Firmware-Assisted Dump

July 2011

The goal of firmware-assisted dump is to enable the dump of a crashed system, and to do so from a fully-reset system, and to minimize the total elapsed time until the system is back in production use.

- Firmware-Assisted Dump (FADump) infrastructure is intended to replace the existing phyp assisted dump.
- Fadump uses the same firmware interfaces and memory reservation model as phyp assisted dump.
- Unlike phyp dump, FADump exports the memory dump through /proc/vmcore in the ELF format in the same way as kdump. This helps us reuse the kdump infrastructure for dump capture and filtering.
- Unlike phyp dump, userspace tool does not need to refer any sysfs interface while reading /proc/vmcore.
- Unlike phyp dump, FADump allows user to release all the memory reserved for dump, with a single operation of echo 1 > /sys/kernel/fadump_release_mem.
- Once enabled through kernel boot parameter, FADump can be started/stopped through /sys/kernel/fadump_registered interface (see sysfs files section below) and can be easily integrated with kdump service start/stop init scripts.

Comparing with kdump or other strategies, firmware-assisted dump offers several strong, practical advantages:

- Unlike kdump, the system has been reset, and loaded with a fresh copy of the kernel. In particular, PCI and I/O devices have been reinitialized and are in a clean, consistent state.
- Once the dump is copied out, the memory that held the dump is immediately available to the running kernel. And therefore, unlike kdump, FADump doesn't need a 2nd reboot to get back the system to the production configuration.

The above can only be accomplished by coordination with, and assistance from the Power firmware. The procedure is as follows:

- The first kernel registers the sections of memory with the Power firmware for dump preservation during OS initialization. These registered sections of memory are reserved by the first kernel during early boot.
- When system crashes, the Power firmware will copy the registered low memory regions (boot memory) from source to destination area. It will also save hardware PTE's.

  NOTE:

  The term 'boot memory' means size of the low memory chunk that is required for a kernel to boot successfully when booted with restricted memory. By default, the boot memory size will be the larger of 5% of system RAM or 256MB. Alternatively, user can also specify boot memory size through boot parameter 'crashkernel=' which will override the default calculated size. Use this option if default boot memory size is not sufficient for second kernel to boot successfully. For syntax of crashkernel= parameter, refer to Documentation/admin-guide/kdump/kdump.rst. If any offset is provided in crashkernel= parameter, it will be ignored as FADump uses a predefined offset to reserve memory for boot memory dump preservation in case of a crash.
- After the low memory (boot memory) area has been saved, the firmware will reset PCI and other hardware state. It will *not* clear the RAM. It will then launch the bootloader, as normal.
- The freshly booted kernel will notice that there is a new node (rtas/ibm,kernel-dump on pSeries or ibm,opal/dump/mpipl-boot on OPAL platform) in the device tree, indicating that there is crash data available from a previous boot. During the early boot OS will reserve rest of the memory above boot memory size effectively booting with restricted memory size. This will make sure that this kernel (also, referred to as second kernel or capture kernel) will not touch any of the dump memory area.
- User-space tools will read /proc/vmcore to obtain the contents of memory, which holds the previous crashed kernel dump in ELF format. The userspace tools may copy this info to disk, or network, nas, san, iscsi, etc. as desired.
- Once the userspace tool is done saving dump, it will echo '1' to /sys/kernel/fadump_release_mem to release the reserved memory back to general use, except the memory required for next firmware-assisted dump registration.

  e.g.:

```
# echo 1 > /sys/kernel/fadump_release_mem
```

Please note that the firmware-assisted dump feature is only available on POWER6 and above systems on pSeries (PowerVM) platform and POWER9 and above systems with OP940 or later firmware versions on PowerNV (OPAL) platform. Note that, OPAL firmware exports ibm,opal/dump node when FADump is supported on PowerNV platform.

On OPAL based machines, system first boots into an intermittent kernel (referred to as petitboot kernel) before booting into the capture kernel. This kernel would have minimal kernel and/or userspace support to process crash data. Such kernel needs to preserve previously crash'ed kernel's memory for the subsequent capture kernel boot to process this crash data. Kernel config option CONFIG_PRESERVE_FA_DUMP has to be enabled on such kernel to ensure that crash data is preserved to process later.

-- On OPAL based machines (PowerNV), if the kernel is build with

CONFIG_OPAL_CORE=y, OPAL memory at the time of crash is also exported as /sys/firmware/opal/mpipl/core file. This procfs file is helpful in debugging OPAL crashes with GDB. The kernel memory used for exporting this procfs file can be released by echo'ing '1' to /sys/firmware/opal/mpipl/release_core node.

e.g.

```
# echo 1 > /sys/firmware/opal/mpipl/release_core
```

Implementation details:

During boot, a check is made to see if firmware supports this feature on that particular machine. If it does, then we check to see if an active dump is waiting for us. If yes then everything but boot memory size of RAM is reserved during early boot (See Fig. 2). This area is released once we finish collecting the dump from user land scripts (e.g. kdump scripts) that are run. If there is dump data, then the /sys/kernel/fadump_release_mem file is created, and the reserved memory is held.

If there is no waiting dump data, then only the memory required to hold CPU state, HPTE region, boot memory dump, FADump header and elfcore header, is usually reserved at an offset greater than boot memory size (see Fig. 1). This area is *not* released: this region will be kept permanently reserved, so that it can act as a receptacle for a copy of the boot memory content in addition to CPU state and HPTE region, in the case a crash does occur.

Since this reserved memory area is used only after the system crash, there is no point in blocking this significant chunk of memory from production kernel. Hence, the implementation uses the Linux kernel's Contiguous Memory Allocator (CMA) for memory reservation if CMA is configured for kernel. With CMA reservation this memory will be available for applications to use it, while kernel is prevented from using it. With this FADump will still be able to capture all of the kernel memory and most of the user space memory except the user pages that were present in CMA region:

```
o Memory Reservation during first kernel

Low memory                                                  Top of memory
0    boot memory size   |<--- Reserved dump area --->|       |
|           |           |    Permanent Reservation   |       |
V           V           |                            |       V
+-----------+-----/ /---+---+----+-------+-----+-----+----+--+
|           |           |///|////|  DUMP | HDR | ELF |////|  |
+-----------+-----/ /---+---+----+-------+-----+-----+----+--+
      |                   ^    ^     ^      ^           ^
      |                   |    |     |      |           |
      \                  CPU  HPTE   /      |           |
       ------------------------------       |           |
    Boot memory content gets transferred    |           |
    to reserved area by firmware at the     |           |
    time of crash.                          |           |
                                      FADump Header     |
                                       (meta area)      |
                                                        |
                                                        |
                    Metadata: This area holds a metadata structure whose
                    address is registered with f/w and retrieved in the
                    second kernel after crash, on platforms that support
                    tags (OPAL). Having such structure with info needed
                    to process the crashdump eases dump capture process.

                Fig. 1


o Memory Reservation during second kernel after crash

Low memory                                                Top of memory
0      boot memory size                                       |
|           |<------------ Crash preserved area ------------>|
V           V           |<--- Reserved dump area --->|       |
+-----------+-----/ /---+---+----+-------+-----+-----+----+--+
|           |           |///|////|  DUMP | HDR | ELF |////|  |
+-----------+-----/ /---+---+----+-------+-----+-----+----+--+
      |                                        |
      V                                        V
 Used by second                           /proc/vmcore
 kernel to boot

      +---+
      |///| -> Regions (CPU, HPTE & Metadata) marked like this in the above
      +---+    figures are not always present. For example, OPAL platform
               does not have CPU & HPTE regions while Metadata region is
               not supported on pSeries currently.

                 Fig. 2
```

Currently the dump will be copied from /proc/vmcore to a new file upon user intervention. The dump data available through /proc/vmcore will be in ELF format. Hence the existing kdump infrastructure (kdump scripts) to save the dump works fine with minor modifications. KDump scripts on major Distro releases have already been modified to work seamlessly (no user intervention in saving the dump) when FADump is used, instead of KDump, as dump mechanism.

The tools to examine the dump will be same as the ones used for kdump.

How to enable firmware-assisted dump (FADump):

1. Set config option CONFIG_FA_DUMP=y and build kernel.
2. Boot into linux kernel with 'fadump=on' kernel cmdline option. By default, FADump reserved memory will be initialized as CMA area. Alternatively, user can boot linux kernel with 'fadump=nocma' to prevent FADump to use CMA.
3. Optionally, user can also set 'crashkernel=' kernel cmdline to specify size of the memory to reserve for boot memory dump preservation.

NOTE:

1. 'fadump_reserve_mem=' parameter has been deprecated. Instead use 'crashkernel=' to specify size of the memory to reserve for boot memory dump preservation.
2. If firmware-assisted dump fails to reserve memory then it will fallback to existing kdump mechanism if 'crashkernel=' option is set at kernel cmdline.
3. if user wants to capture all of user space memory and ok with reserved memory not available to production system, then 'fadump=nocma' kernel parameter can be used to fallback to old behaviour.

Sysfs/debugfs files:

Firmware-assisted dump feature uses sysfs file system to hold the control files and debugfs file to display memory reserved region.

Here is the list of files under kernel sysfs:

/sys/kernel/fadump_enabled

This is used to display the FADump status.

- 0 = FADump is disabled
- 1 = FADump is enabled

This interface can be used by kdump init scripts to identify if FADump is enabled in the kernel and act accordingly.

/sys/kernel/fadump_registered

This is used to display the FADump registration status as well as to control (start/stop) the FADump registration.

- 0 = FADump is not registered.
- 1 = FADump is registered and ready to handle system crash.

To register FADump echo 1 > /sys/kernel/fadump_registered and echo 0 > /sys/kernel/fadump_registered for un-register and stop the FADump. Once the FADump is un-registered, the system crash will not be handled and vmcore will not be captured. This interface can be easily integrated with kdump service start/stop.

/sys/kernel/fadump/mem_reserved

This is used to display the memory reserved by FADump for saving the crash dump.

/sys/kernel/fadump_release_mem

This file is available only when FADump is active during second kernel. This is used to release the reserved memory region that are held for saving crash dump. To release the reserved memory echo 1 to it:

```
echo 1  > /sys/kernel/fadump_release_mem
```

After echo 1, the content of the /sys/kernel/debug/powerpc/fadump_region file will change to reflect the new memory reservations.

The existing userspace tools (kdump infrastructure) can be easily enhanced to use this interface to release the memory reserved for dump and continue without 2nd reboot.

Note: /sys/kernel/fadump_release_opalcore sysfs has moved to

/sys/firmware/opal/mpipl/release_core

/sys/firmware/opal/mpipl/release_core

This file is available only on OPAL based machines when FADump is active during capture kernel. This is used to release the memory used by the kernel to export /sys/firmware/opal/mpipl/core file. To release this memory, echo '1' to it:

echo 1 > /sys/firmware/opal/mpipl/release_core

Note: The following FADump sysfs files are deprecated.

Deprecated	Alternative
/sys/kernel/fadump_enabled	/sys/kernel/fadump/enabled
/sys/kernel/fadump_registered	/sys/kernel/fadump/registered
/sys/kernel/fadump_release_mem	/sys/kernel/fadump/release_mem

Here is the list of files under powerpc debugfs: (Assuming debugfs is mounted on /sys/kernel/debug directory.)

/sys/kernel/debug/powerpc/fadump_region

This file shows the reserved memory regions if FADump is enabled otherwise this file is empty. The output format is:

```
<region>: [<start>-<end>] <reserved-size> bytes, Dumped: <dump-size>
```

and for kernel DUMP region is:

DUMP: Src: <src-addr>, Dest: <dest-addr>, Size: <size>, Dumped: # bytes

e.g. Contents when FADump is registered during first kernel:

```
# cat /sys/kernel/debug/powerpc/fadump_region
CPU : [0x0000006ffb0000-0x0000006fff001f] 0x40020 bytes, Dumped: 0x0
HPTE: [0x0000006fff0020-0x0000006fff101f] 0x1000 bytes, Dumped: 0x0
DUMP: [0x0000006fff1020-0x0000007fff101f] 0x10000000 bytes, Dumped: 0x0
```

Contents when FADump is active during second kernel:

```
# cat /sys/kernel/debug/powerpc/fadump_region
CPU : [0x0000006ffb0000-0x0000006fff001f] 0x40020 bytes, Dumped: 0x40020
HPTE: [0x0000006fff0020-0x0000006fff101f] 0x1000 bytes, Dumped: 0x1000
DUMP: [0x0000006fff1020-0x0000007fff101f] 0x10000000 bytes, Dumped: 0x10000000
    : [0x00000010000000-0x0000006ffaffff] 0x5ffb0000 bytes, Dumped: 0x5ffb0000
```

NOTE:

Please refer to Documentation/filesystems/debugfs.rst on how to mount the debugfs filesystem.

TODO:

- Need to come up with the better approach to find out more accurate boot memory size that is required for a kernel to boot successfully when booted with restricted memory.
- The FADump implementation introduces a FADump crash info structure in the scratch area before the ELF core header. The idea of introducing this structure is to pass some important crash info data to the second kernel which will help second kernel to populate ELF core header with correct data before it gets exported through /proc/vmcore. The current design implementation does not address a possibility of introducing additional fields (in future) to this structure without affecting compatibility. Need to come up with the better approach to address this.

  The possible approaches are:

  1. Introduce version field for version tracking, bump up the version whenever a new field is added to the structure in future. The version field can be used to find out what fields are valid for the current version of the structure. 2. Reserve the area of predefined size (say PAGE_SIZE) for this structure and have unused area as reserved (initialized to zero) for future field additions.

  The advantage of approach 1 over 2 is we don't need to reserve extra space.

Author: Mahesh Salgaonkar <mahesh@linux.vnet.ibm.com>

This document is based on the original documentation written for phyp

assisted dump by Linas Vepstas and Manish Ahuja.