The ftrace and perf cases on ACPI and Hibernation

August, 2017, Beijing

Joey Lee
SUSE Labs Taipei

Agenda

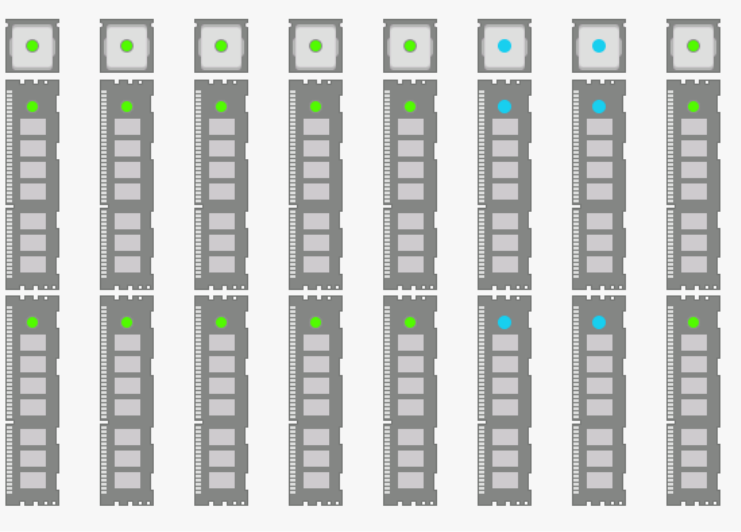
- Ftrace on ACPI hot-removing
- Soft lockups during S4 resume
- Q&A

Ftrace on ACPI hot-removing

bsc#1043764

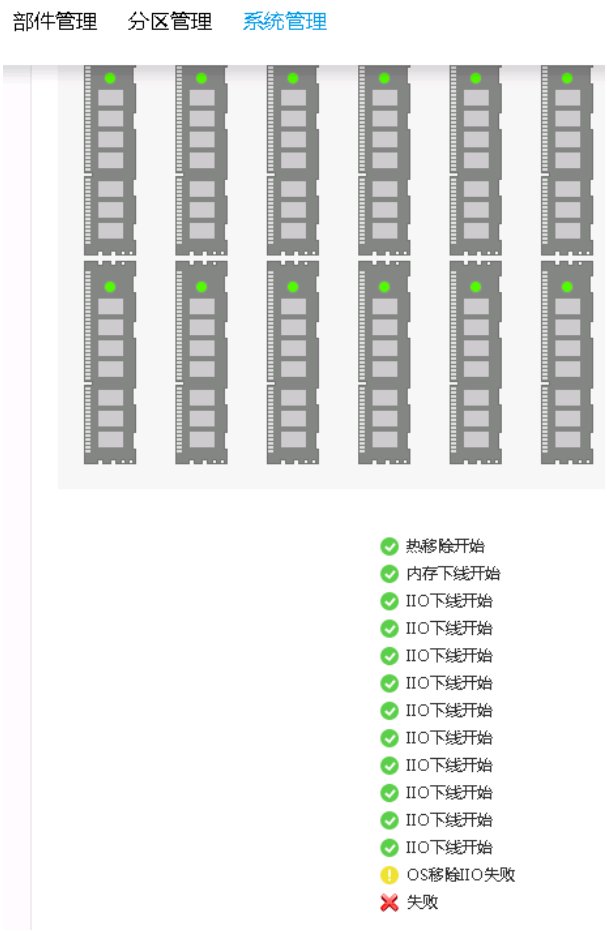
- Bug 1043764 - KunLun Server Hotplug: IO offline failed when offline cpu-pairs
- This is an issue found during stress test from Huawei KunLun Server Hotplug RAS initiative.
- IO offline failed when hot remove cpu-pair node5 and node6 during running oracle on KunLun server. The following is the error message. This issue does not happened everytime.

No loading - success case

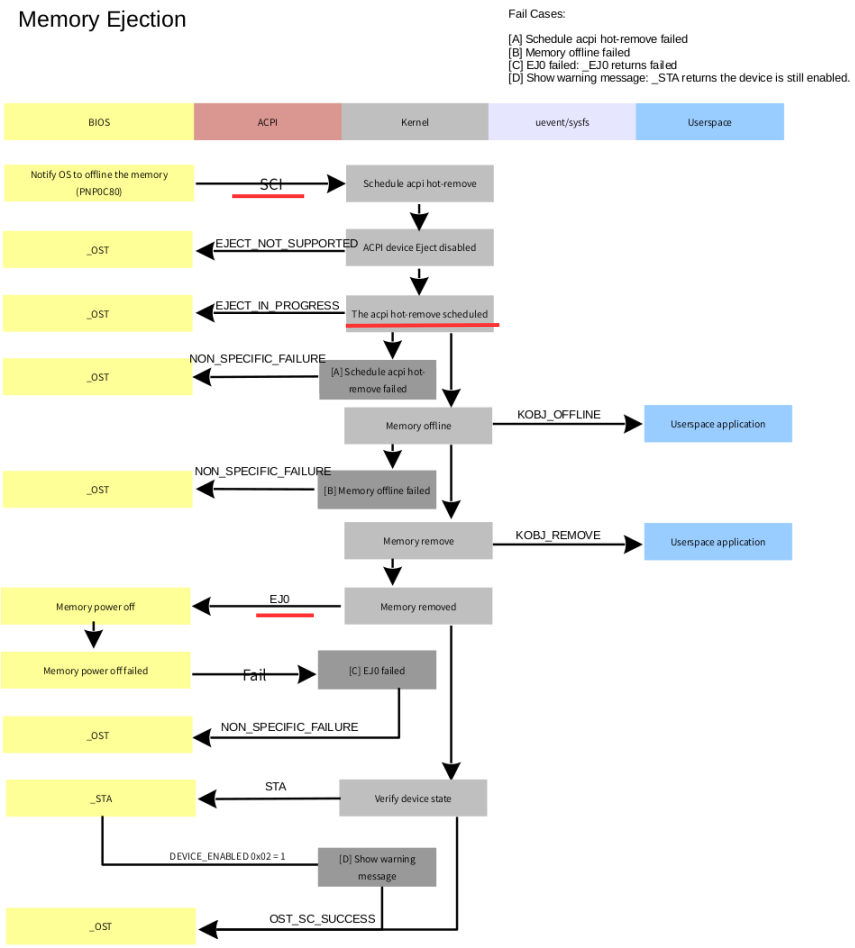


- 热移除开始
- 内存下线开始
- IIO下线开始
- IIO下线完成
- CPU下线开始
- CPU下线开始
- CPU下线开始
- 内存板下电
- 内存板下电
- 内存下线完成
- CPU下线完成
- CPU板下电
- 成功

Stress testing - fail case



Memory ejection flow



ACPI hot remove path

```
drivers/acpi/device_sysfs.c     static DEVICE_ATTR(eject, 0200, NULL, acpi_eject_store);       /* e.g. /sys/devices/LNXSYSTM:00/LNXSYBUS:00/LNXCPU:00/eject */
drivers/acpi/device_sysfs.c     static ssize_t acpi_eject_store(struct device *d, struct device_attribute *attr, const char *buf, size_t count)
        drivers/acpi/osl.c      acpi_status acpi_hotplug_schedule(struct acpi_device *adev, u32 src)

drivers/acpi/bus.c      static void acpi_bus_notify(acpi_handle handle, u32 type, void data)       /* acpi bus notify handler */
        drivers/acpi/osl.c      acpi_status acpi_hotplug_schedule(struct acpi_device *adev, u32 src)
                drivers/acpi/osl.c      static void acpi_hotplug_work_fn(struct work_struct *work)
                        drivers/acpi/scan.c     void acpi_device_hotplug(struct acpi_device *adev, u32 src)
                                drivers/acpi/scan.c     static int acpi_generic_hotplug_event(struct acpi_device *adev, u32 type)   /* adev->flags.hotplug_notify */
                                        drivers/acpi/scan.c     static int acpi_scan_hot_remove(struct acpi_device *device)             /* ACPI_NOTIFY_EJECT_REQUEST or ACPI_OST_EC_OSPM_EJECT
                                                drivers/acpi/scan.c     bool acpi_scan_is_offline(struct acpi_device *adev, bool uevent) /* device->handler->hotplug.demand_offline */
                                                drivers/acpi/scan.c     static int acpi_scan_try_to_offline(struct acpi_device *device)     /* non-container */
                                                        drivers/acpi/scan.c     static acpi_status acpi_bus_offline(acpi_handle handle, u32 lvl, void *data, void **ret_p)
                                                                drivers/base/core.c     int device_offline(struct device *dev)
                                                                        dev->bus->offline(dev);
                                                                                drivers/base/memory.c   static int memory_subsys_offline(struct device *dev)
                                                                                drivers/base/cpu.c      static int cpu_subsys_offline(struct device *dev)
                                                                                drivers/base/container.c        static int container_offline(struct device *dev)
                                                drivers/acpi/scan.c     void acpi_bus_trim(struct acpi_device *adev)                    /* Detach scan handlers and drivers from ACPI device objects. */
                                                        handler->detach(adev);                                                  /* if has acpi_device->handler->detach */
                                                                drivers/acpi/acpi_memhotplug.c  static void acpi_memory_device_remove(struct acpi_device *device)
                                                                drivers/acpi/acpi_processor.c   static void acpi_processor_remove(struct acpi_device *device)
                                                        drivers/base/dd.c       device_release_driver(&adev->dev);                      /* if no acpi_device->handler, acpi_scan_handler */
                                                        drivers/acpi/device_pm.c        acpi_device_set_power(adev, ACPI_STATE_D3_COLD);    /* put device into D3cold before it's going away */
                                                drivers/acpi/utils.c    acpi_status acpi_evaluate_ej0(acpi_handle handle)               /* _EJ0 */
                                                drivers/acpi/utils.c    acpi_evaluate_integer(handle, "_STA", NULL, &sta);              /* _STA, verify device state */
|
                                                                                                                        Offline stage   Remove stage    Power Off
```

Work queue and device_hotplug_lock

Push event to queue

```
drivers/acpi/device_sysfs.c     static DEVICE_ATTR(eject, 0200, NULL, acpi_eject_store);       /* e.g. /sys/devices/LNXSYSTM:00/LNXSYBUS:00/LNXCPU:00/eject */
drivers/acpi/device_sysfs.c     static ssize_t acpi_eject_store(struct device *d, struct device_attribute *attr, const char *buf, size_t count)
        drivers/acpi/osl.c      acpi_status acpi_hotplug_schedule(struct acpi_device *adev, u32 src)

drivers/acpi/bus.c      static void acpi_bus_notify(acpi_handle handle, u32 type, void data)       /* acpi bus notify handler */
        drivers/acpi/osl.c      acpi_status acpi_hotplug_schedule(struct acpi_device *adev, u32 src)
                drivers/acpi/osl.c      static void acpi_hotplug_work_fn(struct work_struct *work)
                        drivers/acpi/scan.c     void acpi_device_hotplug(struct acpi_device *adev, u32 src)
                                drivers/acpi/scan.c     static int acpi_generic_hotplug_event(struct acpi_device *adev, u32 type)   /* adev->flags.hotplug_notify */
                                        drivers/acpi/scan.c     static int acpi_scan_hot_remove(struct acpi_device *device)             /* ACPI_NOTIFY_EJECT_REQUEST or ACPI_OST_EC_OSPM_EJECT
                                                drivers/acpi/scan.c     bool acpi_scan_is_offline(struct acpi_device *adev, bool uevent) /* device->handler->hotplug.demand_offline */
                                                drivers/acpi/scan.c     static int acpi_scan_try_to_offline(struct acpi_device *device)    /* non-container */
                                                        drivers/acpi/scan.c     static acpi_status acpi_bus_offline(acpi_handle handle, u32 lvl, void *data, void **ret_p)
                                                                drivers/base/core.c     int device_offline(struct device *dev)
                                                                        dev->bus->offline(dev);
                                                                                drivers/base/memory.c   static int memory_subsys_offline(struct device *dev)
                                                                                drivers/base/cpu.c      static int cpu_subsys_offline(struct device *dev)
                                                                                drivers/base/container.c        static int container_offline(struct device *dev)
                                                drivers/acpi/scan.c     void acpi_bus_trim(struct acpi_device *adev)                    /* Detach scan handlers and drivers from ACPI device objects. */
                                                        handler->detach(adev);                                                  /* if has acpi_device->handler->detach */
                                                                drivers/acpi/acpi_memhotplug.c  static void acpi_memory_device_remove(struct acpi_device *device)
                                                                drivers/acpi/acpi_processor.c   static void acpi_processor_remove(struct acpi_device *device)
                                                        drivers/base/dd.c       device_release_driver(&adev->dev);                      /* if no acpi_device->handler, acpi_scan_handler */
                                                        drivers/acpi/device_pm.c        acpi_device_set_power(adev, ACPI_STATE_D3_COLD);    /* put device into D3cold before it's going away */
                                                drivers/acpi/utils.c    acpi_status acpi_evaluate_ej0(acpi_handle handle)               /* _EJ0 */
                                                drivers/acpi/utils.c    acpi_evaluate_integer(handle, "_STA", NULL, &sta);              /* _STA, verify device state */
```

lock_device_hotplug() { mutex_lock(&device_hotplug_lock); }

Offline stage Remove stage Power Off

set-acpi-ftrace.sh

```
#!/bin/bash
echo 0 > /sys/kernel/debug/tracing/tracing_on
echo > /sys/kernel/debug/tracing/trace
echo > /sys/kernel/debug/tracing/set_event

echo function_graph > /sys/kernel/debug/tracing/current_tracer

echo acpi_device_hotplug > /sys/kernel/debug/tracing/set_ftrace_filter
echo acpi_evaluate_ej0 >> /sys/kernel/debug/tracing/set_ftrace_filter
echo acpi_bus_trim >> /sys/kernel/debug/tracing/set_ftrace_filter
echo acpi_memory_device_remove >> /sys/kernel/debug/tracing/set_ftrace_filter

echo arch_remove_memory >> /sys/kernel/debug/tracing/set_ftrace_filter
echo remove_memory >> /sys/kernel/debug/tracing/set_ftrace_filter
echo acpi_pci_root_remove >> /sys/kernel/debug/tracing/set_ftrace_filter

echo *pci* >> /sys/kernel/debug/tracing/set_ftrace_filter
echo pci* >> /sys/kernel/debug/tracing/set_ftrace_filter
echo *pci >> /sys/kernel/debug/tracing/set_ftrace_filter
echo pci_bus_read_config_word >> /sys/kernel/debug/tracing/set_ftrace_notrace
echo pci_read >> /sys/kernel/debug/tracing/set_ftrace_notrace
echo raw_pci_read >> /sys/kernel/debug/tracing/set_ftrace_notrace
echo pci_conf1_read >> /sys/kernel/debug/tracing/set_ftrace_notrace

echo 1 > /sys/kernel/debug/tracing/tracing_on
```

/sys/kernel/debug/tracing/trace_pipe

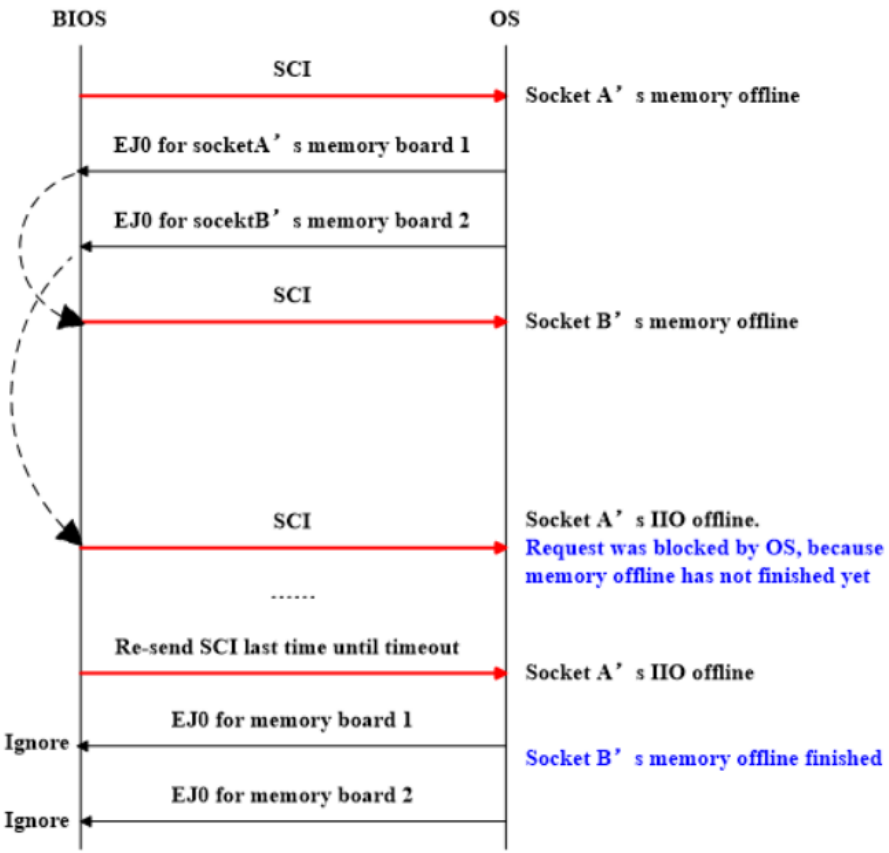
```
 172)               |          arch_remove_memory() {
 348) $ 1754161 us  |          } /* arch_remove_memory */
 348) $ 1754350 us  |        } /* remove_memory */
 348) $ 1754987 us  |      } /* acpi_memory_device_remove */
 348) $ 1754993 us  |    } /* acpi_bus_trim */
 348)               |    acpi_evaluate_ej0() {
 348) ! 496.953 us  |    }
 348)               |    acpi_evaluate_ost() {
 348) + 78.690 us   |    }
 348) $ 469711555 us |  } /* acpi_device_hotplug */
 348)               |  acpi_device_hotplug() {
 348) ! 122.228 us  |    acpi_evaluate_ost();
 348)   1.303 us    |    acpi_bus_trim();
 348)               |    acpi_evaluate_ej0() {
 140) $ 18446656050459455 us |    } /* acpi_evaluate_ej0 */
 140) + 69.866 us   |    acpi_evaluate_ost();
 140) $ 18446656050459826 us |  } /* acpi_device_hotplug */
 140)               |  acpi_device_hotplug() {
 140)               |    acpi_evaluate_ost() {
 140) ! 122.777 us  |    }
 140)               |    acpi_bus_trim() {
 140)   2.563 us    |      acpi_bus_trim();
 140)   0.486 us    |      acpi_bus_trim();
[...snip]
 140)   0.575 us    |      acpi_bus_trim();
 140)   0.761 us    |      acpi_bus_trim();
 140)               |      acpi_pci_root_remove() {
 140)               |        pci_remove_root_bus() {
 153) @ 676601.3 us |        } /* pci_remove_root_bus */
 153) @ 964276.4 us |      } /* acpi_pci_root_remove */
 153) @ 964485.3 us |    } /* acpi_bus_trim */
 153) ! 341.277 us  |    acpi_evaluate_ej0();
 153) + 58.804 us   |    acpi_evaluate_ost();
 153) @ 965701.3 us |  } /* acpi_device_hotplug */
```

BIOS retry time out

- IIO SCI was triggered before the memory offline finished. Base on Huawei's design document , the IIO SCI should be launched _AFTER_ memory hot-remove finished. Looks the practical implementation doesn't like their design.
- At the same time, the memory hot-remove in kernel is still running and the IIO offline event is scheduled in queue to wait the lock. After 11 IIO SCI re-send by BIOS, the whole BIOS process time out because memory hot-remove didn't finish and IIO hotremove didn't start.

BIOS SCI flow

- With pressure test, the flow of CPU-pair hot-remove. It results IIO offline failed



he zhiping, bsc#1043764 comment#46

time_stamp delta too big in ring_buffer

```
[ 1673.151059] Offlined Pages 524288
[ 1673.195754] ------------[ cut here ]------------
[ 1673.195788] WARNING: CPU: 146 PID: 1843 at ../kernel/trace/ring_buffer.c:2682 rb_handle_timestamp.isra.45+0x6c/0x80()
[ 1673.195796] Delta way too big! 18446657706480853711 ts=18446657706480853711 write stamp = 0
               If you just came from a suspend/resume,
               please switch to the trace global clock:
                 echo global > /sys/kernel/debug/tracing/trace_clock
[ 1673.195890] Modules linked in: af_packet iscsi_ibft iscsi_boot_sysfs iTCO_wdt iTCO_vendor_support intel_rapl x86_pkg_t
prng aesni_intel aes_x86_64 lrw gf128mul glue_helper ablk_helper cryptd pcspkr nls_iso8859_1 nls_cp437 vfat igb joydev fa
c_i801 shpchp mfd_core mei edac_core wmi fjes processor ext4 crc16 jbd2 mbcache hid_generic usbhid sd_mod crc32c_intel ql
sas button sg dm_multipath dm_mod scsi_dh_rdac scsi_dh_emc scsi_dh_alua scsi_mod efivarfs autofs4
[ 1673.195892] Supported: Yes
[ 1673.195897] CPU: 146 PID: 1843 Comm: kworker/u3072:1 Tainted: G        W          4.4.49-92.17.2.13047.1.TEST.1027153-
[ 1673.195900] Hardware name: Huawei 9016/IT91SMUB, BIOS BLXSV209 06/02/2017
[ 1673.195915] Workqueue: kacpi_hotplug acpi_hotplug_work_fn
[ 1673.195918]  0000000000000000 ffffffff8130f020 ffff88ffd6cefbf0 ffffffff818608a2
[ 1673.195920]  ffffffff8107c391 ffff88ffd6cefc80 ffff88ffd6cefc40 ffff8801de96caa8
[ 1673.195921]  ffff8801de96ca88 00000000000003e8 ffffffff8107c40c ffffffff81854110
[ 1673.195922] Call Trace:
[ 1673.195957]  [<ffffffff81019a99>] dump_trace+0x59/0x310
```

/sys/kernel/debug/tracing/trace_clock

```
trace_clock:

        Whenever an event is recorded into the ring buffer, a
        "timestamp" is added. This stamp comes from a specified
        clock. By default, ftrace uses the "local" clock. This
        clock is very fast and strictly per cpu, but on some
        systems it may not be monotonic with respect to other
        CPUs. In other words, the local clocks may not be in sync
        with local clocks on other CPUs.

        Usual clocks for tracing:

          # cat trace_clock
          [local] global counter x86-tsc

          local: Default clock, but may not be in sync across CPUs

          global: This clock is in sync with all CPUs but may
                  be a bit slower than the local clock.

          counter: This is not a clock at all, but literally an atomic
                   counter. It counts up one by one, but is in sync
```

Soft lockups during S4 resume

bsc#860441

- Bug 860441 - [HP HPS Bug] Soft lockups during boot after resume on 12 TB system on 3.0.101-0.8 kernel
- Reproducible: Always
- Steps to Reproduce:
  - Boot 12 TB prototype on 3.0.101-0.8 kernel, default resume settings.
- Actual Results:
  - 100 seconds of soft lockup messages, garbled console output, always after the console message: Invoking userspace resume from /dev/disk/by-id/scsi-3600c0ff0001a852504bbce5201000000-part2
- Expected Results:
  - No soft lockup messages, garbled console output, and no boot hangs.

Comment#40

- Randy Wright <rwright@hpe.com> 2014-05-16 18:41:54 UTC
- Output from strace /usr/sbin/resume /dev/mapper/3600c0ff0001a8525e85d695301000000_part2
- The lengthy delay in program execution - and tracebacks on the console - occur while resume is executing close(4) as printed on line 211 of the strace output.

strace /usr/sbin/resume

```
mlockall(MCL_CURRENT|MCL_FUTURE)                = 0
open("/dev/snapshot", O_WRONLY)                 = 4
open("/dev/mapper/3600c0ff0001a8525e85d695301000000_part2", O_RDWR) = 5
lseek(5, 4068, SEEK_SET)                        = 4068
read(5, "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0SWAPSPACE2", 28) = 28
close(5)                                        = 0
close(4)                                        = 0
lseek(3, 0, SEEK_SET)                           = 0
write(3, "1\n", 2)                              = 2
close(3)                                        = 0
munmap(0x7f19a4184000, 4096)                    = 0
munmap(0x7f19a40e8000, 430080)                  = 0
exit_group(0)                                   = ?
```

long delay

perf record/record

Randy Wright 2014-06-12 22:46:18 UTC Comment 47

```
Created attachment 594490 [details]
perf.data collected from 12TB prototype

I got a chance to run resume under perf on one of the 12tb prototypes today.
Attached is the perf.data collected.  For the top entries, the results look
consistent with what I collected on the 1tb system yesterday.

hawk040os1:/tmp # echo first swap device is $sdev
first swap device is /dev/mapper/3600c0ff0001a85252064875301000000_part2
hawk040os1:/tmp # perf record -g -v /usr/sbin/resume $sdev
resume: libgcrypt version: 1.5.0
[ perf record: Woken up 33 times to write data ]
[ perf record: Captured and wrote 8.162 MB perf.data (~356582 samples) ]
hawk040os1:/tmp # perf report --stdio|head -75
# Events: 62K cycles
#
# Overhead  Command         Shared Object                          Symbol
# ........  .......  .....................  ..............................
#
    55.55%   resume  [kernel.kallsyms]     [k] memory_bm_test_bit
               |
               --- memory_bm_test_bit
                   |
                   |--100.00%-- swsusp_free
                   |          snapshot_release
                   |          __fput
                   |          filp_close
                   |          sys_close
                   |          system_call_fastpath
                   |          0x7f05aba367b0
                   |          __libc_close
                    --0.00%-- [...]

    19.59%   resume  [kernel.kallsyms]     [k] swsusp_free
```

memory_bm_test_bit() and swsusp_free()

Joerg Roedel 2014-06-24 09:43:33 UTC Comment 48

Btw, it turns out that the current bitmap implementation behind memory_bm_test_bit() already caches the last position. So the linear walk-through of the bitmaps in swsusp_free() probably can't be improved by a new data structure.

What the radix tree will improve is the average random-access time, so I give it a try. Adding a cond_resched() is also a good idea to avoid the SoftLockups.

Memory bitmap scalability improvements

- *First message in thread*
- **Joerg Roedel**
  - *Joerg Roedel*
  - *Joerg Roedel*
  - *Joerg Roedel*
  - *Joerg Roedel*
  - *Joerg Roedel*
  - *Joerg Roedel*
  - *Pavel Machek*
    - *Joerg Roedel*
      - *Joerg Roedel*
      - *Pavel Machek*
      - *Pavel Machek*
      - *Joerg Roedel*
      - *Pavel Machek*
        - *"Rafael J. Wysocki"*
  - *Joerg Roedel*
    - *Pavel Machek*

From Joerg Roedel <>

Subject [PATCH 0/6 v2] PM / Hibernate: Memory bitmap scalability improvements

Date Mon, 21 Jul 2014 12:26:56 +0200

Changes v1->v2:

* Rebased to v3.16-rc6
* Fixed the style issues in Patch 1 mentioned by Rafael

Hi,

here is the revised patch set to improve the scalability of the memory bitmap implementation used for hibernation. The current implementation does not scale well to machines with several TB of memory. A resume on those machines may cause soft lockups to be reported.

These patches improve the data structure by adding a radix tree to the linked list structure to improve random access performance from O(n) to O(log_b(n)), where b depends on the architecture (b=512 on amd64, 1024 in i386).

A test on a 12TB machine showed an improvement in resume time from 76s with the old implementation to 2.4s with the radix tree and the improved swsusp_free function. See below for details of this test.

Q&A

Reference

- [1] Documentation/trace/tracepoints.txt, Mathieu Desnoyers, Linux Kernel
- [2] Documentation/trace/tracepoint-analysis.txt, Mel Gorman, Linux Kernel

Feedback to
jlee@suse.com

Thank you.


SUSE®
We adapt. You succeed.

Corporate Headquarters
Maxfeldstrasse 5
90409 Nuremberg
Germany

+49 911 740 53 0 (Worldwide)
www.suse.com

Join us on:
www.opensuse.org