill() uses OPAL_PCI_TCE_KILL which is an OPAL API token.

  This fixes an obvious mistype.

OpenCAPI

- Support OpenCAPI on Witherspoon platform
- npu2-opencapi: Enable presence detection on ZZ

  Presence detection for opencapi adapters was broken for ZZ planars v3 and below. All ZZ systems currently used in the lab have had their planar upgraded, so we can now remove the override we had to force presence and activate presence detection. Which should improve boot time.

  Considering the state of opal support on ZZ, this is really only for lab usage on BML. The opencapi enablement team has okay'd the change. In the unlikely case somebody tries opencapi on an old ZZ, the presence detection through i2c will show that no adapter is present and skiboot won't try to access or train the link.

- npu2-opencapi: Don't send commands to NPU when link is down

  Even if an opencapi link is down, we currently always try to issue a config read operation when probing for PCI devices, because of the default scan map used for an opencapi PHB. The config operation fails, as expected, but it can also raise a FIR bit and trigger an HMI.

For opencapi, there's no root device like for a "normal" PCI PHB, so there's no reason to do the config operation. To fix it, we keep the scan map blank by default, and only add a device once the link is trained.

- opal/hmi: Catch NPU2 HMIs for opencapi

  HMIs for NPU2 are filtered with the 'compatible' string of the PHB, so add opencapi to the mix.

- occ: Wait if OCC GPU presence status not immediately available

  It takes a few seconds for the OCC to set everything up in order to read GPU presence. At present, we try to kick off OCC initialisation as early as possible to maximise the time it has to read GPU presence.

  Unfortunately sometimes that's not enough, so add a loop in occ_get_gpu_presence() so that on the first time we try to get GPU presence we keep trying for up to 2 seconds. Experimentally this seems to be adequate.

- hw/npu2-hw-procedures: Enable RX auto recal on OpenCAPI links

  The RX_RC_ENABLE_AUTO_RECAL flag is required on OpenCAPI but not NVLink.

  Traditionally, Hostboot sets this value according to the machine type. However, now that Witherspoon supports both NVLink and OpenCAPI, it can't tell whether or not a link is OpenCAPI.

  So instead, set it in skiboot, where it will only be triggered after we've done device detection and found an OpenCAPI device.

- hw/npu2-opencapi: Fix setting of supported OpenCAPI templates

  In opal_npu_tl_set(), we made a typo that means the OPAL_NPU_TL_SET call may not clear the enable bits for templates that were previously enabled but are now disabled.

  Fix the typo so we clear NPU2_OTL_CONFIG1_TX_TEMP2_EN as well as TEMP{1,3}_EN.

Barreleye G2 and Zaius platforms

- zaius: Add a slot table
- zaius: Add slots for the Barreleye G2 HDD rack

  The Barreleye G2 is distinct from the Zaius in that it features a 24 Bay NVMe/SATA HDD rack. To provide meaningful slot names for each NVMe device we need to define a slot table for the NVMe capable HDD bays.

  Unfortunately this isn't straightforward because the PCIe path to the NVMe devices isn't fixed. The PCIe topology is something like: P9 -> HBA card -> 9797 switch -> 20x NVMe HDD slots

  The 9797 switch is partitioned into two (or four) virtual switches which allow multiple HBA cards to be used (e.g. one per socket). As a result the exact BDFN of the ports will vary depending on how the system is configured.

  That said, the virtual switch configuration of the 9797 does not change the device and function numbers of the switch downports. This means that we can define a single slot table that maps switch ports to the NVMe bay names.

  Unfortunately we still need to guess which bus to use this table on, so we assume that any switch downport we find with the PEX9797 VDID is part of the 9797 that supports the HDD rack.

FSP based platforms (firenze and ZZ)

- phb4/capp: Update the expected Eye-catcher for CAPP ucode lid

  Currently on a FSP based P9 system load_capp_code() expects CAPP ucode lid header to have eye-catcher magic of 'CAPPPSLL'. However skiboot currently supports CAPP ucode only lids that have a eye-catcher magic of 'CAPPLIDH'. This prevents skiboot from loading the ucode with this error message:

```
CAPP: ucode header invalid
```

We fix this issue by updating load_capp_ucode() to use the eye-catcher value of ‘CAPPLIDH’ instead of ‘CAPPPSLL’.

- FSP: Improve Reset/Reload log message

  Below message is confusing. Lets make it clear.

  FSP sends “R/R complete notification” whenever there is a dump. We use *flag* to identify whether its its R/R completion -OR- just new dump notification.

```
[  483.406351956,6] FSP: SP says Reset/Reload complete
[  483.406354278,5] DUMP: FipS dump available. ID = 0x1a00001f [size: 6367640␣
↪bytes]
[  483.406355968,7]   A Reset/Reload was NOT done
```

Witherspoon platform

- platforms/astbmc/witherspoon: Implement OpenCAPI support

  OpenCAPI on Witherspoon is slightly more involved than on Zaius and ZZ, due to the OpenCAPI links using the SXM2 connectors that are used for NVLink GPUs.

  This patch adds the regular OpenCAPI platform information, and also a Witherspoon-specific presence detection callback that uses the previously added OCC GPU presence detection to figure out the device types plugged into each SXM2 socket.

  The SXM2 connectors are capable of carrying 2 OpenCAPI links, and future OpenCAPI devices are expected to make use of this. However, we don’t yet support ganged links and the various implications that has for handling things like device reset, so for now, we only enable 1 brick per device.

5.1.129 skiboot-6.2-rc2

skiboot v6.2-rc2 was released on Thursday November 29th 2018. It is the second release candidate of skiboot 6.2, which will become the new stable release of skiboot following the 6.1 release, first released July 11th 2018.

Skiboot 6.2 will mark the basis for op-build v2.2.

skiboot v6.2-rc2 contains all bug fixes as of *skiboot-6.0.14*, and *skiboot-5.4.10* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

Over *skiboot-6.2-rc1*, we have the following changes:

- npu2-opencapi: Log extra information on link training failure
- npu2-opencapi: Detect if link trained in degraded mode
- platform/firenze: Fix branch-to-null crash

  When the bus alloc and free methods were removed we missed a case in the Firenze platform slot code that relied on the the bus-specific method to the bus pointer in the request structure. This results in a branch-to-null during boot and a crash. This patch fixes it by initialising it manually here.

- libflash: Don’t merge ECC-protected ranges

  Libflash currently merges contiguous ECC-protected ranges, but doesn’t check that the ECC bytes at the end of the first and start of the second range actually match sanely. More importantly, if blocklevel_read() is called with a position at the start of a partition that is contained somewhere within a region that has been merged it will update the position assuming ECC wasn't being accounted for. This results in the position being somewhere well after the actual start of the partition which is incorrect.

For now, remove the code merging ranges. This means more ranges must be held and checked however it prevents incorrectly reading ECC-correct regions like below:

```
[  174.334119453,7] FLASH: CAPP partition has ECC
[  174.437349574,3] ECC: uncorrectable error: ffffffffffffffff ff
[  174.437426306,3] FLASH: failed to read the first 0x1000 from CAPP partition,␣
↪rc 14
[  174.439919343,3] CAPP: Error loading ucode lid. index=201d1
```

- libflash: Restore blocklevel tests

  This fell out in f58be46 "libflash/test: Rewrite Makefile.check to improve scalability". Add it back in as test-blocklevel.

- Warn on long OPAL calls

  Measure entry/exit time for OPAL calls and warn appropriately if the calls take too long (>100ms gets us a DEBUG log, > 1000ms gets us a warning).

CI, testing, and utilities

- travis: Coverity fixed their SSL cert
- opal-ci: Use ubuntu:rolling for Ubuntu latest image
- ffspart: Add test for eraseblock size
- ffspart: Add toc test
- hdata/test: workaround dtc bugs

  In dtc v1.4.5 to at least v1.4.7 there have been a few bugs introduced that change the layout of what's produced in the dts. In order to be immune from them, we should use the (provided) dtdiff utility, but we also need to run the dts we're diffing against through a dtb cycle in order to ensure we get the same format as what the hdat_to_dt to dts conversion will.

  This fixes a bunch of unit test failures on the version of dtc shipped with recent Linux distros such as Fedora 29.

5.1.130 skiboot-6.2.1

skiboot 6.2.1 was released on Wednesday February 20th, 2019. It replaces *skiboot-6.2* as the current stable release in the 6.2.x series.

It is recommended that 6.2.1 be used instead of any previous 6.2.x version due to the bug fixes it contains.

Bug fixes included in this release are:

- libflash/ecc: Fix compilation warning with gcc9

  Fixes: https://github.com/open-power/skiboot/issues/218

- core/opal: Print PIR value in exit path, useful for debugging
- core/ipmi: Improve error message

- firmware-versions: Add test case for parsing VERSION

  If we hit a entry in VERSION that is larger than our buffer size, we skip over it gracefully rather than overwriting the stack. This is only a problem if VERSION isn't trusted, which as of 4b8cc05a94513816d43fb8bd6178896b430af08f it is verified as part of Secure Boot.

- core/cpu: HID update race

  If the per-core HID register is updated concurrently by multiple threads, updates can get lost. This has been observed during fast reboot where the HILE bit does not get cleared on all cores, which can cause machine check exception interrupts to crash.

  Fix this by only updating HID on thread0.

- cpufeatures: Always advertise POWER8NVL as DD2

  Despite the major version of PVR being 1 (0x004c0100) for POWER8NVL, these chips are functionally equalent to P8/P8E DD2 levels.

  This advertises POWER8NVL as DD2. As the result, skiboot adds ibm,powerpc-cpu-features/processor-control-facility for such CPUs and the linux kernel can use hypervisor doorbell messages to wake secondary threads; otherwise "KVM: CPU %d seems to be stuck" would appear because of missing LPCR_PECEDH.

- p9dsu: Fix p9dsu slot tables

  Set the attributes on the slot tables to account for builtin or pluggable etypes, this will allow pci enumeration to calculate subordinate buses.

  Update some slot label strings.

  Add WIO Slot5 which is standard on the ESS config.

- core/lock: Stop drop_my_locks() from always causing abort

  The loop in drop_my_locks() looks like this:

  ```
  while((l = list_pop(&this_cpu()->locks_held, struct lock, list)) != NULL) {
          if (warn)
                  prlog(PR_ERR, "  %s\n", l->owner);
          unlock(l);
  }
  ```

  Both list_pop() and unlock() call list_del(). This means that on the last iteration of the loop, the list will be empty when we get to unlock_check(), causing this:

  ```
  LOCK ERROR: Releasing lock we don't hold depth @0x30493d20 (state:␣
  ↪0x0000000000000001)
  [13836.000173140,0] Aborting!
  CPU 0000 Backtrace:
   S: 0000000031c03930 R: 000000003001d840   ._abort+0x60
   S: 0000000031c039c0 R: 000000003001a0c4   .lock_error+0x64
   S: 0000000031c03a50 R: 0000000030019c70   .unlock+0x54
   S: 0000000031c03af0 R: 000000003001a040   .drop_my_locks+0xf4
  ```

  To fix this, change list_pop() to list_top().

- p9dsu: Fix p9dsu default variant

  Add the default when no riser_id is returned from the ipmi query.

  Allow a little more time for BMC reply and cleanup some label strings.

5.1.131 skiboot-6.2.2

skiboot 6.2.2 was released on Wednesday March 6th, 2019. It replaces *skiboot-6.2.1* as the current stable release in the 6.2.x series.

It is recommended that 6.2.2 be used instead of any previous 6.2.x version due to the bug fixes it contains.

Over *skiboot-6.2.1* we have several bug fixes, including important ones for powercap, ipmi-hiomap, astbmc and BMC communication driver.

powercap

- powercap: occ: Fix the powercapping range allowed for user

  OCC provides two limits for minimum powercap. One being hard powercap minimum which is guaranteed by OCC and the other one is a soft powercap minimum which is lesser than hard-min and may or may not be asserted due to various power-thermal reasons. So to allow the users to access the entire powercap range, this patch exports soft powercap minimum as the "powercap-min" DT property. And it also adds a new DT property called "powercap-hard-min" to export the hard-min powercap limit.

ASTBMC

- astbmc: Enable IPMI HIOMAP for AMI platforms

  Required for Habanero, Palmetto and Romulus.

- astbmc: Try IPMI HIOMAP for P8 (again)

  The HIOMAP protocol was developed after the release of P8 in preparation for P9. As a consequence P9 always uses it, but it has rarely been enabled for P8. P8DTU has recently added IPMI HIOMAP support to its BMC firmware, so enable its use in skiboot with P8 machines. Doing so requires some rework to ensure fallback works correctly as in the past the fallback was to mbox, which will only work for P9.

  Tested on Garrison, Palmetto without HIOMAP, Palmetto with HIOMAP, and Witherspoon.

- ast-io: Rework ast_sio_is_enabled() test sequence

  The postcondition of probing with a lock sequence is easier to make correct than with unlock. The original implementation left SuperIO locked after execution which broke an assumption of some callers.

  Tested on Garrison, Palmetto without HIOMAP, Palmetto with HIOMAP and Witherspoon.

P8DTU

- p8dtu: Enable HIOMAP support
- p8dtu: Configure BMC graphics

  We can no-longer read the values from the BMC in the way we have in the past. Values were provided by Eric Chen of SMC.

IPMI-HIOMAP

- ipmi-hiomap test case enhancements/fixes.

- libflash/ipmi-hiomap: Enforce message size for empty response

  The protocol defines the response to the associated messages as empty except for the command ID and sequence fields. If the BMC is returning extra data consider the message malformed.

- libflash/ipmi-hiomap: Remove unused close handling

  Issuing a HIOMAP_C_CLOSE is not required by the protocol specification, rather a close can be implicit in a subsequent CREATE_{READ,WRITE}_WINDOW request. The implicit close provides an opportunity to reduce LPC traffic and the implementation takes up that optimisation, so remove the case from the IPMI callback handler.

- libflash/ipmi-hiomap: Overhaul event handling

  Reworking the event handling was inspired by a bug report by Vasant where the host would get wedged on multiple flash access attempts in the face of a persistent error state on the BMC-side. The cause of this bug was the early-exit based on ctx->update, which erronously assumed that all events had been completely handled in prior calls to ipmi_hiomap_handle_events(). This is not true if e.g. HIOMAP_E_DAEMON_READY is clear in the prior calls.

  Regardless, there were other correctness and efficiency problems with the handling strategy:

  - Ack-able event state was not restored in the face of errors in the process of re-establishing protocol state
  - It forced needless window restoration with respect to the context in which ipmi_hiomap_handle_events() was called.
  - Tests for HIOMAP_E_DAEMON_READY and HIOMAP_E_FLASH_LOST were redundant with the overhauled error handling introduced in the previous patch

  Fix all of the above issues and add comments to explain the event handling flow.

  Tests for correctness follow later in the series.

- libflash/ipmi-hiomap: Overhaul error handling

  The aim is to improve the robustness with respect to absence of the BMC-side daemon. The current error handling roughly mirrors what was done for the mailbox implementation, but there's room for improvement.

  Errors are split into two classes, those that affect the transport state and those that affect the window validity. From here, we push the transport state error checks right to the bottom of the stack, to ensure the link is known to be in a good state before any message is sent. Window validity tests remain as they were in the hiomap_window_move() and ipmi_hiomap_read() functions. Validity tests are not necessary in the write and erase paths as we will receive an error response from the BMC when performing a dirty or flush on an invalid window.

  Recovery also remains as it was, done on entry to the blocklevel callbacks. If an error state is encountered in the middle of an operation no attempt is made to recover it on the spot, instead the error is returned up the stack and the caller can choose how it wishes to respond.

- libflash/ipmi-hiomap: Fix leak of msg in callback

BMC communication

- core/ipmi: Add ipmi sync messages to top of the list

  In ipmi_queue_msg_sync() path OPAL will wait until it gets response from BMC. If we do not get response ontime we may endup in kernel hardlockups. Hence lets add sync messages to top of the queue. This will reduces the chance of hardlockups.

- hw/bt: Introduce separate list for synchronous messages

BT send logic always sends top of bt message list to BMC. Once BMC reads the message, it clears the interrupt and bt_idle() becomes true.

bt_add_ipmi_msg_head() adds message to top of the list. If bt message list is not empty then:

– if bt_idle() is true then we will endup sending message to BMC before getting response from BMC for inflight message. Looks like on some BMC implementation this results in message timeout.

– else we endup starting message timer without actually sending message to BMC.. which is not correct.

This patch introduces separate list to track synchronous messages. bt_add_ipmi_msg_head() will add messages to tail of this new list. We will always process this queue before processing normal queue.

Finally this patch introduces new variable (inflight_bt_msg) to track inflight message. This will point to current inflight message.

- hw/bt: Fix message retry handler

  In some corner cases (like BMC reboot), bt_send_and_unlock() starts message timer, but won't send message to BMC as driver is not free to send message. bt_expire_old_msg() function enables H2B interrupt without actually sending message.

  This patch fixes above issue.

- ipmi/power: Fix system reboot issue

  Kernel makes reboot/shudown OPAL call for reboot/shutdown. Once kernel gets response from OPAL it runs opal_poll_events() until firmware handles the request.

  On BMC based system, OPAL makes IPMI call (IPMI_CHASSIS_CONTROL) to initiate system reboot/shutdown. At present OPAL queues IPMI messages and return SUCESS to Host. If BMC is not ready to accept command (like BMC reboot), then these message will fail. We have to manually reboot/shutdown the system using BMC interface.

  This patch adds logic to validate message return value. If message failed, then it will resend the message. At some stage BMC will be ready to accept message and handles IPMI message.

- hw/bt: Add backend interface to disable ipmi message retry option

  During boot OPAL makes IPMI_GET_BT_CAPS call to BMC to get BT interface capabilities which includes IPMI message max resend count, message timeout, etc,. Most of the time OPAL gets response from BMC within specified timeout. In some corner cases (like mboxd daemon reset in BMC, BMC reboot, etc) OPAL may not get response within timeout period. In such scenarios, OPAL resends message until max resend count reaches.

  OPAL uses synchronous IPMI message (ipmi_queue_msg_sync()) for few operations like flash read, write, etc. Thread will wait in OPAL until it gets response from BMC. In some corner cases like BMC reboot, thread may wait in OPAL for long time (more than 20 seconds) and results in kernel hardlockup.

  This patch introduces new interface to disable message resend option. We will disable message resend option for synchrous message. This will greatly reduces kernel hardlock up issues.

  This is short term fix. Long term solution is to convert all synchronous messages to asynhrounous one.

- qemu: bt device isn't always hanging off /

  Just use the normal for_each_compatible instead.

  Otherwise in the qemu model as executed by op-test, we wouldn't go down the astbmc_init() path, thus not having flash.

PHB3

- hw/phb3/naples: Disable D-states

  Putting "Mellanox Technologies MT27700 Family [ConnectX-4] [15b3:1013]" (more precisely, the second of 2 its PCI functions, no matter in what order) into the D3 state causes EEH with the "PCT timeout" error. This has been noticed on garrison machines only and firestones do not seem to have this issue.

  This disables D-states changing for devices on root buses on Naples by installing a config space access filter (copied from PHB4).

5.1.132 skiboot-6.2.3

skiboot 6.2.3 was released on Tuesday March 19th, 2019. It replaces *skiboot-6.2.2* as the current stable release in the 6.2.x series.

It is recommended that 6.2.3 be used instead of any previous 6.2.x version due to the bug fixes it contains.

Bug fixes included in this release are:

- p9dsu: Undo slot label name changes

  During some code updates the slot labels were updated to reflect the phb layout, however expectations were that the slot labels be aligned with the riser card slots and not the system planar slots.

  [stewart: The tale of how we got here is long and varied and not at all clear. The first ESS systems went out with a skiboot v5.9.8 with additional SuperMicro patches. It was probably a slot table, but who knows, we don't have the code so can't check. It's possible it was all coming in through HDAT instead). The op-build tree (thus the exact patches) shipped on systems that work correct seems to not be around anywhere anymore (if it ever was). It was only in skiboot v6.0 that a slot table made it in, and, of course, only having remote machines in random configs, including possibly with riser cards from Briggs&Stratton rather than the ones destined for this system, doesn't make for verifying this at all. It also doesn't help that *consistently* there is *never* any review on slot tables, and we've had things be wrong in the past. Combine this with not upstream Hostboot patches.]

- p9dsu: Fix slot labels for p9dsu2u

  Update the slot labels for the p9dsu2u tables.

- fast-reboot: occ: Call occ_pstates_init() on fast-reset on all machines

  Commit 815417dcda2e ("init, occ: Initialise OCC earlier on BMC systems") conditionally invoked occ_pstates_init() only on FSP based systems in load_and_boot_kernel(). Due to this pstate table is re-parsed on FSP system and skipped on BMC system during fast-reboot. So this patch fixes this by invoking occ_pstates_init() on all boxes during fast-reboot.

5.1.133 skiboot-6.2.4

skiboot 6.2.4 was released on Thursday May 9th, 2019. It replaces *skiboot-6.2.3* as the current stable release in the 6.2.x series.

It is recommended that 6.2.4 be used instead of any previous 6.2.x version due to the bug fixes it contains.

Bug fixes included in this release are:

- core/flash: Retry requests as necessary in flash_load_resource()

  We would like to successfully boot if we have a dependency on the BMC for flash even if the BMC is not current ready to service flash requests. On the assumption that it will become ready, retry for several minutes to cover a BMC reboot cycle and *eventually* rather than *immediately* crash out with:

```
[  269.549748] reboot: Restarting system
[  390.297462587,5] OPAL: Reboot request...
[  390.297737995,5] RESET: Initiating fast reboot 1...
[  391.074707590,5] Clearing unused memory:
[  391.075198880,5] PCI: Clearing all devices...
[  391.075201618,7] Clearing region 201ffe000000-201fff800000
[  391.086235699,5] PCI: Resetting PHBs and training links...
[  391.254089525,3] FFS: Error 17 reading flash header
[  391.254159668,3] FLASH: Can't open ffs handle: 17
[  392.307245135,5] PCI: Probing slots...
[  392.363723191,5] PCI Summary:
...
[  393.423255262,5] OCC: All Chip Rdy after 0 ms
[  393.453092828,5] INIT: Starting kernel at 0x20000000, fdt at
0x30800a88 390645 bytes
[  393.453202605,0] FATAL: Kernel is zeros, can't execute!
[  393.453247064,0] Assert fail: core/init.c:593:0
[  393.453289682,0] Aborting!
CPU 0040 Backtrace:
 S: 0000000031e03ca0 R: 000000003001af60   ._abort+0x4c
 S: 0000000031e03d20 R: 000000003001afdc   .assert_fail+0x34
 S: 0000000031e03da0 R: 00000000300146d8   .load_and_boot_kernel+0xb30
 S: 0000000031e03e70 R: 0000000030026cf0   .fast_reboot_entry+0x39c
 S: 0000000031e03f00 R: 0000000030002a4c   fast_reset_entry+0x2c
 --- OPAL boot ---
```

The OPAL flash API hooks directly into the blocklevel layer, so there's no delay for e.g. the host kernel, just for asynchronously loaded resources during boot.

- pci/iov: Remove skiboot VF tracking

  This feature was added a few years ago in response to a request to make the MaxPayloadSize (MPS) field of a Virtual Function match the MPS of the Physical Function that hosts it.

  The SR-IOV specification states the the MPS field of the VF is "ResvP". This indicates the VF will use whatever MPS is configured on the PF and that the field should be treated as a reserved field in the config space of the VF. In other words, a SR-IOV spec compliant VF should always return zero in the MPS field. Adding hacks in OPAL to make it non-zero is… misguided at best.

  Additionally, there is a bug in the way pci_device structures are handled by VFs that results in a crash on fast-reboot that occurs if VFs are enabled and then disabled prior to rebooting. This patch fixes the bug by removing the code entirely. This patch has no impact on SR-IOV support on the host operating system.

- astbmc: Handle failure to initialise raw flash

  Initialising raw flash lead to a dead assignment to rc. Check the return code and take the failure path as necessary. Both before and after the fix we see output along the lines of the following when flash_init() fails:

```
[   53.283182881,7] IRQ: Registering 0800..0ff7 ops @0x300d4b98 (data 0x3052b9d8)
[   53.283184335,7] IRQ: Registering 0ff8..0fff ops @0x300d4bc8 (data 0x3052b9d8)
[   53.283185513,7] PHB#0000: Initializing PHB...
[   53.288260827,4] FLASH: Can't load resource id:0. No system flash found
[   53.288354442,4] FLASH: Can't load resource id:1. No system flash found
[   53.342933439,3] CAPP: Error loading ucode lid. index=200ea
[   53.462749486,2] NVRAM: Failed to load
[   53.462819095,2] NVRAM: Failed to load
[   53.462894236,2] NVRAM: Failed to load
[   53.462967071,2] NVRAM: Failed to load
```

(continues on next page)

(continued from previous page)

```
[   53.463033077,2] NVRAM: Failed to load
[   53.463144847,2] NVRAM: Failed to load
```

Eventually followed by:

```
[   57.216942479,5] INIT: platform wait for kernel load failed
[   57.217051132,5] INIT: Assuming kernel at 0x20000000
[   57.217127508,3] INIT: ELF header not found. Assuming raw binary.
[   57.217249886,2] NVRAM: Failed to load
[   57.221294487,0] FATAL: Kernel is zeros, can't execute!
[   57.221397429,0] Assert fail: core/init.c:615:0
[   57.221471414,0] Aborting!
CPU 0028 Backtrace:
 S: 0000000031d43c60 R: 000000003001b274   ._abort+0x4c
 S: 0000000031d43ce0 R: 000000003001b2f0   .assert_fail+0x34
 S: 0000000031d43d60 R: 0000000030014814   .load_and_boot_kernel+0xae4
 S: 0000000031d43e30 R: 0000000030015164   .main_cpu_entry+0x680
 S: 0000000031d43f00 R: 0000000030002718   boot_entry+0x1c0
 --- OPAL boot ---
```

Analysis of the execution paths suggests we'll always "safely" end this way due the setup sequence for the blocklevel callbacks in flash_init() and error handling in blocklevel_get_info(), and there's no current risk of executing from unexpected memory locations. As such the issue is reduced to down to a fix for poor error hygene in the original change and a resolution for a Coverity warning (famous last words etc).

- hw/xscom: Enable sw xstop by default on p9

  This was disabled at some point during bringup to make life easier for the lab folks trying to debug NVLink issues. This hack really should have never made it out into the wild though, so we now have the following situation occuring in the field:

  1) A bad happens
  2) The host kernel recieves an unrecoverable HMI and calls into OPAL to request a platform reboot.
  3) OPAL rejects the reboot attempt and returns to the kernel with OPAL_PARAMETER.
  4) Kernel panics and attempts to kexec into a kdump kernel.

  A side effect of the HMI seems to be CPUs becoming stuck which results in the initialisation of the kdump kernel taking a extremely long time (6+ hours). It's also been observed that after performing a dump the kdump kernel then crashes itself because OPAL has ended up in a bad state as a side effect of the HMI.

  All up, it's not very good so re-enable the software checkstop by default. If people still want to turn it off they can using the nvram override.

- opal/hmi: Initialize the hmi event with old value of TFMR.

  Do this before we fix TFAC errors. Otherwise the event at host console shows no thread error reported in TFMR register.

  Without this patch the console event show TFMR with no thread error: (DEC parity error TFMR[59] injection)

```
[   53.737572] Severe Hypervisor Maintenance interrupt [Recovered]
[   53.737596]  Error detail: Timer facility experienced an error
[   53.737611]  HMER: 0840000000000000
[   53.737621]  TFMR: 3212000870e04000
```

  After this patch it shows old TFMR value on host console:

```
[ 2302.267271] Severe Hypervisor Maintenance interrupt [Recovered]
[ 2302.267305]  Error detail: Timer facility experienced an error
[ 2302.267320]  HMER: 0840000000000000
[ 2302.267330]  TFMR: 3212000870e14010
```

- libflash/ipmi-hiomap: Fix blocks count issue

  We convert data size to block count and pass block count to BMC. If data size is not block aligned then we endup sending block count less than actual data. BMC will write partial data to flash memory.

  Sample log

```
[  594.388458416,7] HIOMAP: Marked flash dirty at 0x42010 for 8
[  594.398756487,7] HIOMAP: Flushed writes
[  594.409596439,7] HIOMAP: Marked flash dirty at 0x42018 for 3970
[  594.419897507,7] HIOMAP: Flushed writes
```

  In this case HIOMAP sent data with block count=0 and hence BMC didn't flush data to flash.

  Lets fix this issue by adjusting block count before sending it to BMC.

- Fix hang in pnv_platform_error_reboot path due to TOD failure.

  On TOD failure, with TB stuck, when linux heads down to pnv_platform_error_reboot() path due to unrecoverable hmi event, the panic cpu gets stuck in OPAL inside ipmi_queue_msg_sync(). At this time, rest all other cpus are in smp_handle_nmi_ipi() waiting for panic cpu to proceed. But with panic cpu stuck inside OPAL, linux never recovers/reboot.

```
p0 c1 t0
NIA : 0x000000003001dd3c <.time_wait+0x64>
CFAR : 0x000000003001dce4 <.time_wait+0xc>
MSR : 0x9000000002803002
LR : 0x000000003002ecf8 <.ipmi_queue_msg_sync+0xec>

STACK: SP NIA
0x0000000031c236e0 0x0000000031c23760 (big-endian)
0x0000000031c23760 0x000000003002ecf8 <.ipmi_queue_msg_sync+0xec>
0x0000000031c237f0 0x00000000300aa5f8 <.hiomap_queue_msg_sync+0x7c>
0x0000000031c23880 0x00000000300aaadc <.hiomap_window_move+0x150>
0x0000000031c23950 0x00000000300ab1d8 <.ipmi_hiomap_write+0xcc>
0x0000000031c23a90 0x00000000300a7b18 <.blocklevel_raw_write+0xbc>
0x0000000031c23b30 0x00000000300a7c34 <.blocklevel_write+0xfc>
0x0000000031c23bf0 0x0000000030030be0 <.flash_nvram_write+0xd4>
0x0000000031c23c90 0x000000003002c128 <.opal_write_nvram+0xd0>
0x0000000031c23d20 0x00000000300051e4 <opal_entry+0x134>
0xc000001fea6e7870 0xc0000000000a9060 <opal_nvram_write+0x80>
0xc000001fea6e78c0 0xc000000000030b84 <nvram_write_os_partition+0x94>
0xc000001fea6e7960 0xc0000000000310b0 <nvram_pstore_write+0xb0>
0xc000001fea6e7990 0xc0000000004792d4 <pstore_dump+0x1d4>
0xc000001fea6e7ad0 0xc00000000018a570 <kmsg_dump+0x140>
0xc000001fea6e7b40 0xc000000000028e5c <panic_flush_kmsg_end+0x2c>
0xc000001fea6e7b60 0xc0000000000a7168 <pnv_platform_error_reboot+0x68>
0xc000001fea6e7bd0 0xc0000000000ac9b8 <hmi_event_handler+0x1d8>
0xc000001fea6e7c80 0xc00000000012d6c8 <process_one_work+0x1b8>
0xc000001fea6e7d20 0xc00000000012da28 <worker_thread+0x88>
0xc000001fea6e7db0 0xc0000000001366f4 <kthread+0x164>
0xc000001fea6e7e20 0xc00000000000b65c <ret_from_kernel_thread+0x5c>
```

  This is because, there is a while loop towards the end of ipmi_queue_msg_sync() which keeps looping until "sync_msg" does not match with "msg". It loops over time_wait_ms() until exit condition is met. In normal scenario time_wait_ms() calls run pollers so that ipmi backend gets a chance to check ipmi response and set sync_msg to NULL.

```
while (sync_msg == msg)
        time_wait_ms(10);
```

But in the event when TB is in failed state time_wait_ms()->time_wait_poll() returns immediately without calling pollers and hence we end up looping forever. This patch fixes this hang by calling opal_run_pollers() in TB failed state as well.

- core/ipmi: Print correct netfn value
- libffs: Fix string truncation gcc warning.

  Use memcpy as other libffs functions do.

5.1.134 skiboot-6.3

skiboot v6.3 was released on Friday May 3rd 2019. It is the first release of skiboot 6.3, which becomes the new stable release of skiboot following the 6.2 release, first released December 14th 2018.

Skiboot 6.3 will mark the basis for op-build v2.3.

skiboot v6.3 contains all bug fixes as of *skiboot-6.0.20*, and *skiboot-6.2.3* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

Over skiboot 6.2, we have the following changes:

New Features

- hw/imc: Enable opal calls to init/start/stop IMC Trace mode

  New OPAL APIs for In-Memory Collection Counter infrastructure(IMC), including a new device type called OPAL_IMC_COUNTERS_TRACE.

- xive: Add calls to save/restore the queues and VPs HW state

  To be able to support migration of guests using the XIVE native exploitation mode, (where the queue is effectively owned by the guest), KVM needs to be able to save and restore the HW-modified fields of the queue, such as the current queue producer pointer and generation bit, and to retrieve the modified thread context registers of the VP from the NVT structure : the VP interrupt pending bits.

  However, there is no need to set back the NVT structure on P9. P10 should be the same.

- witherspoon: Add nvlink2 interconnect information

  GPUs on Redbud and Sequoia platforms are interconnected in groups of 2 or 3 GPUs. The problem with that is if the user decides to pass a single GPU from a group to the userspace, we need to ensure that links between GPUs do not get enabled.

  A V100 GPU provides a way to disable selected links. In order to only disable links to peer GPUs, we need a topology map.

  This adds an "ibm,nvlink-peers" property to a GPU DT node with phandles of peer GPUs and NVLink2 bridges. The index in the property is a GPU link number.

- platforms/romulus: Also support talos

  The two are similar enough and I'd like to have a slot table for our Talos.

- OpenCAPI support! (see *OpenCAPI* section)
- opal/hmi: set a flag to inform OS that TOD/TB has failed.

  Set a flag to indicate OS about TOD/TB failure as part of new opal_handle_hmi2 handler. This flag then can be used by OS to make sure functions depending on TB value (e.g. udelay()) are aware of TB not ticking.
- astbmc: Enable IPMI HIOMAP for AMI platforms

  Required for Habanero, Palmetto and Romulus.
- power-mgmt : occ : Add 'freq-domain-mask' DT property

  Add a new device-tree property freq-domain-indicator to define group of CPUs which would share same frequency. This property has been added under power-mgmt node. It is a bitmask.

  Bitwise AND is taken between this bitmask value and PIR of cpu. All the CPUs lying in the same frequency domain will have same result for AND.

  For example, For POWER9, 0xFFF0 indicates quad wide frequency domain. Taking AND with the PIR of CPUs will yield us frequency domain which is quad wise distribution as last 4 bits have been masked which represent the cores.

  Similarly, 0xFFF8 will represent core wide frequency domain for P8.

  Also, Add a new device-tree property domain-runs-at which will denote the strategy OCC is using to change the frequency of a frequency-domain. There can be two strategy - FREQ_MOST_RECENTLY_SET and FREQ_MAX_IN_DOMAIN.

  FREQ_MOST_RECENTLY_SET : the OCC sets the frequency of the quad to the most recent frequency value requested by the CPUs in the quad.

  FREQ_MAX_IN_DOMAIN : the OCC sets the frequency of the CPUs in the Quad to the maximum of the latest frequency requested by each of the component cores.
- powercap: occ: Fix the powercapping range allowed for user

  OCC provides two limits for minimum powercap. One being hard powercap minimum which is guaranteed by OCC and the other one is a soft powercap minimum which is lesser than hard-min and may or may not be asserted due to various power-thermal reasons. So to allow the users to access the entire powercap range, this patch exports soft powercap minimum as the "powercap-min" DT property. And it also adds a new DT property called "powercap-hard-min" to export the hard-min powercap limit.
- Add NVDIMM support

  NVDIMMs are memory modules that use a battery backup system to allow the contents RAM to be saved to non-volatile storage if system power goes away unexpectedly. This allows them to be used a high-performance storage device, suitable for serving as a cache for SSDs and the like.

  Configuration of NVDIMMs is handled by hostboot and communicated to OPAL via the HDAT. We need to parse out the NVDIMM memory ranges and create memory regions with the "pmem-region" compatible label to make them available to the host.
- core/exceptions: implement support for MCE interrupts in powersave

  The ISA specifies that MCE interrupts in power saving modes will enter at 0x200 with powersave bits in SRR1 set. This is not currently supported properly, the MCE will just happen like a normal interrupt, but GPRs could be lost, which would lead to crashes (e.g., r1, r2, r13 etc).

  So check the power save bits similarly to the sreset vector, and handle this properly.
- core/exceptions: allow recoverable sreset exceptions

  This requires implementing the MSR[RI] bit. Then just allow all non-fatal sreset exceptions to recover.

- core/exceptions: implement an exception handler for non-powersave sresets

  Detect non-powersave sresets and send them to the normal exception handler which prints registers and stack.

- Add PVR_TYPE_P9P

  Enable a new PVR to get us running on another p9 variant.

Since v6.3-rc2:

- Expose PNOR Flash partitions to host MTD driver via devicetree

  This makes it possible for the host to directly address each partition without requiring each application to directly parse the FFS headers. This has been in use for some time already to allow BOOTKERNFW partition updates from the host.

  All partitions except BOOTKERNFW are marked readonly.

  The BOOTKERNFW partition is currently exclusively used by the TalosII platform

- Write boot progress to LPC port 80h

  This is an adaptation of what we currently do for op_display() on FSP machines, inventing an encoding for what we can write into the single byte at LPC port 80h.

  Port 80h is often used on x86 systems to indicate boot progress/status and dates back a decent amount of time. Since a byte isn't exactly very expressive for everything that can go on (and wrong) during boot, it's all about compromise.

  Some systems (such as Zaius/Barreleye G2) have a physical dual 7 segment display that display these codes. So far, this has only been driven by hostboot (see hostboot commit 90ec2e65314c).

- Write boot progress to LPC ports 81 and 82

  There's a thought to write more extensive boot progress codes to LPC ports 81 and 82 to supplement/replace any reliance on port 80.

  We want to still emit port 80 for platforms like Zaius and Barreleye that have the physical display. Ports 81 and 82 can be monitored by a BMC though.

- Add Talos II platform

  Talos II has some hardware differences from Romulus, therefore we cannot guarantee Talos II == Romulus in skiboot. Copy and slightly modify the Romulus files for Talos II.

Since v6.3-rc1:

- cpufeatures: Add tm-suspend-hypervisor-assist and tm-suspend-xer-so-bug node

  tm-suspend-hypervisor-assist for P9 >=DD2.2 And a tm-suspend-xer-so-bug node for P9 DD2.2 only.

  I also treat P9P as P9 DD2.3 and add a unit test for the cpufeatures infrastructure.

  Fixes: https://github.com/open-power/skiboot/issues/233

Deprecated/Removed Features

- opal: Deprecate reading the PHB status

  The OPAL_PCI_EEH_FREEZE_STATUS call takes a bunch of parameters, one of them is @phb_status. It is defined as __be64* and always NULL in the current Linux upstream but if anyone ever decides to read that status, then the PHB3's handler will assume it is struct OpalIoPhb3ErrorData* (which is a lot bigger than 8 bytes) and zero it causing the stack corruption; p7ioc-phb has the same issue.

This removes @phb_status from all eeh_freeze_status() hooks and moves the error message from PHB4 to the affected OPAL handlers.

As far as we can tell, nobody has ever used this and thus it's safe to remove.

- Remove POWER9N DD1 support

  This is not a shipping product and is no longer supported by Linux or other firmware components.

Since v6.3-rc3:

- Disable fast-reset for POWER8

  There is a bug with fast-reset when CPU cores are busy, which can be reproduced by running *stress* and then trying *reboot -ff* (this is what the op-test test cases FastRebootHostStress and FastRebootHostStressTorture do). What happens is the cores lock up, which isn't the best thing in the world when you want them to start executing instructions again.

  A workaround is to use instruction ramming, which while greatly increasing the reliability of fast-reset on p8, doesn't make it perfect.

  Instruction ramming is what pdbg was modified to do in order to have the sreset functionality work reliably on p8. pdbg patches: https://patchwork.ozlabs.org/project/pdbg/list/?series=96593&state=*

  Fixes: https://github.com/open-power/skiboot/issues/185

General

- core/i2c: Various bits of refactoring
- refactor backtrace generation infrastructure
- astbmc: Handle failure to initialise raw flash

  Initialising raw flash lead to a dead assignment to rc. Check the return code and take the failure path as necessary. Both before and after the fix we see output along the lines of the following when flash_init() fails:

```
[   53.283182881,7] IRQ: Registering 0800..0ff7 ops @0x300d4b98 (data 0x3052b9d8)
[   53.283184335,7] IRQ: Registering 0ff8..0fff ops @0x300d4bc8 (data 0x3052b9d8)
[   53.283185513,7] PHB#0000: Initializing PHB...
[   53.288260827,4] FLASH: Can't load resource id:0. No system flash found
[   53.288354442,4] FLASH: Can't load resource id:1. No system flash found
[   53.342933439,3] CAPP: Error loading ucode lid. index=200ea
[   53.462749486,2] NVRAM: Failed to load
[   53.462819095,2] NVRAM: Failed to load
[   53.462894236,2] NVRAM: Failed to load
[   53.462967071,2] NVRAM: Failed to load
[   53.463033077,2] NVRAM: Failed to load
[   53.463144847,2] NVRAM: Failed to load
```

  Eventually followed by:

```
[   57.216942479,5] INIT: platform wait for kernel load failed
[   57.217051132,5] INIT: Assuming kernel at 0x20000000
[   57.217127508,3] INIT: ELF header not found. Assuming raw binary.
[   57.217249886,2] NVRAM: Failed to load
[   57.221294487,0] FATAL: Kernel is zeros, can't execute!
[   57.221397429,0] Assert fail: core/init.c:615:0
[   57.221471414,0] Aborting!
CPU 0028 Backtrace:
 S: 0000000031d43c60 R: 000000003001b274   ._abort+0x4c
```

(continues on next page)

(continued from previous page)

```
 S: 0000000031d43ce0 R: 000000003001b2f0   .assert_fail+0x34
 S: 0000000031d43d60 R: 0000000030014814   .load_and_boot_kernel+0xae4
 S: 0000000031d43e30 R: 0000000030015164   .main_cpu_entry+0x680
 S: 0000000031d43f00 R: 0000000030002718   boot_entry+0x1c0
 --- OPAL boot ---
```

Analysis of the execution paths suggests we'll always "safely" end this way due the setup sequence for the blocklevel callbacks in flash_init() and error handling in blocklevel_get_info(), and there's no current risk of executing from unexpected memory locations. As such the issue is reduced to down to a fix for poor error hygene in the original change and a resolution for a Coverity warning (famous last words etc).

- core/flash: Retry requests as necessary in flash_load_resource()

  We would like to successfully boot if we have a dependency on the BMC for flash even if the BMC is not current ready to service flash requests. On the assumption that it will become ready, retry for several minutes to cover a BMC reboot cycle and *eventually* rather than *immediately* crash out with:

```
[  269.549748] reboot: Restarting system
[  390.297462587,5] OPAL: Reboot request...
[  390.297737995,5] RESET: Initiating fast reboot 1...
[  391.074707590,5] Clearing unused memory:
[  391.075198880,5] PCI: Clearing all devices...
[  391.075201618,7] Clearing region 201ffe000000-201fff800000
[  391.086235699,5] PCI: Resetting PHBs and training links...
[  391.254089525,3] FFS: Error 17 reading flash header
[  391.254159668,3] FLASH: Can't open ffs handle: 17
[  392.307245135,5] PCI: Probing slots...
[  392.363723191,5] PCI Summary:
...
[  393.423255262,5] OCC: All Chip Rdy after 0 ms
[  393.453092828,5] INIT: Starting kernel at 0x20000000, fdt at
0x30800a88 390645 bytes
[  393.453202605,0] FATAL: Kernel is zeros, can't execute!
[  393.453247064,0] Assert fail: core/init.c:593:0
[  393.453289682,0] Aborting!
CPU 0040 Backtrace:
 S: 0000000031e03ca0 R: 000000003001af60   ._abort+0x4c
 S: 0000000031e03d20 R: 000000003001afdc   .assert_fail+0x34
 S: 0000000031e03da0 R: 00000000300146d8   .load_and_boot_kernel+0xb30
 S: 0000000031e03e70 R: 0000000030026cf0   .fast_reboot_entry+0x39c
 S: 0000000031e03f00 R: 0000000030002a4c   fast_reset_entry+0x2c
 --- OPAL boot ---
```

  The OPAL flash API hooks directly into the blocklevel layer, so there's no delay for e.g. the host kernel, just for asynchronously loaded resources during boot.

- fast-reboot: occ: Call occ_pstates_init() on fast-reset on all machines

  Commit 815417dcda2e ("init, occ: Initialise OCC earlier on BMC systems") conditionally invoked occ_pstates_init() only on FSP based systems in load_and_boot_kernel(). Due to this pstate table is re-parsed on FSP system and skipped on BMC system during fast-reboot. So this patch fixes this by invoking occ_pstates_init() on all boxes during fast-reboot.

- opal/hmi: Don't retry TOD recovery if it is already in failed state.

  On TOD failure, all cores/thread receives HMI and very first thread that gets interrupt fixes the TOD where as others just resets the respective HMER error bit and return. But when TOD is unrecoverable, all the threads try to do TOD recovery one by one causing threads to spend more time inside opal. Set a global flag when TOD is unrecoverable so that rest of the threads go back to linux immediately avoiding lock ups in system reboot/panic path.

- hw/bt: Do not disable ipmi message retry during OPAL boot

  Currently OPAL doesn't know whether BMC is functioning or not. If BMC is down (like BMC reboot), then we keep on retry sending message to BMC. So in some corner cases we may hit hard lockup issue in kernel.

  Ideally we should avoid using synchronous path as much as possible. But for now commit 01f977c3 added option to disable message retry in synchronous. But this fix is not required during boot. Hence lets disable IPMI message retry during OPAL boot.

- hdata/memory: Fix warning message

  Even though we added memory to device tree, we are getting below warning.

```
[   57.136949696,3] Unable to use memory range 0 from MSAREA 0
[   57.137049753,3] Unable to use memory range 0 from MSAREA 1
[   57.137152335,3] Unable to use memory range 0 from MSAREA 2
[   57.137251218,3] Unable to use memory range 0 from MSAREA 3
```

- hw/bt: Add backend interface to disable ipmi message retry option

  During boot OPAL makes IPMI_GET_BT_CAPS call to BMC to get BT interface capabilities which includes IPMI message max resend count, message timeout, etc,. Most of the time OPAL gets response from BMC within specified timeout. In some corner cases (like mboxd daemon reset in BMC, BMC reboot, etc) OPAL may not get response within timeout period. In such scenarios, OPAL resends message until max resend count reaches.

  OPAL uses synchronous IPMI message (ipmi_queue_msg_sync()) for few operations like flash read, write, etc. Thread will wait in OPAL until it gets response from BMC. In some corner cases like BMC reboot, thread may wait in OPAL for long time (more than 20 seconds) and results in kernel hardlockup.

  This patch introduces new interface to disable message resend option. We will disable message resend option for synchrous message. This will greatly reduces kernel hardlock up issues.

  This is short term fix. Long term solution is to convert all synchronous messages to asynhrounous one.

- ipmi/power: Fix system reboot issue

  Kernel makes reboot/shudown OPAL call for reboot/shutdown. Once kernel gets response from OPAL it runs opal_poll_events() until firmware handles the request.

  On BMC based system, OPAL makes IPMI call (IPMI_CHASSIS_CONTROL) to initiate system reboot/shutdown. At present OPAL queues IPMI messages and return SUCESS to Host. If BMC is not ready to accept command (like BMC reboot), then these message will fail. We have to manually reboot/shutdown the system using BMC interface.

  This patch adds logic to validate message return value. If message failed, then it will resend the message. At some stage BMC will be ready to accept message and handles IPMI message.

- firmware-versions: Add test case for parsing VERSION

  Also make it possible to use with afl-lop/afl-fuzz just to help make *sure* we're all good.

  Additionally, if we hit a entry in VERSION that is larger than our buffer size, we skip over it gracefully rather than overwriting the stack. This is only a problem if VERSION isn't trusted, which as of 4b8cc05a94513816d43fb8bd6178896b430af08f it is verified as part of Secure Boot.

- core/fast-reboot: improve NMI handling during fast reset

  Improve sreset and MCE handling in fast reboot. Switch the HILE bit off before copying OPAL's exception vectors, so NMIs can be handled properly. Also disable MSR[ME] while the vectors are being overwritten

- core/cpu: HID update race

  If the per-core HID register is updated concurrently by multiple threads, updates can get lost. This has been observed during fast reboot where the HILE bit does not get cleared on all cores, which can cause machine check exception interrupts to crash.

  Fix this by only updating HID on thread0.

- SLW: Print verbose info on errors only

  Change print level from debug to warning for reporting bad EC_PPM_SPECIAL_WKUP_* scom values. To reduce cluttering in the log print only on error.

Since v6.3-rc2:

- hw/xscom: add missing P9P chip name
- asm/head: balance branches to avoid link stack predictor mispredicts

  The Linux wrapper for OPAL call and return is arranged like this:

  ```
  __opal_call:
      mflr   r0
      std    r0,PPC_STK_LROFF(r1)
      LOAD_REG_ADDR(r11, opal_return)
      mtlr   r11
      hrfid  -> OPAL

  opal_return:
      ld     r0,PPC_STK_LROFF(r1)
      mtlr   r0
      blr
  ```

  When skiboot returns to Linux, it branches to LR (i.e., opal_return) with a blr. This unbalances the link stack predictor and will cause mispredicts back up the return stack.

- external/mambo: also invoke readline for the non-autorun case
- asm/head.S: set POWER9 radix HID bit at entry

  When running in virtual memory mode, the radix MMU hid bit should not be changed, so set this in the initial boot SPR setup.

  As a side effect, fast reboot also has HID0:RADIX bit set by the shared spr init, so no need for an explicit call.

- build: link with –orphan-handling=warn

  The linker can warn when the linker script does not explicitly place all sections. These orphan sections are placed according to heuristics, which may not always be desirable. Enable this warning.

- build: -fno-asynchronous-unwind-tables

  skiboot does not use unwind tables, this option saves about 100kB, mostly from .text.

- opal/hmi: Initialize the hmi event with old value of TFMR.

  Do this before we fix TFAC errors. Otherwise the event at host console shows no thread error reported in TFMR register.

  Without this patch the console event show TFMR with no thread error: (DEC parity error TFMR[59] injection)

  ```
  [   53.737572] Severe Hypervisor Maintenance interrupt [Recovered]
  [   53.737596]  Error detail: Timer facility experienced an error
  [   53.737611]  HMER: 0840000000000000
  [   53.737621]  TFMR: 3212000870e04000
  ```

After this patch it shows old TFMR value on host console:

```
[ 2302.267271] Severe Hypervisor Maintenance interrupt [Recovered]
[ 2302.267305]  Error detail: Timer facility experienced an error
[ 2302.267320]  HMER: 0840000000000000
[ 2302.267330]  TFMR: 3212000870e14010
```

IBM FSP based platforms

- platforms/firenze: Rework I2C controller fixups
- platforms/zz: Re-enable LXVPD slot information parsing

  From memory this was disabled in the distant past since we were waiting for an updates to the LXPVD format. It looks like that never happened so re-enable it for the ZZ platform so that we can get PCI slot location codes on ZZ.

HIOMAP

- astbmc: Try IPMI HIOMAP for P8

  The HIOMAP protocol was developed after the release of P8 in preparation for P9. As a consequence P9 always uses it, but it has rarely been enabled for P8. P8DTU has recently added IPMI HIOMAP support to its BMC firmware, so enable its use in skiboot with P8 machines. Doing so requires some rework to ensure fallback works correctly as in the past the fallback was to mbox, which will only work for P9.
- libflash/ipmi-hiomap: Enforce message size for empty response

  The protocol defines the response to the associated messages as empty except for the command ID and sequence fields. If the BMC is returning extra data consider the message malformed.
- libflash/ipmi-hiomap: Remove unused close handling

  Issuing a HIOMAP_C_CLOSE is not required by the protocol specification, rather a close can be implicit in a subsequent CREATE_{READ,WRITE}_WINDOW request. The implicit close provides an opportunity to reduce LPC traffic and the implementation takes up that optimisation, so remove the case from the IPMI callback handler.
- libflash/ipmi-hiomap: Overhaul event handling

  Reworking the event handling was inspired by a bug report by Vasant where the host would get wedged on multiple flash access attempts in the face of a persistent error state on the BMC-side. The cause of this bug was the early-exit based on ctx->update, which erronously assumed that all events had been completely handled in prior calls to ipmi_hiomap_handle_events(). This is not true if e.g. HIOMAP_E_DAEMON_READY is clear in the prior calls.

  Regardless, there were other correctness and efficiency problems with the handling strategy:

  – Ack-able event state was not restored in the face of errors in the process of re-establishing protocol state
  – It forced needless window restoration with respect to the context in which ipmi_hiomap_handle_events() was called.
  – Tests for HIOMAP_E_DAEMON_READY and HIOMAP_E_FLASH_LOST were redundant with the overhauled error handling introduced in the previous patch

  Fix all of the above issues and add comments to explain the event handling flow.

- libflash/ipmi-hiomap: Overhaul error handling

  The aim is to improve the robustness with respect to absence of the BMC-side daemon. The current error handling roughly mirrors what was done for the mailbox implementation, but there's room for improvement.

  Errors are split into two classes, those that affect the transport state and those that affect the window validity. From here, we push the transport state error checks right to the bottom of the stack, to ensure the link is known to be in a good state before any message is sent. Window validity tests remain as they were in the hiomap_window_move() and ipmi_hiomap_read() functions. Validity tests are not necessary in the write and erase paths as we will receive an error response from the BMC when performing a dirty or flush on an invalid window.

  Recovery also remains as it was, done on entry to the blocklevel callbacks. If an error state is encountered in the middle of an operation no attempt is made to recover it on the spot, instead the error is returned up the stack and the caller can choose how it wishes to respond.

- libflash/ipmi-hiomap: Fix leak of msg in callback

Since v6.3-rc1:

- libflash/ipmi-hiomap: Fix blocks count issue

  We convert data size to block count and pass block count to BMC. If data size is not block aligned then we endup sending block count less than actual data. BMC will write partial data to flash memory.

  Sample log

  ```
  [  594.388458416,7] HIOMAP: Marked flash dirty at 0x42010 for 8
  [  594.398756487,7] HIOMAP: Flushed writes
  [  594.409596439,7] HIOMAP: Marked flash dirty at 0x42018 for 3970
  [  594.419897507,7] HIOMAP: Flushed writes
  ```

  In this case HIOMAP sent data with block count=0 and hence BMC didn't flush data to flash.

POWER8

- hw/phb3/naples: Disable D-states

  Putting "Mellanox Technologies MT27700 Family [ConnectX-4] [15b3:1013]" (more precisely, the second of 2 its PCI functions, no matter in what order) into the D3 state causes EEH with the "PCT timeout" error. This has been noticed on garrison machines only and firestones do not seem to have this issue.

  This disables D-states changing for devices on root buses on Naples by installing a config space access filter (copied from PHB4).

- cpufeatures: Always advertise POWER8NVL as DD2

  Despite the major version of PVR being 1 (0x004c0100) for POWER8NVL, these chips are functionally equalent to P8/P8E DD2 levels.

  This advertises POWER8NVL as DD2. As the result, skiboot adds ibm,powerpc-cpu-features/processor-control-facility for such CPUs and the linux kernel can use hypervisor doorbell messages to wake secondary threads; otherwise "KVM: CPU %d seems to be stuck" would appear because of missing LPCR_PECEDH.

p8dtu Platform

- p8dtu: Configure BMC graphics

  We can no-longer read the values from the BMC in the way we have in the past. Values were provided by Eric Chen of SMC.

- p8dtu: Enable HIOMAP support

Vesnin Platform

- platforms/vesnin: Disable PCIe port bifurcation

  PCIe ports connected to CPU1 and CPU3 now work as x16 instead of x8x8.

- Fix hang in pnv_platform_error_reboot path due to TOD failure.

  On TOD failure, with TB stuck, when linux heads down to pnv_platform_error_reboot() path due to unrecoverable hmi event, the panic cpu gets stuck in OPAL inside ipmi_queue_msg_sync(). At this time, rest all other cpus are in smp_handle_nmi_ipi() waiting for panic cpu to proceed. But with panic cpu stuck inside OPAL, linux never recovers/reboot.

```
p0 c1 t0
NIA : 0x000000003001dd3c <.time_wait+0x64>
CFAR : 0x000000003001dce4 <.time_wait+0xc>
MSR : 0x9000000002803002
LR : 0x000000003002ecf8 <.ipmi_queue_msg_sync+0xec>

STACK: SP NIA
0x0000000031c236e0 0x0000000031c23760 (big-endian)
0x0000000031c23760 0x000000003002ecf8 <.ipmi_queue_msg_sync+0xec>
0x0000000031c237f0 0x00000000300aa5f8 <.hiomap_queue_msg_sync+0x7c>
0x0000000031c23880 0x00000000300aaadc <.hiomap_window_move+0x150>
0x0000000031c23950 0x00000000300ab1d8 <.ipmi_hiomap_write+0xcc>
0x0000000031c23a90 0x00000000300a7b18 <.blocklevel_raw_write+0xbc>
0x0000000031c23b30 0x00000000300a7c34 <.blocklevel_write+0xfc>
0x0000000031c23bf0 0x0000000030030be0 <.flash_nvram_write+0xd4>
0x0000000031c23c90 0x000000003002c128 <.opal_write_nvram+0xd0>
0x0000000031c23d20 0x00000000300051e4 <opal_entry+0x134>
0xc000001fea6e7870 0xc0000000000a9060 <opal_nvram_write+0x80>
0xc000001fea6e78c0 0xc000000000030b84 <nvram_write_os_partition+0x94>
0xc000001fea6e7960 0xc0000000000310b0 <nvram_pstore_write+0xb0>
0xc000001fea6e7990 0xc0000000004792d4 <pstore_dump+0x1d4>
0xc000001fea6e7ad0 0xc00000000018a570 <kmsg_dump+0x140>
0xc000001fea6e7b40 0xc000000000028e5c <panic_flush_kmsg_end+0x2c>
0xc000001fea6e7b60 0xc0000000000a7168 <pnv_platform_error_reboot+0x68>
0xc000001fea6e7bd0 0xc0000000000ac9b8 <hmi_event_handler+0x1d8>
0xc000001fea6e7c80 0xc00000000012d6c8 <process_one_work+0x1b8>
0xc000001fea6e7d20 0xc00000000012da28 <worker_thread+0x88>
0xc000001fea6e7db0 0xc0000000001366f4 <kthread+0x164>
0xc000001fea6e7e20 0xc00000000000b65c <ret_from_kernel_thread+0x5c>
```

  This is because, there is a while loop towards the end of ipmi_queue_msg_sync() which keeps looping until "sync_msg" does not match with "msg". It loops over time_wait_ms() until exit condition is met. In normal scenario time_wait_ms() calls run pollers so that ipmi backend gets a chance to check ipmi response and set sync_msg to NULL.

```
while (sync_msg == msg)
        time_wait_ms(10);
```

  But in the event when TB is in failed state time_wait_ms()->time_wait_poll() returns immediately without calling pollers and hence we end up looping forever. This patch fixes this hang by calling opal_run_pollers() in TB failed state as well.

POWER9

- Retry link training at PCIe GEN1 if presence detected but training repeatedly failed

  Certain older PCIe 1.0 devices will not train unless the training process starts at GEN1 speeds. As a last resort when a device will not train, fall back to GEN1 speed for the last training attempt.

  This is verified to fix devices based on the Conexant CX23888 on the Talos II platform.

- hw/phb4: Drop FRESET_DEASSERT_DELAY state

  The delay between the ASSERT_DELAY and DEASSERT_DELAY states is set to one timebase tick. This state seems to have been a hold over from PHB3 where it was used to add a 1s delay between de-asserting PERST and polling the link for the CAPI FPGA. There's no requirement for that here since the link polling on PHB4 is a bit smarter so we should be fine.

- hw/phb4: Factor out PERST control

  Some time ago Mikey added some code work around a bug we found where a certain RAID card wouldn't come back again after a fast-reboot. The workaround is setting the Link Disable bit before asserting PERST and clear it after de-asserting PERST.

  Currently we do this in the FRESET path, but not in the CRESET path. This patch moves the PERST control into its own function to reduce duplication and to the workaround is applied in all circumstances.

- hw/phb4: Remove FRESET presence check

  When we do an freset the first step is to check if a card is present in the slot. However, this only occurs when we enter phb4_freset() with the slot state set to SLOT_NORMAL. This occurs in:

  a) The creset path, and

  b) When the OS manually requests an FRESET via an OPAL call.

  (a) is problematic because in the boot path the generic code will put the slot into FRESET_START manually before calling into phb4_freset(). This can result in a situation where a device is detected on boot, but not after a CRESET.

  I've noticed this occurring on systems where the PHB's slot presence detect signal is not wired to an adapter. In this situation we can rely on the in-band presence mechanism, but the presence check will make us exit before that has a chance to work.

  Additionally, if we enter from the CRESET path this early exit leaves the slot's PERST signal being left asserted. This isn't currently an issue, but if we want to support hotplug of devices into the root port it will be.

- hw/phb4: Skip FRESET PERST when coming from CRESET

  PERST is asserted at the beginning of the CRESET process to prevent the downstream device from interacting with the host while the PHB logic is being reset and re-initialised. There is at least a 100ms wait during the CRESET processing so it's not necessary to wait this time again in the FRESET handler.

  This patch extends the delay after re-setting the PHB logic to extend to the 250ms PERST wait period that we typically use and sets the skip_perst flag so that we don't wait this time again in the FRESET handler.

- hw/phb4: Look for the hub-id from in the PBCQ node

  The hub-id is stored in the PBCQ node rather than the stack node so we never add it to the PHB node. This breaks the lxvpd slot lookup code since the hub-id is encoded in the VPD record that we need to find the slot information.

- hdata/iohub: Look for IOVPD on P9

  P8 and P9 use the same IO VPD setup, so we need to load the IOHUB VPD on P9 systems too.

Since v6.3-rc2:

- hw/phb4: Squash the IO bridge window

  The PCI-PCI bridge spec says that bridges that implement an IO window should hardcode the IO base and limit registers to zero. Unfortunately, these registers only define the upper bits of the IO window and the low bits are assumed to be 0 for the base and 1 for the limit address. As a result, setting both to zero can be mis-interpreted as a 4K IO window.

  This patch fixes the problem the same way PHB3 does. It sets the IO base and limit values to 0xf000 and 0x1000 respectively which most software interprets as a disabled window.

  lspci before patch:

  ```
  0000:00:00.0 PCI bridge: IBM Device 04c1 (prog-if 00 [Normal decode])
          I/O behind bridge: 00000000-00000fff
  ```

  lspci after patch:

  ```
  0000:00:00.0 PCI bridge: IBM Device 04c1 (prog-if 00 [Normal decode])
          I/O behind bridge: None
  ```

- hw/xscom: Enable sw xstop by default on p9

  This was disabled at some point during bringup to make life easier for the lab folks trying to debug NVLink issues. This hack really should have never made it out into the wild though, so we now have the following situation occuring in the field:

  1) A bad happens
  2) The host kernel recieves an unrecoverable HMI and calls into OPAL to request a platform reboot.
  3) OPAL rejects the reboot attempt and returns to the kernel with OPAL_PARAMETER.
  4) Kernel panics and attempts to kexec into a kdump kernel.

  A side effect of the HMI seems to be CPUs becoming stuck which results in the initialisation of the kdump kernel taking a extremely long time (6+ hours). It's also been observed that after performing a dump the kdump kernel then crashes itself because OPAL has ended up in a bad state as a side effect of the HMI.

  All up, it's not very good so re-enable the software checkstop by default. If people still want to turn it off they can using the nvram override.

CAPI2

- capp/phb4: Prevent HMI from getting triggered when disabling CAPP

  While disabling CAPP an HMI gets triggered as soon as ETU is put in reset mode. This is caused as before we can disabled CAPP, it detects PHB link going down and triggers an HMI requesting Opal to perform CAPP recovery. This has an un-intended side effect of spamming the Opal logs with malfunction alert messages and may also confuse the user.

  To prevent this we mask the CAPP FIR error 'PHB Link Down' Bit(31) when we are disabling CAPP just before we put ETU in reset in phb4_creset(). Also now since bringing down the PHB link now wont trigger an HMI and CAPP recovery, hence we manually set the PHB4_CAPP_RECOVERY flag on the phb to force recovery during creset.

- phb4/capp: Implement sequence to disable CAPP and enable fast-reset

  We implement h/w sequence to disable CAPP in disable_capi_mode() and with it also enable fast-reset for CAPI mode in phb4_set_capi_mode().

Sequence to disable CAPP is executed in three phases. The first two phase is implemented in disable_capi_mode() where we reset the CAPP registers followed by PEC registers to their init values. The final third final phase is to reset the PHB CAPI Compare/Mask Register and is done in phb4_init_ioda3(). The reason to move the PHB reset to phb4_init_ioda3() is because by the time Opal PCI reset state machine reaches this function the PHB is already un-fenced and its configuration registers accessible via mmio.

- capp/phb4: Force CAPP to PCIe mode during kernel shutdown

  This patch introduces a new opal syncer for PHB4 named phb4_host_sync_reset(). We register this opal syncer when CAPP is activated successfully in phb4_set_capi_mode() so that it will be called at kernel shutdown during fast-reset.

  During kernel shutdown the function will then repeatedly call phb->ops->set_capi_mode() to switch switch CAPP to PCIe mode. In case set_capi_mode() indicates its OPAL_BUSY, which indicates that CAPP is still transitioning to new state; it calls slot->ops.run_sm() to ensure that Opal slot reset state machine makes forward progress.

Witherspoon Platform

- platforms/witherspoon: Make PCIe shared slot error message more informative

  If we're missing chips for some reason, we print a warning when configuring the PCIe shared slot.

  The warning doesn't really make it clear what "shared slot" is, and if it's printed, it'll come right after a bunch of messages about NPU setup, so let's clarify the message to explicitly mention PCI.

- witherspoon: Add nvlink2 interconnect information

  See *New Features* for details.

Zaius Platform

- zaius: Add BMC description

  Frederic reported that Zaius was failing with a NULL dereference when trying to initialise IPMI HIOMAP. It turns out that the BMC wasn't described at all, so add a description.

p9dsu platform

- p9dsu: Fix p9dsu default variant

  Add the default when no riser_id is returned from the ipmi query.

  Allow a little more time for BMC reply and cleanup some label strings.

PCIe

See *POWER9* for POWER9 specific PCIe changes.

- core/pcie-slot: Don't bail early in the power on case

  Exiting early in the power off case makes sense since we can't disable slot power (or assert PERST) for suprise hotplug slots. However, we should not exit early in the power-on case since it's possible slot power may have been disabled (or just not enabled at boot time).

- firenze-pci: Always init slot info from LXVPD

  We can slot information from the LXVPD without having power control information about that slot. This patch changes the init path so that we always override the add_properties() call rather than only when we have power control information about the slot.

- fsp/lxvpd: Print more LXVPD slot information

  Useful to know since it changes the behaviour of the slot core.

- core/pcie-slot: Set power state from the PWRCTL flag

  For some reason we look at the power control indicator and use that to determine if the slot is "off" rather than the power control flag that is used to power down the slot.

  While we're here change the default behaviour so that the slot is assumed to be powered on if there's no slot capability, or if there's no power control available.

- core/pci: Increase the max slot string size

  The maximum string length for the slot label / device location code in the PCI summary is currently 32 characters. This results in some IBM location codes being truncated due to their length, e.g.

```
PHB#0001:02:11.0 [SWDN]  SLOT=C11  x8
PHB#0001:13:00.0 [EP  ] *snip* LOC_CODE=U78D3.ND1.WZS004A-P1-C
PHB#0001:13:00.1 [EP  ] *snip* LOC_CODE=U78D3.ND1.WZS004A-P1-C
PHB#0001:13:00.2 [EP  ] *snip* LOC_CODE=U78D3.ND1.WZS004A-P1-C
PHB#0001:13:00.3 [EP  ] *snip* LOC_CODE=U78D3.ND1.WZS004A-P1-C
```

  Which obscure the actual location of the card, and it looks bad. This patch increases the maximum length of the label string to 80 characters since that's the maximum length for a location code.

Since v6.3-rc3:

- pci: Try harder to add meaningful ibm,loc-code

  We keep the existing logic of looking to the parent for the slot-label or slot-location-code, but we add logic to (if all that fails) we look directly for the slot-location-code (as this should give us the correct loc code for things directly under the PHB), and otherwise we just look for a loc-code.

  The applicable bit of PAPR here is:

  > R1–12.1–1. Each instance of a hardware entity (FRU) has a platform unique location code and any node in the OF device tree that describes a part of a hardware entity must include the "ibm,loc-code" property with a value that represents the location code for that hardware entity.

  which we weren't really fully obeying at any recent (ever?) point in time. Now we should do okay, at least for PCI.

Since v6.3-rc2: - core/pci: Use PHB io-base-location by default for PHB slots

  On witherspoon only the GPU slots and the three pluggable PCI slots (SLOT0, 1, 2) have platform defined slot names. For builtin devices such as the SATA controller or the PLX switch that fans out to the GPU slots we have no location codes which some people consider an issue.

  This patch address the problem by making the ibm,slot-location-code for the root port device default to the ibm,io-base-location-code which is typically the location code for the system itself.

  e.g.

```
pciex@600c3c0100000/ibm,loc-code
                 "UOPWR.0000000-Node0-Proc0"
```

(continues on next page)

(continued from previous page)

```
pciex@600c3c0100000/pci@0/ibm,loc-code
                    "UOPWR.0000000-Node0-Proc0"

pciex@600c3c0100000/pci@0/usb-xhci@0/ibm,loc-code
                    "UOPWR.0000000-Node0"
```

The PHB node, and the root complex nodes have a loc code of the processor they are attached to, while the usb-xhci device under the root port has a location code of the system itself.

- hw/phb4: Read ibm,loc-code from PBCQ node

  On P9 the PBCQs are subdivided by stacks which implement the PCI Express logic. When phb4 was forked from phb3 most of the properties that were in the pbcq node moved into the stack node, but ibm,loc-code was not one of them. This patch fixes the phb4 init sequence to read the base location code from the PBCQ node (parent of the stack node) rather than the stack node itself.

OpenCAPI

- npu2/hw-procedures: Fix parallel zcal for opencapi

  For opencapi, we currently do impedance calibration when initializing the PHY for the device, which could run in parallel if we have multiple opencapi devices. But if 2 devices are on the same obus, the 2 calibration sequences could overlap, which likely yields bad results and is useless anyway since it only needs to be done once per obus.

  This patch splits the opencapi PHY reset in 2 parts:

  – a ‘init’ part called serially at boot. That’s when zcal is done. If we have 2 devices on the same socket, the zcal won’t be redone, since we’re called serially and we’ll see it has already be done for the obus

  – a ‘reset’ part called during fundamental reset as a prereq for link training. It does the PHY setup for a set of lanes and the dccal.

  The PHY team confirmed there’s no dependency between zcal and the other reset steps and it can be moved earlier.

- npu2-hw-procedures: Fix zcal in mixed opencapi and nvlink mode

  The zcal procedure needs to be run once per obus. We keep track of which obus is already calibrated in an array indexed by the obus number. However, the obus number is inferred from the brick index, which works well for nvlink but not for opencapi.

  Create an obus_index() function, which, from a device, returns the correct obus index, irrespective of the device type.

- npu2-opencapi: Fix adapter reset when using 2 adapters

  If two opencapi adapters are on the same obus, we may try to train the two links in parallel at boot time, when all the PCI links are being trained. Both links use the same i2c controller to handle the reset signal, so some care is needed to make sure resetting one doesn’t interfere with the reset of the other. We need to keep track of the current state of the i2c controller (and use locking).

  This went mostly unnoticed as you need to have 2 opencapi cards on the same socket and links tended to train anyway because of the retries.

- npu2-opencapi: Extend delay after releasing reset on adapter

  Give more time to the FPGA to process the reset signal. The previous delay, 5ms, is too short for newer adapters with bigger FPGAs. Extend it to 250ms. Ultimately, that delay will likely end up being added to the opencapi specification, but we are not there yet.

- npu2-opencapi: ODL should be in reset when enabled

  We haven't hit any problem so far, but from the ODL designer, the ODL should be in reset when it is enabled.

  The ODL remains in reset until we start a fundamental reset to initiate link training. We still assert and deassert the ODL reset signal as part of the normal procedure just before training the link. Asserting is therefore useless at boot, since the ODL is already in reset, but we keep it as it's only a scom write and it's needed when we reset/retrain from the OS.

- npu2-opencapi: Keep ODL and adapter in reset at the same time

  Split the function to assert and deassert the reset signal on the ODL, so that we can keep the ODL in reset while we reset the adapter, therefore having a window where both sides are in reset.

  It is actually not required with our current DLx at boot time, but I need to split the ODL reset function for the following patch and it will become useful/required later when we introduce resetting an opencapi link from the OS.

- npu2-opencapi: Setup perf counters to detect CRC errors

  It's possible to set up performance counters for the PLL to detect various conditions for the links in nvlink or opencapi mode. Since those counters are currently unused, let's configure them when an obus is in opencapi mode to detect CRC errors on the link. Each link has two counters: - CRC error detected by the host - CRC error detected by the DLx (NAK received by the host)

  We also dump the counters shortly after the link trains, but they can be read multiple times through cronus, pdbg or linux. The counters are configured to be reset after each read.

Since v6.3-rc1:

- opal/hmi: Never trust a cow!

  With opencapi, it's fairly common to trigger HMIs during AFU development on the FPGA, by not replying in time to an NPU command, for example. So shift the blame reported by that cow to avoid crowding my mailbox.

- hw/npu2: Dump (more) npu2 registers on link error and HMIs

  We were already logging some NPU registers during an HMI. This patch cleans up a bit how it is done and separates what is global from what is specific to nvlink or opencapi.

  Since we can now receive an error interrupt when an opencapi link goes down unexpectedly, we also dump the NPU state but we limit it to the registers of the brick which hit the error.

  The list of registers to dump was worked out with the hw team to allow for proper debugging. For each register, we print the name as found in the NPU workbook, the scom address and the register value.

- hw/npu2: Report errors to the OS if an OpenCAPI brick is fenced

  Now that the NPU may report interrupts due to the link going down unexpectedly, report those errors to the OS when queried by the 'next_error' PHB callback.

  The hardware doesn't support recovery of the link when it goes down unexpectedly. So we report the PHB as dead, so that the OS can log the proper message, notify the drivers and take the devices down.

- hw/npu2: Fix OpenCAPI PE assignment

  When we support mixing NVLink and OpenCAPI devices on the same NPU, we're going to have to share the same range of 16 PE numbers between NVLink and OpenCAPI PHBs.

  For OpenCAPI devices, PE assignment is only significant for determining which System Interrupt Log register is used for a particular brick - unlike NVLink, it doesn't play any role in determining how links are fenced.

  Split the PE range into a lower half which is used for NVLink, and an upper half that is used for OpenCAPI, with a fixed PE number assigned per brick.

As the PE assignment for OpenCAPI devices is fixed, set the PE once during device init and then ignore calls to the set_pe() operation.

- opal-api: Reserve 2 OPAL API calls for future OpenCAPI LPC use

  OpenCAPI Lowest Point of Coherency (LPC) memory is going to require some extra OPAL calls to set up NPU BARs. These calls will most likely be called OPAL_NPU_LPC_ALLOC and OPAL_NPU_LPC_RELEASE, we're not quite ready to upstream that code yet though.

NVLINK2

- npu2: Allow ATSD for LPAR other than 0

  Each XTS MMIO ATSD# register is accompanied by another register - XTS MMIO ATSD0 LPARID# - which controls LPID filtering for ATSD transactions.

  When a host system passes a GPU through to a guest, we need to enable some ATSD for an LPAR. At the moment the host assigns one ATSD to a NVLink bridge and this maps it to an LPAR when GPU is assigned to the LPAR. The link number is used for an ATSD index.

  ATSD6&7 stay mapped to the host (LPAR=0) all the time which seems to be acceptable price for the simplicity.

- npu2: Add XTS_BDF_MAP wildcard refcount

  Currently PID wildcard is programmed into the NPU once and never cleared up. This works for the bare metal as MSR does not change while the host OS is running.

  However with the device virtualization, we need to keep track of wildcard entries use and clear them up before switching a GPU from a host to a guest or vice versa.

  This adds refcount to a NPU2, one counter per wildcard entry. The index is a short lparid (4 bits long) which is allocated in opal_npu_map_lpar() and should be smaller than NPU2_XTS_BDF_MAP_SIZE (defined as 16).

Since v6.3-rc2: - npu2: Disable Probe-to-Invalid-Return-Modified-or-Owned snarfing by default

  V100 GPUs are known to violate NVLink2 protocol in some cases (one is when memory was accessed by the CPU and they by GPU using so called block linear mapping) and issue double probes to NPU which can cope with this problem only if CONFIG_ENABLE_SNARF_CPM ("disable/enable Probe.I.MO snarfing a cp_m") is not set in the CQ_SM Misc Config register #0. If the bit is set (which is the case today), NPU issues the machine check stop.

  The snarfing feature is designed to detect 2 probes in flight and combine them into one.

  This adds a new "opal-npu2-snarf-cpm" nvram variable which controls CONFIG_ENABLE_SNARF_CPM for all NVLinks to prevent the machine check stop from happening.

  This disables snarfing by default as otherwise a broken GPU driver can crash the entire box even when a GPU is passed through to a guest. This provides a dial to allow regression tests (might be useful for a bare metal). To enable snarfing, the user needs to run:

  ```
  sudo nvram -p ibm,skiboot --update-config opal-npu2-snarf-cpm=enable
  ```

  and reboot the host system.

- hw/npu2: Show name of opencapi error interrupts

Debugging and simulation

- external/mambo: Error out if kernel is too large

If you're trying to boot a gigantic kernel in mambo (which you can reproduce by building a kernel with CONFIG_MODULES=n) you'll get misleading errors like:

```
WARNING: 0: (0): [0:0]: Invalid/unsupported instr 0x00000000[INVALID]
WARNING: 0: (0):  PC(EA): 0x0000000030000010 PC(RA):0x0000000030000010 MSR:␣
↪0x9000000000000000 LR: 0x0000000000000000
WARNING: 0: (0):  numInstructions = 0
WARNING: 1: (1): [0:0]: Invalid/unsupported instr 0x00000000[INVALID]
WARNING: 1: (1):  PC(EA): 0x0000000000000E40 PC(RA):0x0000000000000E40 MSR:␣
↪0x9000000000000000 LR: 0x0000000000000000
WARNING: 1: (1):  numInstructions = 1
WARNING: 1: (1): Interrupt to 0x0000000000000E40 from 0x0000000000000E40
INFO: 1: (2): ** Execution stopped: Continuous Interrupt, Instruction caused␣
↪exception,  **
```

So add an error to skiboot.tcl to warn the user before this happens. Making PAYLOAD_ADDR further back is one way to do this but if there's a less gross way to generally work around this very niche problem, I can suggest that instead.

- external/mambo: Populate kernel-base-address in the DT

  skiboot.tcl defines PAYLOAD_ADDR as 0x20000000, which is the default in skiboot. This is also the default in skiboot unless kernel-base-address is set in the device tree.

  If you change PAYLOAD_ADDR to something else for mambo, skiboot won't see it because it doesn't set that DT property, so fix it so that it does.

- external/mambo: allow CPU targeting for most debug utils

  Debug util functions target CPU 0:0:0 by default Some can be overidden explicitly per invocation, and others can't at all. Even for those that can be overidden, it is a pain to type them out when you're debugging a particular thread.

  Provide a new 'target' function that allows the default CPU target to be changed. Wire that up that default to all other utils. Provide a new 'S' step command which only steps the target CPU.

- qemu: bt device isn't always hanging off /

  Just use the normal for_each_compatible instead.

  Otherwise in the qemu model as executed by op-test, we wouldn't go down the astbmc_init() path, thus not having flash.

- devicetree: Add p9-simics.dts

  Add a p9-based devicetree that's suitable for use with Simics.

- devicetree: Move power9-phb4.dts

  Clean up the formatting of power9-phb4.dts and move it to external/devicetree/p9.dts. This sets us up to include it as the basis for other trees.

- devicetree: Add nx node to power9-phb4.dts

  A (non-qemu) p9 without an nx node will assert in p9_darn_init():

```
dt_for_each_compatible(dt_root, nx, "ibm,power9-nx")
        break;
if (!nx) {
        if (!dt_node_is_compatible(dt_root, "qemu,powernv"))
              assert(nx);
        return;
}
```

Since NX is this essential, add it to the device tree.

- devicetree: Fix typo in power9-phb4.dts

  Change "impi" to "ipmi".

- devicetree: Fix syntax error in power9-phb4.dts

  Remove the extra space causing this:

  ```
  Error: power9-phb4.dts:156.15-16 syntax error
  FATAL ERROR: Unable to parse input tree
  ```

- core/init: enable machine check on secondaries

  Secondary CPUs currently run with MSR[ME]=0 during boot, whih means if they take a machine check, the system will checkstop.

  Enable ME where possible and allow them to print registers.

Utilities

- pflash: Don't try update RO ToC

  In the future it's likely the ToC will be marked as read-only. Don't error out by assuming its writable.

- pflash: Support encoding/decoding ECC'd partitions

  With the new –ecc option, pflash can add/remove ECC when reading/writing flash partitions protected by ECC.

  This is *not* flawless with current PNORs out in the wild though, as they do not typically fill the whole partition with valid ECC data, so you have to know how big the valid ECC'd data is and specify the size manually. Note that for some partitions this is pratically impossible without knowing the details of the content of the partition.

  A future patch is likely to introduce an option to "stop reading data when ECC starts failing and assume everything is okay rather than error out" to support reading the "valid" data from existing PNOR images.

Since v6.3-rc2:

- opal-prd: Fix memory leak in is-fsp-system check
- opal-prd: Check malloc return value

5.1.135 skiboot-6.3-rc1

skiboot v6.3-rc1 was released on Friday March 29th 2019. It is the first release candidate of skiboot 6.3, which will become the new stable release of skiboot following the 6.2 release, first released December 14th 2018.

Skiboot 6.3 will mark the basis for op-build v2.3. I expect to tag the final skiboot 6.3 in the next week.

skiboot v6.3-rc1 contains all bug fixes as of *skiboot-6.0.19*, and *skiboot-6.2.3* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

This release has been a longer cycle than typical for a variety of reasons. It also contains a lot of cleanup work and minor bug fixes (much like skiboot 6.2 did).

Over skiboot 6.2, we have the following changes:

New Features

- hw/imc: Enable opal calls to init/start/stop IMC Trace mode

  New OPAL APIs for In-Memory Collection Counter infrastructure(IMC), including a new device type called OPAL_IMC_COUNTERS_TRACE.

- xive: Add calls to save/restore the queues and VPs HW state

  To be able to support migration of guests using the XIVE native exploitation mode, (where the queue is effectively owned by the guest), KVM needs to be able to save and restore the HW-modified fields of the queue, such as the current queue producer pointer and generation bit, and to retrieve the modified thread context registers of the VP from the NVT structure : the VP interrupt pending bits.

  However, there is no need to set back the NVT structure on P9. P10 should be the same.

- witherspoon: Add nvlink2 interconnect information

  GPUs on Redbud and Sequoia platforms are interconnected in groups of 2 or 3 GPUs. The problem with that is if the user decides to pass a single GPU from a group to the userspace, we need to ensure that links between GPUs do not get enabled.

  A V100 GPU provides a way to disable selected links. In order to only disable links to peer GPUs, we need a topology map.

  This adds an "ibm,nvlink-peers" property to a GPU DT node with phandles of peer GPUs and NVLink2 bridges. The index in the property is a GPU link number.

- platforms/romulus: Also support talos

  The two are similar enough and I'd like to have a slot table for our Talos.

- OpenCAPI support! (see *OpenCAPI* section)

- opal/hmi: set a flag to inform OS that TOD/TB has failed.

  Set a flag to indicate OS about TOD/TB failure as part of new opal_handle_hmi2 handler. This flag then can be used by OS to make sure functions depending on TB value (e.g. udelay()) are aware of TB not ticking.

- astbmc: Enable IPMI HIOMAP for AMI platforms

  Required for Habanero, Palmetto and Romulus.

- power-mgmt : occ : Add 'freq-domain-mask' DT property

  Add a new device-tree property freq-domain-indicator to define group of CPUs which would share same frequency. This property has been added under power-mgmt node. It is a bitmask.

  Bitwise AND is taken between this bitmask value and PIR of cpu. All the CPUs lying in the same frequency domain will have same result for AND.

  For example, For POWER9, 0xFFF0 indicates quad wide frequency domain. Taking AND with the PIR of CPUs will yield us frequency domain which is quad wise distribution as last 4 bits have been masked which represent the cores.

  Similarly, 0xFFF8 will represent core wide frequency domain for P8.

  Also, Add a new device-tree property domain-runs-at which will denote the strategy OCC is using to change the frequency of a frequency-domain. There can be two strategy - FREQ_MOST_RECENTLY_SET and FREQ_MAX_IN_DOMAIN.

  FREQ_MOST_RECENTLY_SET : the OCC sets the frequency of the quad to the most recent frequency value requested by the CPUs in the quad.

FREQ_MAX_IN_DOMAIN : the OCC sets the frequency of the CPUs in the Quad to the maximum of the latest frequency requested by each of the component cores.

- powercap: occ: Fix the powercapping range allowed for user

  OCC provides two limits for minimum powercap. One being hard powercap minimum which is guaranteed by OCC and the other one is a soft powercap minimum which is lesser than hard-min and may or may not be asserted due to various power-thermal reasons. So to allow the users to access the entire powercap range, this patch exports soft powercap minimum as the "powercap-min" DT property. And it also adds a new DT property called "powercap-hard-min" to export the hard-min powercap limit.

- Add NVDIMM support

  NVDIMMs are memory modules that use a battery backup system to allow the contents RAM to be saved to non-volatile storage if system power goes away unexpectedly. This allows them to be used a high-performance storage device, suitable for serving as a cache for SSDs and the like.

  Configuration of NVDIMMs is handled by hostboot and communicated to OPAL via the HDAT. We need to parse out the NVDIMM memory ranges and create memory regions with the "pmem-region" compatible label to make them available to the host.

- core/exceptions: implement support for MCE interrupts in powersave

  The ISA specifies that MCE interrupts in power saving modes will enter at 0x200 with powersave bits in SRR1 set. This is not currently supported properly, the MCE will just happen like a normal interrupt, but GPRs could be lost, which would lead to crashes (e.g., r1, r2, r13 etc).

  So check the power save bits similarly to the sreset vector, and handle this properly.

- core/exceptions: allow recoverable sreset exceptions

  This requires implementing the MSR[RI] bit. Then just allow all non-fatal sreset exceptions to recover.

- core/exceptions: implement an exception handler for non-powersave sresets

  Detect non-powersave sresets and send them to the normal exception handler which prints registers and stack.

- Add PVR_TYPE_P9P

  Enable a new PVR to get us running on another p9 variant.

Deprecated/Removed Features

- opal: Deprecate reading the PHB status

  The OPAL_PCI_EEH_FREEZE_STATUS call takes a bunch of parameters, one of them is @phb_status. It is defined as __be64* and always NULL in the current Linux upstream but if anyone ever decides to read that status, then the PHB3's handler will assume it is struct OpalIoPhb3ErrorData* (which is a lot bigger than 8 bytes) and zero it causing the stack corruption; p7ioc-phb has the same issue.

  This removes @phb_status from all eeh_freeze_status() hooks and moves the error message from PHB4 to the affected OPAL handlers.

  As far as we can tell, nobody has ever used this and thus it's safe to remove.

- Remove POWER9N DD1 support

  This is not a shipping product and is no longer supported by Linux or other firmware components.

General

- core/i2c: Various bits of refactoring
- refactor backtrace generation infrastructure
- astbmc: Handle failure to initialise raw flash

  Initialising raw flash lead to a dead assignment to rc. Check the return code and take the failure path as necessary. Both before and after the fix we see output along the lines of the following when flash_init() fails:

```
[   53.283182881,7] IRQ: Registering 0800..0ff7 ops @0x300d4b98 (data 0x3052b9d8)
[   53.283184335,7] IRQ: Registering 0ff8..0fff ops @0x300d4bc8 (data 0x3052b9d8)
[   53.283185513,7] PHB#0000: Initializing PHB...
[   53.288260827,4] FLASH: Can't load resource id:0. No system flash found
[   53.288354442,4] FLASH: Can't load resource id:1. No system flash found
[   53.342933439,3] CAPP: Error loading ucode lid. index=200ea
[   53.462749486,2] NVRAM: Failed to load
[   53.462819095,2] NVRAM: Failed to load
[   53.462894236,2] NVRAM: Failed to load
[   53.462967071,2] NVRAM: Failed to load
[   53.463033077,2] NVRAM: Failed to load
[   53.463144847,2] NVRAM: Failed to load
```

  Eventually followed by:

```
[   57.216942479,5] INIT: platform wait for kernel load failed
[   57.217051132,5] INIT: Assuming kernel at 0x20000000
[   57.217127508,3] INIT: ELF header not found. Assuming raw binary.
[   57.217249886,2] NVRAM: Failed to load
[   57.221294487,0] FATAL: Kernel is zeros, can't execute!
[   57.221397429,0] Assert fail: core/init.c:615:0
[   57.221471414,0] Aborting!
CPU 0028 Backtrace:
 S: 0000000031d43c60 R: 000000003001b274   ._abort+0x4c
 S: 0000000031d43ce0 R: 000000003001b2f0   .assert_fail+0x34
 S: 0000000031d43d60 R: 0000000030014814   .load_and_boot_kernel+0xae4
 S: 0000000031d43e30 R: 0000000030015164   .main_cpu_entry+0x680
 S: 0000000031d43f00 R: 0000000030002718   boot_entry+0x1c0
 --- OPAL boot ---
```

  Analysis of the execution paths suggests we'll always "safely" end this way due the setup sequence for the blocklevel callbacks in flash_init() and error handling in blocklevel_get_info(), and there's no current risk of executing from unexpected memory locations. As such the issue is reduced to down to a fix for poor error hygene in the original change and a resolution for a Coverity warning (famous last words etc).

- core/flash: Retry requests as necessary in flash_load_resource()

  We would like to successfully boot if we have a dependency on the BMC for flash even if the BMC is not current ready to service flash requests. On the assumption that it will become ready, retry for several minutes to cover a BMC reboot cycle and *eventually* rather than *immediately* crash out with:

```
[  269.549748] reboot: Restarting system
[  390.297462587,5] OPAL: Reboot request...
[  390.297737995,5] RESET: Initiating fast reboot 1...
[  391.074707590,5] Clearing unused memory:
[  391.075198880,5] PCI: Clearing all devices...
[  391.075201618,7] Clearing region 201ffe000000-201fff800000
[  391.086235699,5] PCI: Resetting PHBs and training links...
```

(continues on next page)

(continued from previous page)

```
[  391.254089525,3] FFS: Error 17 reading flash header
[  391.254159668,3] FLASH: Can't open ffs handle: 17
[  392.307245135,5] PCI: Probing slots...
[  392.363723191,5] PCI Summary:
...
[  393.423255262,5] OCC: All Chip Rdy after 0 ms
[  393.453092828,5] INIT: Starting kernel at 0x20000000, fdt at
0x30800a88 390645 bytes
[  393.453202605,0] FATAL: Kernel is zeros, can't execute!
[  393.453247064,0] Assert fail: core/init.c:593:0
[  393.453289682,0] Aborting!
CPU 0040 Backtrace:
 S: 0000000031e03ca0 R: 000000003001af60   ._abort+0x4c
 S: 0000000031e03d20 R: 000000003001afdc   .assert_fail+0x34
 S: 0000000031e03da0 R: 00000000300146d8   .load_and_boot_kernel+0xb30
 S: 0000000031e03e70 R: 0000000030026cf0   .fast_reboot_entry+0x39c
 S: 0000000031e03f00 R: 0000000030002a4c   fast_reset_entry+0x2c
 --- OPAL boot ---
```

The OPAL flash API hooks directly into the blocklevel layer, so there's no delay for e.g. the host kernel, just for asynchronously loaded resources during boot.

- fast-reboot: occ: Call occ_pstates_init() on fast-reset on all machines

  Commit 815417dcda2e ("init, occ: Initialise OCC earlier on BMC systems") conditionally invoked occ_pstates_init() only on FSP based systems in load_and_boot_kernel(). Due to this pstate table is re-parsed on FSP system and skipped on BMC system during fast-reboot. So this patch fixes this by invoking occ_pstates_init() on all boxes during fast-reboot.

- opal/hmi: Don't retry TOD recovery if it is already in failed state.

  On TOD failure, all cores/thread receives HMI and very first thread that gets interrupt fixes the TOD where as others just resets the respective HMER error bit and return. But when TOD is unrecoverable, all the threads try to do TOD recovery one by one causing threads to spend more time inside opal. Set a global flag when TOD is unrecoverable so that rest of the threads go back to linux immediately avoiding lock ups in system reboot/panic path.

- hw/bt: Do not disable ipmi message retry during OPAL boot

  Currently OPAL doesn't know whether BMC is functioning or not. If BMC is down (like BMC reboot), then we keep on retry sending message to BMC. So in some corner cases we may hit hard lockup issue in kernel.

  Ideally we should avoid using synchronous path as much as possible. But for now commit 01f977c3 added option to disable message retry in synchronous. But this fix is not required during boot. Hence lets disable IPMI message retry during OPAL boot.

- hdata/memory: Fix warning message

  Even though we added memory to device tree, we are getting below warning.

```
[   57.136949696,3] Unable to use memory range 0 from MSAREA 0
[   57.137049753,3] Unable to use memory range 0 from MSAREA 1
[   57.137152335,3] Unable to use memory range 0 from MSAREA 2
[   57.137251218,3] Unable to use memory range 0 from MSAREA 3
```

- hw/bt: Add backend interface to disable ipmi message retry option

  During boot OPAL makes IPMI_GET_BT_CAPS call to BMC to get BT interface capabilities which includes IPMI message max resend count, message timeout, etc,. Most of the time OPAL gets response from BMC within specified timeout. In some corner cases (like mboxd daemon reset in BMC, BMC reboot, etc) OPAL may not get response within timeout period. In such scenarios, OPAL resends message until max resend count reaches.

OPAL uses synchronous IPMI message (ipmi_queue_msg_sync()) for few operations like flash read, write, etc. Thread will wait in OPAL until it gets response from BMC. In some corner cases like BMC reboot, thread may wait in OPAL for long time (more than 20 seconds) and results in kernel hardlockup.

This patch introduces new interface to disable message resend option. We will disable message resend option for synchrous message. This will greatly reduces kernel hardlock up issues.

This is short term fix. Long term solution is to convert all synchronous messages to asynhrounous one.

- ipmi/power: Fix system reboot issue

  Kernel makes reboot/shudown OPAL call for reboot/shutdown. Once kernel gets response from OPAL it runs opal_poll_events() until firmware handles the request.

  On BMC based system, OPAL makes IPMI call (IPMI_CHASSIS_CONTROL) to initiate system reboot/shutdown. At present OPAL queues IPMI messages and return SUCESS to Host. If BMC is not ready to accept command (like BMC reboot), then these message will fail. We have to manually reboot/shutdown the system using BMC interface.

  This patch adds logic to validate message return value. If message failed, then it will resend the message. At some stage BMC will be ready to accept message and handles IPMI message.

- firmware-versions: Add test case for parsing VERSION

  Also make it possible to use with afl-lop/afl-fuzz just to help make *sure* we're all good.

  Additionally, if we hit a entry in VERSION that is larger than our buffer size, we skip over it gracefully rather than overwriting the stack. This is only a problem if VERSION isn't trusted, which as of 4b8cc05a94513816d43fb8bd6178896b430af08f it is verified as part of Secure Boot.

- core/fast-reboot: improve NMI handling during fast reset

  Improve sreset and MCE handling in fast reboot. Switch the HILE bit off before copying OPAL's exception vectors, so NMIs can be handled properly. Also disable MSR[ME] while the vectors are being overwritten

- core/cpu: HID update race

  If the per-core HID register is updated concurrently by multiple threads, updates can get lost. This has been observed during fast reboot where the HILE bit does not get cleared on all cores, which can cause machine check exception interrupts to crash.

  Fix this by only updating HID on thread0.

- SLW: Print verbose info on errors only

  Change print level from debug to warning for reporting bad EC_PPM_SPECIAL_WKUP_* scom values. To reduce cluttering in the log print only on error.

IBM FSP based platforms

- platforms/firenze: Rework I2C controller fixups
- platforms/zz: Re-enable LXVPD slot information parsing

  From memory this was disabled in the distant past since we were waiting for an updates to the LXPVD format. It looks like that never happened so re-enable it for the ZZ platform so that we can get PCI slot location codes on ZZ.

HIOMAP

- astbmc: Try IPMI HIOMAP for P8

  The HIOMAP protocol was developed after the release of P8 in preparation for P9. As a consequence P9 always uses it, but it has rarely been enabled for P8. P8DTU has recently added IPMI HIOMAP support to its BMC firmware, so enable its use in skiboot with P8 machines. Doing so requires some rework to ensure fallback works correctly as in the past the fallback was to mbox, which will only work for P9.

- libflash/ipmi-hiomap: Enforce message size for empty response

  The protocol defines the response to the associated messages as empty except for the command ID and sequence fields. If the BMC is returning extra data consider the message malformed.

- libflash/ipmi-hiomap: Remove unused close handling

  Issuing a HIOMAP_C_CLOSE is not required by the protocol specification, rather a close can be implicit in a subsequent CREATE_{READ,WRITE}_WINDOW request. The implicit close provides an opportunity to reduce LPC traffic and the implementation takes up that optimisation, so remove the case from the IPMI callback handler.

- libflash/ipmi-hiomap: Overhaul event handling

  Reworking the event handling was inspired by a bug report by Vasant where the host would get wedged on multiple flash access attempts in the face of a persistent error state on the BMC-side. The cause of this bug was the early-exit based on ctx->update, which erronously assumed that all events had been completely handled in prior calls to ipmi_hiomap_handle_events(). This is not true if e.g. HIOMAP_E_DAEMON_READY is clear in the prior calls.

  Regardless, there were other correctness and efficiency problems with the handling strategy:

  – Ack-able event state was not restored in the face of errors in the process of re-establishing protocol state
  – It forced needless window restoration with respect to the context in which ipmi_hiomap_handle_events() was called.
  – Tests for HIOMAP_E_DAEMON_READY and HIOMAP_E_FLASH_LOST were redundant with the overhauled error handling introduced in the previous patch

  Fix all of the above issues and add comments to explain the event handling flow.

- libflash/ipmi-hiomap: Overhaul error handling

  The aim is to improve the robustness with respect to absence of the BMC-side daemon. The current error handling roughly mirrors what was done for the mailbox implementation, but there's room for improvement.

  Errors are split into two classes, those that affect the transport state and those that affect the window validity. From here, we push the transport state error checks right to the bottom of the stack, to ensure the link is known to be in a good state before any message is sent. Window validity tests remain as they were in the hiomap_window_move() and ipmi_hiomap_read() functions. Validity tests are not necessary in the write and erase paths as we will receive an error response from the BMC when performing a dirty or flush on an invalid window.

  Recovery also remains as it was, done on entry to the blocklevel callbacks. If an error state is encountered in the middle of an operation no attempt is made to recover it on the spot, instead the error is returned up the stack and the caller can choose how it wishes to respond.

- libflash/ipmi-hiomap: Fix leak of msg in callback

POWER8

- hw/phb3/naples: Disable D-states

Putting "Mellanox Technologies MT27700 Family [ConnectX-4] [15b3:1013]" (more precisely, the second of 2 its PCI functions, no matter in what order) into the D3 state causes EEH with the "PCT timeout" error. This has been noticed on garrison machines only and firestones do not seem to have this issue.

This disables D-states changing for devices on root buses on Naples by installing a config space access filter (copied from PHB4).

- cpufeatures: Always advertise POWER8NVL as DD2

  Despite the major version of PVR being 1 (0x004c0100) for POWER8NVL, these chips are functionally equalent to P8/P8E DD2 levels.

  This advertises POWER8NVL as DD2. As the result, skiboot adds ibm,powerpc-cpu-features/processor-control-facility for such CPUs and the linux kernel can use hypervisor doorbell messages to wake secondary threads; otherwise "KVM: CPU %d seems to be stuck" would appear because of missing LPCR_PECEDH.

p8dtu Platform

- p8dtu: Configure BMC graphics

  We can no-longer read the values from the BMC in the way we have in the past. Values were provided by Eric Chen of SMC.

- p8dtu: Enable HIOMAP support

Vesnin Platform

- platforms/vesnin: Disable PCIe port bifurcation

  PCIe ports connected to CPU1 and CPU3 now work as x16 instead of x8x8.

- Fix hang in pnv_platform_error_reboot path due to TOD failure.

  On TOD failure, with TB stuck, when linux heads down to pnv_platform_error_reboot() path due to unrecoverable hmi event, the panic cpu gets stuck in OPAL inside ipmi_queue_msg_sync(). At this time, rest all other cpus are in smp_handle_nmi_ipi() waiting for panic cpu to proceed. But with panic cpu stuck inside OPAL, linux never recovers/reboot.

```
p0 c1 t0
NIA : 0x000000003001dd3c <.time_wait+0x64>
CFAR : 0x000000003001dce4 <.time_wait+0xc>
MSR : 0x9000000002803002
LR : 0x000000003002ecf8 <.ipmi_queue_msg_sync+0xec>

STACK: SP NIA
0x0000000031c236e0 0x0000000031c23760 (big-endian)
0x0000000031c23760 0x000000003002ecf8 <.ipmi_queue_msg_sync+0xec>
0x0000000031c237f0 0x00000000300aa5f8 <.hiomap_queue_msg_sync+0x7c>
0x0000000031c23880 0x00000000300aaadc <.hiomap_window_move+0x150>
0x0000000031c23950 0x00000000300ab1d8 <.ipmi_hiomap_write+0xcc>
0x0000000031c23a90 0x00000000300a7b18 <.blocklevel_raw_write+0xbc>
0x0000000031c23b30 0x00000000300a7c34 <.blocklevel_write+0xfc>
0x0000000031c23bf0 0x0000000030030be0 <.flash_nvram_write+0xd4>
0x0000000031c23c90 0x000000003002c128 <.opal_write_nvram+0xd0>
0x0000000031c23d20 0x00000000300051e4 <opal_entry+0x134>
0xc000001fea6e7870 0xc0000000000a9060 <opal_nvram_write+0x80>
0xc000001fea6e78c0 0xc000000000030b84 <nvram_write_os_partition+0x94>
```

(continues on next page)

(continued from previous page)

```
0xc000001fea6e7960 0xc0000000000310b0 <nvram_pstore_write+0xb0>
0xc000001fea6e7990 0xc0000000004792d4 <pstore_dump+0x1d4>
0xc000001fea6e7ad0 0xc00000000018a570 <kmsg_dump+0x140>
0xc000001fea6e7b40 0xc000000000028e5c <panic_flush_kmsg_end+0x2c>
0xc000001fea6e7b60 0xc0000000000a7168 <pnv_platform_error_reboot+0x68>
0xc000001fea6e7bd0 0xc0000000000ac9b8 <hmi_event_handler+0x1d8>
0xc000001fea6e7c80 0xc00000000012d6c8 <process_one_work+0x1b8>
0xc000001fea6e7d20 0xc00000000012da28 <worker_thread+0x88>
0xc000001fea6e7db0 0xc0000000001366f4 <kthread+0x164>
0xc000001fea6e7e20 0xc00000000000b65c <ret_from_kernel_thread+0x5c>
```

This is because, there is a while loop towards the end of ipmi_queue_msg_sync() which keeps looping until "sync_msg" does not match with "msg". It loops over time_wait_ms() until exit condition is met. In normal scenario time_wait_ms() calls run pollers so that ipmi backend gets a chance to check ipmi response and set sync_msg to NULL.

```
while (sync_msg == msg)
        time_wait_ms(10);
```

But in the event when TB is in failed state time_wait_ms()->time_wait_poll() returns immediately without calling pollers and hence we end up looping forever. This patch fixes this hang by calling opal_run_pollers() in TB failed state as well.

POWER9

- Retry link training at PCIe GEN1 if presence detected but training repeatedly failed

  Certain older PCIe 1.0 devices will not train unless the training process starts at GEN1 speeds. As a last resort when a device will not train, fall back to GEN1 speed for the last training attempt.

  This is verified to fix devices based on the Conexant CX23888 on the Talos II platform.

- hw/phb4: Drop FRESET_DEASSERT_DELAY state

  The delay between the ASSERT_DELAY and DEASSERT_DELAY states is set to one timebase tick. This state seems to have been a hold over from PHB3 where it was used to add a 1s delay between de-asserting PERST and polling the link for the CAPI FPGA. There's no requirement for that here since the link polling on PHB4 is a bit smarter so we should be fine.

- hw/phb4: Factor out PERST control

  Some time ago Mikey added some code work around a bug we found where a certain RAID card wouldn't come back again after a fast-reboot. The workaround is setting the Link Disable bit before asserting PERST and clear it after de-asserting PERST.

  Currently we do this in the FRESET path, but not in the CRESET path. This patch moves the PERST control into its own function to reduce duplication and to the workaround is applied in all circumstances.

- hw/phb4: Remove FRESET presence check

  When we do an freset the first step is to check if a card is present in the slot. However, this only occurs when we enter phb4_freset() with the slot state set to SLOT_NORMAL. This occurs in:

  a) The creset path, and

  b) When the OS manually requests an FRESET via an OPAL call.

(a) is problematic because in the boot path the generic code will put the slot into FRESET_START manually before calling into phb4_freset(). This can result in a situation where a device is detected on boot, but not after a CRESET.

I've noticed this occurring on systems where the PHB's slot presence detect signal is not wired to an adapter. In this situation we can rely on the in-band presence mechanism, but the presence check will make us exit before that has a chance to work.

Additionally, if we enter from the CRESET path this early exit leaves the slot's PERST signal being left asserted. This isn't currently an issue, but if we want to support hotplug of devices into the root port it will be.

- hw/phb4: Skip FRESET PERST when coming from CRESET

  PERST is asserted at the beginning of the CRESET process to prevent the downstream device from interacting with the host while the PHB logic is being reset and re-initialised. There is at least a 100ms wait during the CRESET processing so it's not necessary to wait this time again in the FRESET handler.

  This patch extends the delay after re-setting the PHB logic to extend to the 250ms PERST wait period that we typically use and sets the skip_perst flag so that we don't wait this time again in the FRESET handler.

- hw/phb4: Look for the hub-id from in the PBCQ node

  The hub-id is stored in the PBCQ node rather than the stack node so we never add it to the PHB node. This breaks the lxvpd slot lookup code since the hub-id is encoded in the VPD record that we need to find the slot information.

- hdata/iohub: Look for IOVPD on P9

  P8 and P9 use the same IO VPD setup, so we need to load the IOHUB VPD on P9 systems too.

CAPI2

- capp/phb4: Prevent HMI from getting triggered when disabling CAPP

  While disabling CAPP an HMI gets triggered as soon as ETU is put in reset mode. This is caused as before we can disabled CAPP, it detects PHB link going down and triggers an HMI requesting Opal to perform CAPP recovery. This has an un-intended side effect of spamming the Opal logs with malfunction alert messages and may also confuse the user.

  To prevent this we mask the CAPP FIR error 'PHB Link Down' Bit(31) when we are disabling CAPP just before we put ETU in reset in phb4_creset(). Also now since bringing down the PHB link now wont trigger an HMI and CAPP recovery, hence we manually set the PHB4_CAPP_RECOVERY flag on the phb to force recovery during creset.

- phb4/capp: Implement sequence to disable CAPP and enable fast-reset

  We implement h/w sequence to disable CAPP in disable_capi_mode() and with it also enable fast-reset for CAPI mode in phb4_set_capi_mode().

  Sequence to disable CAPP is executed in three phases. The first two phase is implemented in disable_capi_mode() where we reset the CAPP registers followed by PEC registers to their init values. The final third final phase is to reset the PHB CAPI Compare/Mask Register and is done in phb4_init_ioda3(). The reason to move the PHB reset to phb4_init_ioda3() is because by the time Opal PCI reset state machine reaches this function the PHB is already un-fenced and its configuration registers accessible via mmio.

- capp/phb4: Force CAPP to PCIe mode during kernel shutdown

  This patch introduces a new opal syncer for PHB4 named phb4_host_sync_reset(). We register this opal syncer when CAPP is activated successfully in phb4_set_capi_mode() so that it will be called at kernel shutdown during fast-reset.

During kernel shutdown the function will then repeatedly call phb->ops->set_capi_mode() to switch switch CAPP to PCIe mode. In case set_capi_mode() indicates its OPAL_BUSY, which indicates that CAPP is still transitioning to new state; it calls slot->ops.run_sm() to ensure that Opal slot reset state machine makes forward progress.

Witherspoon Platform

- platforms/witherspoon: Make PCIe shared slot error message more informative

  If we're missing chips for some reason, we print a warning when configuring the PCIe shared slot.

  The warning doesn't really make it clear what "shared slot" is, and if it's printed, it'll come right after a bunch of messages about NPU setup, so let's clarify the message to explicitly mention PCI.

- witherspoon: Add nvlink2 interconnect information

  See *New Features* for details.

Zaius Platform

- zaius: Add BMC description

  Frederic reported that Zaius was failing with a NULL dereference when trying to initialise IPMI HIOMAP. It turns out that the BMC wasn't described at all, so add a description.

p9dsu platform

- p9dsu: Fix p9dsu default variant

  Add the default when no riser_id is returned from the ipmi query.

  Allow a little more time for BMC reply and cleanup some label strings.

PCIe

See *POWER9* for POWER9 specific PCIe changes.

- core/pcie-slot: Don't bail early in the power on case

  Exiting early in the power off case makes sense since we can't disable slot power (or assert PERST) for suprise hotplug slots. However, we should not exit early in the power-on case since it's possible slot power may have been disabled (or just not enabled at boot time).

- firenze-pci: Always init slot info from LXVPD

  We can slot information from the LXVPD without having power control information about that slot. This patch changes the init path so that we always override the add_properties() call rather than only when we have power control information about the slot.

- fsp/lxvpd: Print more LXVPD slot information

  Useful to know since it changes the behaviour of the slot core.

- core/pcie-slot: Set power state from the PWRCTL flag

  For some reason we look at the power control indicator and use that to determine if the slot is "off" rather than the power control flag that is used to power down the slot.

While we're here change the default behaviour so that the slot is assumed to be powered on if there's no slot capability, or if there's no power control available.

- core/pci: Increase the max slot string size

  The maximum string length for the slot label / device location code in the PCI summary is currently 32 characters. This results in some IBM location codes being truncated due to their length, e.g.

```
PHB#0001:02:11.0 [SWDN]  SLOT=C11  x8
PHB#0001:13:00.0 [EP  ] *snip* LOC_CODE=U78D3.ND1.WZS004A-P1-C
PHB#0001:13:00.1 [EP  ] *snip* LOC_CODE=U78D3.ND1.WZS004A-P1-C
PHB#0001:13:00.2 [EP  ] *snip* LOC_CODE=U78D3.ND1.WZS004A-P1-C
PHB#0001:13:00.3 [EP  ] *snip* LOC_CODE=U78D3.ND1.WZS004A-P1-C
```

  Which obscure the actual location of the card, and it looks bad. This patch increases the maximum length of the label string to 80 characters since that's the maximum length for a location code.

OpenCAPI

- npu2/hw-procedures: Fix parallel zcal for opencapi

  For opencapi, we currently do impedance calibration when initializing the PHY for the device, which could run in parallel if we have multiple opencapi devices. But if 2 devices are on the same obus, the 2 calibration sequences could overlap, which likely yields bad results and is useless anyway since it only needs to be done once per obus.

  This patch splits the opencapi PHY reset in 2 parts:

  – a 'init' part called serially at boot. That's when zcal is done. If we have 2 devices on the same socket, the zcal won't be redone, since we're called serially and we'll see it has already be done for the obus

  – a 'reset' part called during fundamental reset as a prereq for link training. It does the PHY setup for a set of lanes and the dccal.

  The PHY team confirmed there's no dependency between zcal and the other reset steps and it can be moved earlier.

- npu2-hw-procedures: Fix zcal in mixed opencapi and nvlink mode

  The zcal procedure needs to be run once per obus. We keep track of which obus is already calibrated in an array indexed by the obus number. However, the obus number is inferred from the brick index, which works well for nvlink but not for opencapi.

  Create an obus_index() function, which, from a device, returns the correct obus index, irrespective of the device type.

- npu2-opencapi: Fix adapter reset when using 2 adapters

  If two opencapi adapters are on the same obus, we may try to train the two links in parallel at boot time, when all the PCI links are being trained. Both links use the same i2c controller to handle the reset signal, so some care is needed to make sure resetting one doesn't interfere with the reset of the other. We need to keep track of the current state of the i2c controller (and use locking).

  This went mostly unnoticed as you need to have 2 opencapi cards on the same socket and links tended to train anyway because of the retries.

- npu2-opencapi: Extend delay after releasing reset on adapter

  Give more time to the FPGA to process the reset signal. The previous delay, 5ms, is too short for newer adapters with bigger FPGAs. Extend it to 250ms. Ultimately, that delay will likely end up being added to the opencapi specification, but we are not there yet.

- npu2-opencapi: ODL should be in reset when enabled

  We haven't hit any problem so far, but from the ODL designer, the ODL should be in reset when it is enabled.

  The ODL remains in reset until we start a fundamental reset to initiate link training. We still assert and deassert the ODL reset signal as part of the normal procedure just before training the link. Asserting is therefore useless at boot, since the ODL is already in reset, but we keep it as it's only a scom write and it's needed when we reset/retrain from the OS.

- npu2-opencapi: Keep ODL and adapter in reset at the same time

  Split the function to assert and deassert the reset signal on the ODL, so that we can keep the ODL in reset while we reset the adapter, therefore having a window where both sides are in reset.

  It is actually not required with our current DLx at boot time, but I need to split the ODL reset function for the following patch and it will become useful/required later when we introduce resetting an opencapi link from the OS.

- npu2-opencapi: Setup perf counters to detect CRC errors

  It's possible to set up performance counters for the PLL to detect various conditions for the links in nvlink or opencapi mode. Since those counters are currently unused, let's configure them when an obus is in opencapi mode to detect CRC errors on the link. Each link has two counters: - CRC error detected by the host - CRC error detected by the DLx (NAK received by the host)

  We also dump the counters shortly after the link trains, but they can be read multiple times through cronus, pdbg or linux. The counters are configured to be reset after each read.

NVLINK2

- npu2: Allow ATSD for LPAR other than 0

  Each XTS MMIO ATSD# register is accompanied by another register - XTS MMIO ATSD0 LPARID# - which controls LPID filtering for ATSD transactions.

  When a host system passes a GPU through to a guest, we need to enable some ATSD for an LPAR. At the moment the host assigns one ATSD to a NVLink bridge and this maps it to an LPAR when GPU is assigned to the LPAR. The link number is used for an ATSD index.

  ATSD6&7 stay mapped to the host (LPAR=0) all the time which seems to be acceptable price for the simplicity.

- npu2: Add XTS_BDF_MAP wildcard refcount

  Currently PID wildcard is programmed into the NPU once and never cleared up. This works for the bare metal as MSR does not change while the host OS is running.

  However with the device virtualization, we need to keep track of wildcard entries use and clear them up before switching a GPU from a host to a guest or vice versa.

  This adds refcount to a NPU2, one counter per wildcard entry. The index is a short lparid (4 bits long) which is allocated in opal_npu_map_lpar() and should be smaller than NPU2_XTS_BDF_MAP_SIZE (defined as 16).

Debugging and simulation

- external/mambo: Error out if kernel is too large

  If you're trying to boot a gigantic kernel in mambo (which you can reproduce by building a kernel with CONFIG_MODULES=n) you'll get misleading errors like:

```
WARNING: 0: (0): [0:0]: Invalid/unsupported instr 0x00000000[INVALID]
WARNING: 0: (0):  PC(EA): 0x0000000030000010 PC(RA):0x0000000030000010 MSR:␣
↪0x9000000000000000 LR: 0x0000000000000000
WARNING: 0: (0):  numInstructions = 0
WARNING: 1: (1): [0:0]: Invalid/unsupported instr 0x00000000[INVALID]
WARNING: 1: (1):  PC(EA): 0x0000000000000E40 PC(RA):0x0000000000000E40 MSR:␣
↪0x9000000000000000 LR: 0x0000000000000000
WARNING: 1: (1):  numInstructions = 1
WARNING: 1: (1): Interrupt to 0x0000000000000E40 from 0x0000000000000E40
INFO: 1: (2): ** Execution stopped: Continuous Interrupt, Instruction caused␣
↪exception,  **
```

So add an error to skiboot.tcl to warn the user before this happens. Making PAYLOAD_ADDR further back is one way to do this but if there's a less gross way to generally work around this very niche problem, I can suggest that instead.

- external/mambo: Populate kernel-base-address in the DT

  skiboot.tcl defines PAYLOAD_ADDR as 0x20000000, which is the default in skiboot. This is also the default in skiboot unless kernel-base-address is set in the device tree.

  If you change PAYLOAD_ADDR to something else for mambo, skiboot won't see it because it doesn't set that DT property, so fix it so that it does.

- external/mambo: allow CPU targeting for most debug utils

  Debug util functions target CPU 0:0:0 by default Some can be overidden explicitly per invocation, and others can't at all. Even for those that can be overidden, it is a pain to type them out when you're debugging a particular thread.

  Provide a new 'target' function that allows the default CPU target to be changed. Wire that up that default to all other utils. Provide a new 'S' step command which only steps the target CPU.

- qemu: bt device isn't always hanging off /

  Just use the normal for_each_compatible instead.

  Otherwise in the qemu model as executed by op-test, we wouldn't go down the astbmc_init() path, thus not having flash.

- devicetree: Add p9-simics.dts

  Add a p9-based devicetree that's suitable for use with Simics.

- devicetree: Move power9-phb4.dts

  Clean up the formatting of power9-phb4.dts and move it to external/devicetree/p9.dts. This sets us up to include it as the basis for other trees.

- devicetree: Add nx node to power9-phb4.dts

  A (non-qemu) p9 without an nx node will assert in p9_darn_init():

```
dt_for_each_compatible(dt_root, nx, "ibm,power9-nx")
        break;
if (!nx) {
        if (!dt_node_is_compatible(dt_root, "qemu,powernv"))
              assert(nx);
        return;
}
```

  Since NX is this essential, add it to the device tree.

- devicetree: Fix typo in power9-phb4.dts

  Change "impi" to "ipmi".

- devicetree: Fix syntax error in power9-phb4.dts

  Remove the extra space causing this:

  ```
  Error: power9-phb4.dts:156.15-16 syntax error
  FATAL ERROR: Unable to parse input tree
  ```

- core/init: enable machine check on secondaries

  Secondary CPUs currently run with MSR[ME]=0 during boot, whih means if they take a machine check, the system will checkstop.

  Enable ME where possible and allow them to print registers.

Utilities

- pflash: Don't try update RO ToC

  In the future it's likely the ToC will be marked as read-only. Don't error out by assuming its writable.

- pflash: Support encoding/decoding ECC'd partitions

  With the new –ecc option, pflash can add/remove ECC when reading/writing flash partitions protected by ECC.

  This is *not* flawless with current PNORs out in the wild though, as they do not typically fill the whole partition with valid ECC data, so you have to know how big the valid ECC'd data is and specify the size manually. Note that for some partitions this is pratically impossible without knowing the details of the content of the partition.

  A future patch is likely to introduce an option to "stop reading data when ECC starts failing and assume everything is okay rather than error out" to support reading the "valid" data from existing PNOR images.

5.1.136 skiboot-6.3-rc2

skiboot v6.3-rc2 was released on Thursday April 11th 2019. It is the second release candidate of skiboot 6.3, which will become the new stable release of skiboot following the 6.2 release, first released December 14th 2018.

Skiboot 6.3 will mark the basis for op-build v2.3. I expect to tag the final skiboot 6.3 in the next week.

skiboot v6.3-rc2 contains all bug fixes as of *skiboot-6.0.19*, and *skiboot-6.2.3* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

Over *skiboot-6.3-rc1*, we have the following changes:

- libflash/ipmi-hiomap: Fix blocks count issue

  We convert data size to block count and pass block count to BMC. If data size is not block aligned then we endup sending block count less than actual data. BMC will write partial data to flash memory.

  Sample log

  ```
  [  594.388458416,7] HIOMAP: Marked flash dirty at 0x42010 for 8
  [  594.398756487,7] HIOMAP: Flushed writes
  [  594.409596439,7] HIOMAP: Marked flash dirty at 0x42018 for 3970
  [  594.419897507,7] HIOMAP: Flushed writes
  ```

  In this case HIOMAP sent data with block count=0 and hence BMC didn't flush data to flash.

- opal/hmi: Never trust a cow!

  With opencapi, it's fairly common to trigger HMIs during AFU development on the FPGA, by not replying in time to an NPU command, for example. So shift the blame reported by that cow to avoid crowding my mailbox.

- hw/npu2: Dump (more) npu2 registers on link error and HMIs

  We were already logging some NPU registers during an HMI. This patch cleans up a bit how it is done and separates what is global from what is specific to nvlink or opencapi.

  Since we can now receive an error interrupt when an opencapi link goes down unexpectedly, we also dump the NPU state but we limit it to the registers of the brick which hit the error.

  The list of registers to dump was worked out with the hw team to allow for proper debugging. For each register, we print the name as found in the NPU workbook, the scom address and the register value.

- hw/npu2: Report errors to the OS if an OpenCAPI brick is fenced

  Now that the NPU may report interrupts due to the link going down unexpectedly, report those errors to the OS when queried by the 'next_error' PHB callback.

  The hardware doesn't support recovery of the link when it goes down unexpectedly. So we report the PHB as dead, so that the OS can log the proper message, notify the drivers and take the devices down.

- hw/npu2: Fix OpenCAPI PE assignment

  When we support mixing NVLink and OpenCAPI devices on the same NPU, we're going to have to share the same range of 16 PE numbers between NVLink and OpenCAPI PHBs.

  For OpenCAPI devices, PE assignment is only significant for determining which System Interrupt Log register is used for a particular brick - unlike NVLink, it doesn't play any role in determining how links are fenced.

  Split the PE range into a lower half which is used for NVLink, and an upper half that is used for OpenCAPI, with a fixed PE number assigned per brick.

  As the PE assignment for OpenCAPI devices is fixed, set the PE once during device init and then ignore calls to the set_pe() operation.

- opal-api: Reserve 2 OPAL API calls for future OpenCAPI LPC use

  OpenCAPI Lowest Point of Coherency (LPC) memory is going to require some extra OPAL calls to set up NPU BARs. These calls will most likely be called OPAL_NPU_LPC_ALLOC and OPAL_NPU_LPC_RELEASE, we're not quite ready to upstream that code yet though.

- cpufeatures: Add tm-suspend-hypervisor-assist and tm-suspend-xer-so-bug node

  tm-suspend-hypervisor-assist for P9 >=DD2.2 And a tm-suspend-xer-so-bug node for P9 DD2.2 only.

  I also treat P9P as P9 DD2.3 and add a unit test for the cpufeatures infrastructure.

  Fixes: https://github.com/open-power/skiboot/issues/233

5.1.137 skiboot-6.3-rc3

skiboot v6.3-rc3 was released on Thursday May 2nd 2019. It is the third release candidate of skiboot 6.3, which will become the new stable release of skiboot following the 6.2 release, first released December 14th 2018.

Skiboot 6.3 will mark the basis for op-build v2.3. I expect to tag the final skiboot 6.3 in the next week (I also predicted this last time, so take my predictions with a large amount of sodium).

skiboot v6.3-rc3 contains all bug fixes as of *skiboot-6.0.19*, and *skiboot-6.2.3* (the currently maintained stable releases).

For how the skiboot stable releases work, see *Skiboot stable tree rules and releases* for details.

Over *skiboot-6.3-rc2*, we have the following changes:

- Expose PNOR Flash partitions to host MTD driver via devicetree

  This makes it possible for the host to directly address each partition without requiring each application to directly parse the FFS headers. This has been in use for some time already to allow BOOTKERNFW partition updates from the host.

  All partitions except BOOTKERNFW are marked readonly.

  The BOOTKERNFW partition is currently exclusively used by the TalosII platform

- Write boot progress to LPC port 80h

  This is an adaptation of what we currently do for op_display() on FSP machines, inventing an encoding for what we can write into the single byte at LPC port 80h.

  Port 80h is often used on x86 systems to indicate boot progress/status and dates back a decent amount of time. Since a byte isn't exactly very expressive for everything that can go on (and wrong) during boot, it's all about compromise.

  Some systems (such as Zaius/Barreleye G2) have a physical dual 7 segment display that display these codes. So far, this has only been driven by hostboot (see hostboot commit 90ec2e65314c).

- Write boot progress to LPC ports 81 and 82

  There's a thought to write more extensive boot progress codes to LPC ports 81 and 82 to supplement/replace any reliance on port 80.

  We want to still emit port 80 for platforms like Zaius and Barreleye that have the physical display. Ports 81 and 82 can be monitored by a BMC though.

- Copy and convert Romulus descriptors to Talos

  Talos II has some hardware differences from Romulus, therefore we cannot guarantee Talos II == Romulus in skiboot. Copy and slightly modify the Romulus files for Talos II.

- npu2: Disable Probe-to-Invalid-Return-Modified-or-Owned snarfing by default

  V100 GPUs are known to violate NVLink2 protocol in some cases (one is when memory was accessed by the CPU and they by GPU using so called block linear mapping) and issue double probes to NPU which can cope with this problem only if CONFIG_ENABLE_SNARF_CPM ("disable/enable Probe.I.MO snarfing a cp_m") is not set in the CQ_SM Misc Config register #0. If the bit is set (which is the case today), NPU issues the machine check stop.

  The snarfing feature is designed to detect 2 probes in flight and combine them into one.

  This adds a new "opal-npu2-snarf-cpm" nvram variable which controls CONFIG_ENABLE_SNARF_CPM for all NVLinks to prevent the machine check stop from happening.

  This disables snarfing by default as otherwise a broken GPU driver can crash the entire box even when a GPU is passed through to a guest. This provides a dial to allow regression tests (might be useful for a bare metal). To enable snarfing, the user needs to run:

  ```
  sudo nvram -p ibm,skiboot --update-config opal-npu2-snarf-cpm=enable
  ```

  and reboot the host system.

- hw/npu2: Show name of opencapi error interrupts

- core/pci: Use PHB io-base-location by default for PHB slots

  On witherspoon only the GPU slots and the three pluggable PCI slots (SLOT0, 1, 2) have platform defined slot names. For builtin devices such as the SATA controller or the PLX switch that fans out to the GPU slots we have no location codes which some people consider an issue.

This patch address the problem by making the ibm,slot-location-code for the root port device default to the ibm,io-base-location-code which is typically the location code for the system itself.

e.g.

```
pciex@600c3c0100000/ibm,loc-code
                 "UOPWR.0000000-Node0-Proc0"

pciex@600c3c0100000/pci@0/ibm,loc-code
                 "UOPWR.0000000-Node0-Proc0"

pciex@600c3c0100000/pci@0/usb-xhci@0/ibm,loc-code
                 "UOPWR.0000000-Node0"
```

The PHB node, and the root complex nodes have a loc code of the processor they are attached to, while the usb-xhci device under the root port has a location code of the system itself.

- hw/phb4: Read ibm,loc-code from PBCQ node

  On P9 the PBCQs are subdivided by stacks which implement the PCI Express logic. When phb4 was forked from phb3 most of the properties that were in the pbcq node moved into the stack node, but ibm,loc-code was not one of them. This patch fixes the phb4 init sequence to read the base location code from the PBCQ node (parent of the stack node) rather than the stack node itself.

- hw/xscom: add missing P9P chip name
- asm/head: balance branches to avoid link stack predictor mispredicts

  The Linux wrapper for OPAL call and return is arranged like this:

```
__opal_call:
    mflr   r0
    std    r0,PPC_STK_LROFF(r1)
    LOAD_REG_ADDR(r11, opal_return)
    mtlr   r11
    hrfid  -> OPAL

opal_return:
    ld     r0,PPC_STK_LROFF(r1)
    mtlr   r0
    blr
```

  When skiboot returns to Linux, it branches to LR (i.e., opal_return) with a blr. This unbalances the link stack predictor and will cause mispredicts back up the return stack.

- external/mambo: also invoke readline for the non-autorun case
- asm/head.S: set POWER9 radix HID bit at entry

  When running in virtual memory mode, the radix MMU hid bit should not be changed, so set this in the initial boot SPR setup.

  As a side effect, fast reboot also has HID0:RADIX bit set by the shared spr init, so no need for an explicit call.

- opal-prd: Fix memory leak in is-fsp-system check
- opal-prd: Check malloc return value
- hw/phb4: Squash the IO bridge window

  The PCI-PCI bridge spec says that bridges that implement an IO window should hardcode the IO base and limit registers to zero. Unfortunately, these registers only define the upper bits of the IO window and the low bits are assumed to be 0 for the base and 1 for the limit address. As a result, setting both to zero can be mis-interpreted as a 4K IO window.

This patch fixes the problem the same way PHB3 does. It sets the IO base and limit values to 0xf000 and 0x1000 respectively which most software interprets as a disabled window.

lspci before patch:

```
0000:00:00.0 PCI bridge: IBM Device 04c1 (prog-if 00 [Normal decode])
        I/O behind bridge: 00000000-00000fff
```

lspci after patch:

```
0000:00:00.0 PCI bridge: IBM Device 04c1 (prog-if 00 [Normal decode])
        I/O behind bridge: None
```

- build: link with –orphan-handling=warn

  The linker can warn when the linker script does not explicitly place all sections. These orphan sections are placed according to heuristics, which may not always be desirable. Enable this warning.

- build: -fno-asynchronous-unwind-tables

  skiboot does not use unwind tables, this option saves about 100kB, mostly from .text.

- hw/xscom: Enable sw xstop by default on p9

  This was disabled at some point during bringup to make life easier for the lab folks trying to debug NVLink issues. This hack really should have never made it out into the wild though, so we now have the following situation occuring in the field:

  1) A bad happens
  2) The host kernel recieves an unrecoverable HMI and calls into OPAL to request a platform reboot.
  3) OPAL rejects the reboot attempt and returns to the kernel with OPAL_PARAMETER.
  4) Kernel panics and attempts to kexec into a kdump kernel.

  A side effect of the HMI seems to be CPUs becoming stuck which results in the initialisation of the kdump kernel taking a extremely long time (6+ hours). It's also been observed that after performing a dump the kdump kernel then crashes itself because OPAL has ended up in a bad state as a side effect of the HMI.

  All up, it's not very good so re-enable the software checkstop by default. If people still want to turn it off they can using the nvram override.

- opal/hmi: Initialize the hmi event with old value of TFMR.

  Do this before we fix TFAC errors. Otherwise the event at host console shows no thread error reported in TFMR register.

  Without this patch the console event show TFMR with no thread error: (DEC parity error TFMR[59] injection)

```
[   53.737572] Severe Hypervisor Maintenance interrupt [Recovered]
[   53.737596]  Error detail: Timer facility experienced an error
[   53.737611]  HMER: 0840000000000000
[   53.737621]  TFMR: 3212000870e04000
```

  After this patch it shows old TFMR value on host console:

```
[ 2302.267271] Severe Hypervisor Maintenance interrupt [Recovered]
[ 2302.267305]  Error detail: Timer facility experienced an error
```

(continues on next page)

(continued from previous page)

```
[ 2302.267320]  HMER: 0840000000000000
[ 2302.267330]  TFMR: 3212000870e14010
```

5.1.138 skiboot-6.3.1

skiboot 6.3.1 was released on Friday May 10th, 2019. It replaces *skiboot-6.3* as the current stable release in the 6.3.x series.

It is recommended that 6.3.1 be used instead of 6.3 version due to the bug fixes it contains.

Bug fixes included in this release are:

- platforms/astbmc: Check for SBE validation step

  On some POWER8 astbmc systems an update to the SBE requires pausing at runtime to ensure integrity of the SBE. If this is required the BMC will set a chassis boot option IPMI flag using the OEM parameter 0x62. If Skiboot sees this flag is set it waits until the SBE update is complete and the flag is cleared. Unfortunately the mystery operation that validates the SBE also leaves it in a bad state and unable to be used for timer operations. To workaround this the flag is checked as soon as possible (ie. when IPMI and the console are set up), and once complete the system is rebooted.

- ipmi: ensure forward progress on ipmi_queue_msg_sync()

  BT responses are handled using a timer doing the polling. To hope to get an answer to an IPMI synchronous message, the timer needs to run.

  We can't just check all timers though as there may be a timer that wants a lock that's held by a code path calling ipmi_queue_msg_sync(), and if we did enforce that as a requirement, it's a pretty subtle API that is asking to be broken.

  So, if we just run a poll function to crank anything that the IPMI backend needs, then we should be fine.

  This issue shows up very quickly under QEMU when loading the first flash resource with the IPMI HIOMAP backend.

- pci/iov: Remove skiboot VF tracking

  This feature was added a few years ago in response to a request to make the MaxPayloadSize (MPS) field of a Virtual Function match the MPS of the Physical Function that hosts it.

  The SR-IOV specification states the the MPS field of the VF is "ResvP". This indicates the VF will use whatever MPS is configured on the PF and that the field should be treated as a reserved field in the config space of the VF. In other words, a SR-IOV spec compliant VF should always return zero in the MPS field. Adding hacks in OPAL to make it non-zero is… misguided at best.

  Additionally, there is a bug in the way pci_device structures are handled by VFs that results in a crash on fast-reboot that occurs if VFs are enabled and then disabled prior to rebooting. This patch fixes the bug by removing the code entirely. This patch has no impact on SR-IOV support on the host operating system.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search