HITACHI
Inspire the Next

Development Status of Troubleshooting Features, Tracing, Message Logging in Linux Kernel

5/20/2014

Seiji Aguchi

Information & Telecommunication Systems Company
IT Platform Division Group, IT Platform R&D Management Division

Hitachi, Ltd,.

HITACHI
Inspire the Next

Contents

1.Introduction

2.Case Study (Trace)

3.Case Study (Message log)

4. Persistent Store

5. Remaining message log issue

6. Summary

HITACHI
Inspire the Next

1.Introduction

1-1. Introduction

HITACHI
Inspire the Next

Background

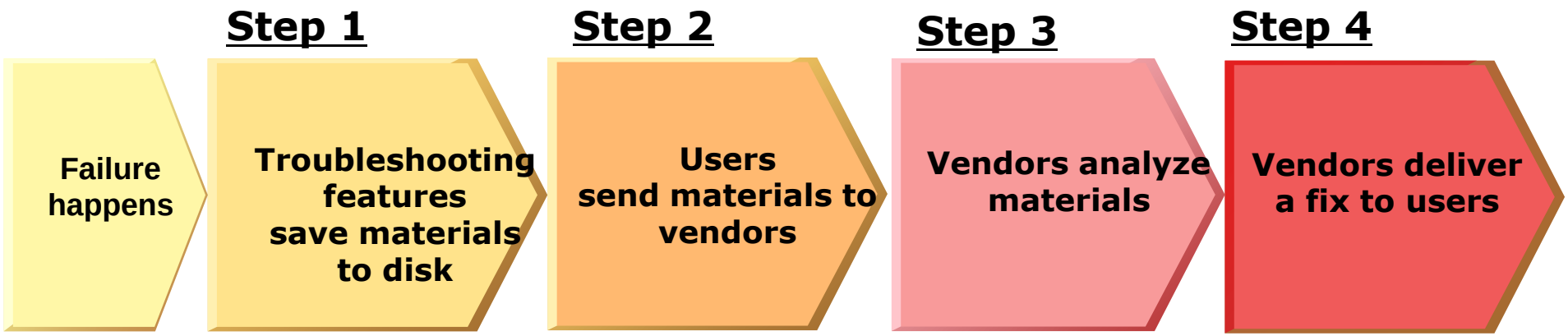
- Users in enterprise area have migrated Linux from commercial UNIX/mainframe .
- Recently, they try to apply Linux to in virtualized environment as well.

Expectation and goal

- Linux system is expected to have same level of troubleshooting features as commercial UNIX/mainframe.
- Our goal is enhancing troubleshooting features of Linux, including virtualized environment.

1-2. Troubleshooting Process

HITACHI
Inspire the Next



- **Step 1: To get sufficient materials to diagnose**
- **Step 2: To send materials quickly**
- **Step 3: To detect a root cause of failure quickly**
- **Step 4: To create a fix and deliver it to user quickly**

It is important for vendors/users that Linux has troubleshooting features enough to diagnose a failure.

1-3. How to diagnose

HITACHI
Inspire the Next

- Detailed analysis
  - Detect root cause of issues like performance and system down.
    - Tracing, memory dump are used.

- Overall analysis
  - diagnose overview of the current state of the system.
    - Massage Log is used.

1-4. Troubleshooting Feature Overview

HITACHI
Inspire the Next

- Trace (ftrace, perf, SystemTap, LTTng, uprobes)
  - Logging system's behavior in detail.
- Message Log (pstore, kmsg_dump, syslog)
  - All components, both kernel and applications, output their messages to let users know what happens in a system.
- Memory Dump (kdump, coredump)
  - Snapshot of memory to analyze a root cause of failure
- Performance Monitoring Tool (sar, iostat, netstat, oprofile, perf)
  - Gathering statistics information related to performance
- Configuration Acquisition (sosreport)
  - Gathering user's system configuration

1-4. Troubleshooting Feature Overview

HITACHI
Inspire the Next

- Trace (ftrace, perf, SystemTap, LTTng, uprobes)
  - Logging system's behavior in detail.
- Message Log (pstore, serial console, syslog)
  - All components, both kernel and applications, output their messages to let users know what happens in a system.
- Memory Dump (kdump, coredump)
  - Snapshot of memory to analyze a root cause of failure
- Performance Monitoring Tool (sar, iostat, netstat, oprofile, perf)
  - Gathering statistics information related to performance
- Configuration Acquisition (sosreport)
  - Gathering user's system configuration

HITACHI
Inspire the Next

2. Case Study (Trace)

2-1. Problem: Delay of user process

HITACHI
Inspire the Next

Problem

- A user observed that some was delayed, very infrequently.

Investigation

(1) Analyze with kernel trace by enabling system calls, context switches, process wakeups, and local timer interrupts.

(2) Analyze with kernel trace by enabling page fault events.

2-2. Analyze with context switch events.

HITACHI
Inspire the Next

#	Process cpu timestamp	trace information
1	dd-20968 [002] 5410.836234	sys_read(fd: 0, buf: a79000, count: 200)
2	dd-20968 [002] 5410.836235	sys_enter: NR 0 (0, a79000, 200, 0, 0, 0)
3	dd-20968 [002] 5410.836237	sched_switch: prev_comm=dd prev_pid=20968 prev_prio=120 prev_state=D|W ==> next_comm=kswapd next_pid=55 next_prio=120
4	dd-22237 [002] 5410.838105	sched_wakeup: comm=dd pid=20968 prio=120 success=1 target_cpu=002
5	kswapd-55 [003] 5410.884150	sched_switch: prev_comm=kswapd prev_pid=55 prev_prio=100 prev_state=S ==> next_comm=dd next_pid=20968 next_prio=120
6	dd-20968 [003] 5410.884152	sys_read -> 0x200

delay

- #1 - #2: Process 20968 runs on cpu 2.
- #3: Context switch happens.
- #4 - #5: Process 20968 moves to cpu 3 and some delay happens.
- #6: Process 20968 reruns on cpu 3.

Workaround

- Separating the CPUs by CPUSET was effective.

2-3. Analyze with page fault event.

HITACHI
Inspire the Next

#	Process	cpu	timestamp	trace information
1	app-1915	[003]	309.604958	page_fault_user: address=0x3216e01a94 ip=0x3216f2f927 error_code=0x4
2	app-1915	[003]	309.604960	page_fault_user: address=0x3216e04006 ip=0x3216f2f8ba error_code=0x4
3	app-1915	[003]	309.604964	page_fault_user: address=0x3216e0500e ip=0x3216f2f8ba error_code=0x4
4	app-1915	[003]	309.604967	page_fault_user: address=0x3216e02000 ip=0x3216f2f927 error_code=0x4
5	app-1915	[003]	309.604969	page_fault_user: address=0x3216e06000 ip=0x3216f2f8db error_code=0x4
6	app-1915	[003]	309.604973	page_fault_user: address=0x3216e07006 ip=0x3216f2f8ba error_code=0x4

- While process1915 is running, some delay happens because of page fault events.

Workaround

- To avoid page fault, pinning memory area with mlock() is effective.

HITACHI
Inspire the Next

3. Case Study (Message Log)

3-1. Boot up issue

HITACHI
Inspire the Next

Problem

- Linux system sometimes fail to boot up for some reason.
  Ex) Kernel panics before kdump service enables.

Investigation

- In these cases, administrators check kernel message.
  - If serial console is usable, kernel messages can be logged.
  - But, disk/serial console are not available, or trusted in the case of a crash, persistent store(pstore) is useful.

HITACHI
Inspire the Next

4. Persistent Store
4.1 Overview
4.2 Basice usage
4.3 Advanced usage

HITACHI
Inspire the Next

4. Persistent Store

4.1 Overview

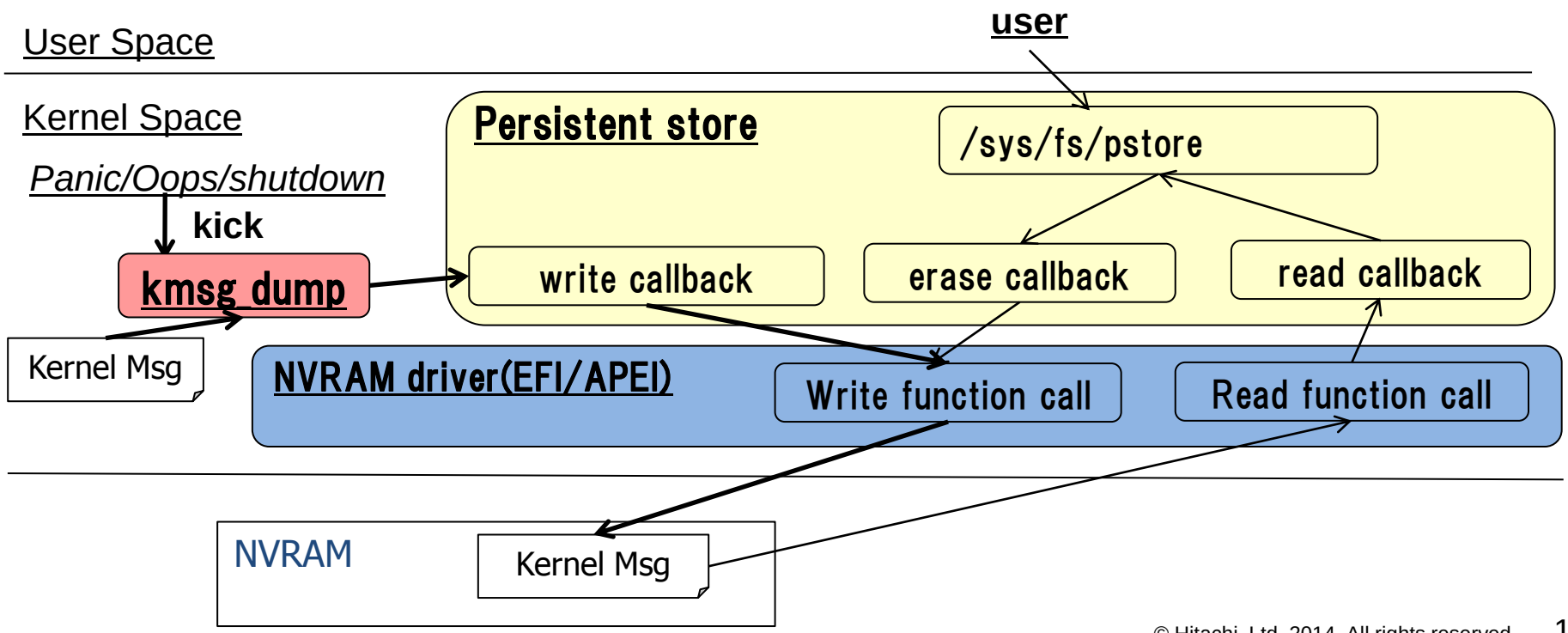
4.2 Basic usage

4.3 Advanced usage

4-1-1. Persistent Store overview

HITACHI
Inspire the Next

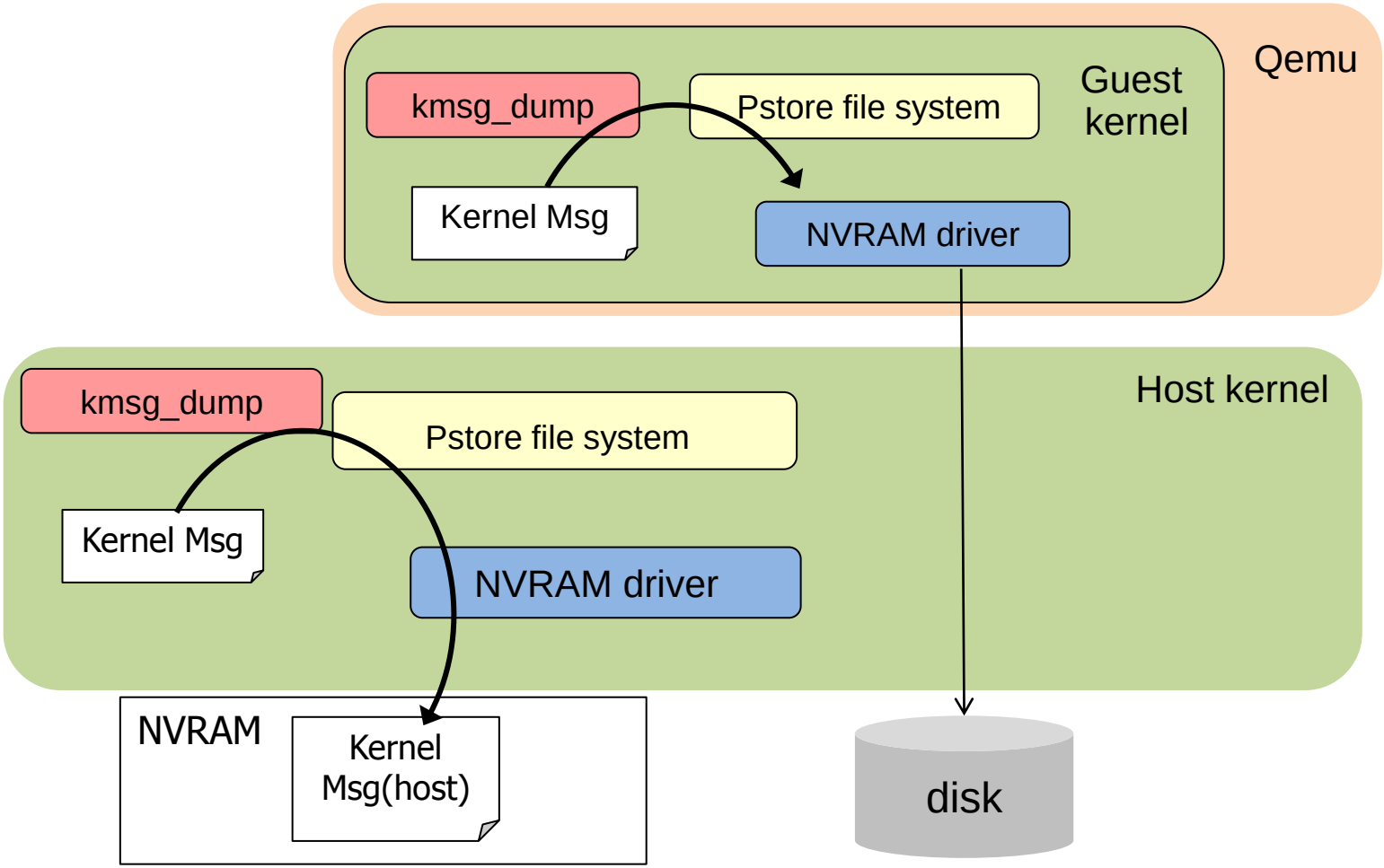
- When Linux systems crash, generally we rely on writing to log files on disk, or serial console.
- In some case, disk/serial console may not be available, or trusted in the case of a crash.
- Pstore logs messages into a platform-specific place, NVRAM.
- And it reads the data by a subsequent boot.



4-1-3. Persistent Store (Virtualized environment)

HITACHI
Inspire the Next

- Pstore is usable in kvm-virtualized environment.



HITACHI
Inspire the Next

4. Persistent Store

4.1 Overview

4.2 Basic usage

4.3 Advanced usage

4-2-1. Basic usage

HITACHI
Inspire the Next

1. Set kernel parameter to enable pstore functionality.

2. Kick trigger to log kernel message, and reboot system.

3. Check the message files via pstore filesystem.

4-2-2. Step 1: Set kernel parameter

HITACHI
Inspire the Next

- Boot system up with adding kernel parameter
  - EFI driver

```
pstore.backend=efi efi_pstore.disable=N
```

  - ERST driver

```
pstore.backend=erst
```

4-2-3. Step 2: Kick trigger

HITACHI
Inspire the Next

- The following triggers are suported.
  - kernel panic, oops, emergency_restart, reboot, power off, halt
- If you want to try “kernel panic”, execute sysrq-trigger.

```
echo c > /proc/sysrq-trigger
```

- And then, system boots up again.

4-2-4. Step 3: Check message files

HITACHI
Inspire the Next

- Mount pstore filesystem.

```
# mount –t pstore - /sys/fs/pstore
```

- Check message files.

10kB messages are logged.

Timestamp of trigger is shown.

```
# ls –l /sys/fs/pstore
total 0
-r--r--r--. 1 root root 1016 May 13 07:46 dmesg-efi-1
-r--r--r--. 1 root root 1012 May 13 07:46 dmesg-efi-10
-r--r--r--. 1 root root  948 May 13 07:46 dmesg-efi-11
-r--r--r--. 1 root root  943 May 13 07:46 dmesg-efi-2
-r--r--r--. 1 root root  677 May 13 07:46 dmesg-efi-3
-r--r--r--. 1 root root  993 May 13 07:46 dmesg-efi-4
-r--r--r--. 1 root root 1010 May 13 07:46 dmesg-efi-5
-r--r--r--. 1 root root  999 May 13 07:46 dmesg-efi-6
-r--r--r--. 1 root root  976 May 13 07:46 dmesg-efi-7
-r--r--r--. 1 root root 1006 May 13 07:46 dmesg-efi-8
-r--r--r--. 1 root root  949 May 13 07:46 dmesg-efi-9
```

4-2-5. Step 3: Check message files (Contd.)

HITACHI
Inspire the Next

- Check message.

```
# cat dmesg-efi-4
cat /sys/fs/pstore/dmesg-efi-4
Panic#2 Part4
<1>[  306.271891] IP: [<ffffffff813ba3e6>] sysrq_handle_crash+0x16/0x20
<4>[  306.271917] PGD 80a98c067 PUD 807e8e067 PMD 0
<4>[  306.271937] Oops: 0002 [#1] SMP
<4>[  306.271952] Modules linked in: tcp_lp rfcomm fuse xt_CHECKSUM
nf_conntrack_netbios_ns nf_conntrack_broadcast ipt_MASQUERADE ip6t_REJECT
xt_conntrack ebtable_nat ebtable_broute bridge stp llc ebtable_filter
ebtables ip6table_nat nf_conntrack_ipv6 nf_defrag_ipv6 nf_nat_ipv6
ip6table_mangle ip6table_security ip6table_raw ip6table_filter ip6_tables
iptable_nat nf_conntrack_ipv4 nf_defrag_ipv4 nf_nat_ipv4 nf_nat
nf_conntrack iptable_mangle iptable_security iptable_raw bnep arc4
snd_hda_codec_realtek snd_hda_codec_hdmi iTCO_wdt iTCO_vendor_support vfat
fat x86_pkg_temp_thermal coretemp kvm iwlmvm crc32_pclmul mac80211
crc32c_intel ghash_clmulni_intel microcode joydev serio_raw iwlwifi
i2c_i801 cfg80211 sdhci_pci sdhci lpc_ich mmc_core mfd_core snd_hda_intel
snd_hda_codec btusb snd_hwdep bluetooth snd_seq
```

Sysrq is triggered in this panic.

HITACHI
Inspire the Next

4. Persistent Store
   4.1 Overview
   4.2 Basic usage
   4.3 Advanced usage

4-3-1. Advanced usage (Tune log size)

HITACHI
Inspire the Next

Usage

10KB messages are logged by default. And the size is tunable by specifying “kmsg_byte” when mounting pstore filesystem.

```
# mount –t pstore –o kmsg_byte=102400 - /sys/fs/pstore
```

Usecase

- In case of kernel panic, the default 10KB messages are enough to save the panic log.
- In case of disk failure, a lot of I/O failure messages are shown, so more than 10KB messages are needed to diagnose the root cause.

4-3-2. Advanced usage (unexpected reboot)

HITACHI
Inspire the Next

Usage

By adding “printk.always_kmsg_dump” to a kernel parameter, pstore logs more than panic log.

```
printk.always_kmsg_dump=Y pstore.backend=efi efi_pstore.disable=N
```

Usecase

- Linux system sometimes reboot unexpectedly. One of the candidates is “emergency_restart” event.
- “emergency_restart” event is kicked by following triggers.
  - watchdog timer
  - hangcheck-timer
  - echo c > /proc/sysrq-trigger
- By logging the kernel message, we understand the reason why a system reboots.

4-3-3. Advanced usage (normal reboot)

HITACHI
Inspire the Next

Usage

Also, by adding “printk.always_kmsg_dump”, normal reboot events can be logged.

```
printk.always_kmsg_dump=Y pstore.backend=efi efi_pstore.disable=N
```

Usecase

- Currently, device names, sdX, may change every boot on Linux system.
- So, some command to get performance statistics like sar and iostat, may not work when this issue happens, because they access to the device name directly.

```
# iostat
…
Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda              31.29      1160.34       430.39    2181836     809280
```

“sda” may change ever boot

HITACHI
Inspire the Next

Usecase (Contd.)

- In this case, the statistics data cannot get correctly, but administrators cannot predict when this issue happens.
- To detect when the inconsistent device naming issue happened, they have to check kernel massages back past months.
- To do this, message logs need to be saved to log disk every boot as follows.

```
mv /sys/fs/pstore/*  /var/log/pstore
```

HITACHI
Inspire the Next

5. Remaining message log issue

5-1. Remaining message log issue

HITACHI
Inspire the Next

Current kernel message

- In a following pseudo SCSI error test, the device information and the detail error are divided.
- This implementation may have some issues.
  - When other messages are inserted, it may be difficult for users to match device information and the detail error.
  - When user tools handle the error messages, those divided messages will create some inconveniences.

```
[ 17.842110] sd 2:0:0:0: [sdb] Attached SCSI disk
[ 18.859098] sd 2:0:0:0: [sdb] Unhandled sense code
[ 18.859103] sd 2:0:0:0: [sdb] [ 18.859106] Result: hostbyte=DID_OK driverbyte=DRIVER_SENSE
[ 18.859108] sd 2:0:0:0: [sdb]
[ 18.859110] Sense Key : Medium Error [current]
[ 18.859114] Info fld=0x1234
[ 18.859116] sd 2:0:0:0: [sdb]
[ 18.859119] Add. Sense: Unrecovered read error
[ 18.859122] sd 2:0:0:0: [sdb] CDB:
[ 18.859124] Read(10): 28 00 00 00 11 e0 00 01 00 00
```

5-2. Remaining message log issue

HITACHI
Inspire the Next

Expected kernel message

- The device information and the detail error should be displayed in a single line to fix the issues.

```
[ 17.842110] sd 2:0:0:0: [sdb] Attached SCSI disk
[ 18.859098] sd 2:0:0:0: [sdb] Unhandled sense code
[ 18.859103] sd 2:0:0:0: [sdb] [ 18.859106] Result: hostbyte=DID_OK driverbyte=DRIVER_SENSE
[ 18.859108] sd 2:0:0:0: [sdb] Sense Key : Medium Error [current]
[ 18.859114] Info fld=0x1234
[ 18.859116] sd 2:0:0:0: [sdb] Add. Sense: Unrecovered read error
[ 18.859122] sd 2:0:0:0: [sdb] CDB: Read(10): 28 00 00 00 11 e0 00 01 00 00
```

HITACHI
Inspire the Next

6. Summary

6-1. Summary

HITACHI
Inspire the Next

- Hitachi has experienced troubleshooting thorough support service in Linux system.
- Trace is useful for diagnosing performance issue.
- Pstore is useful for troubleshooting in boot up/shutdown failure cases.
  - Kernel panics before kdump service is enabled.
  - Disk failure
  - Unexpected reboot.
  - Normal reboot.
- There is a remaining issue on kernel message and should be fixed near future.

HITACHI
Inspire the Next

END

Development Status of Troubleshooting Features, Tracing, Message Logging in Linux Kernel

5/20/2014

Seiji Aguchi

Information & Telecommunication Systems Company
IT Platform Division Group, IT Platform R&D Management Division
Hitachi, Ltd,.

Trademarks

HITACHI
Inspire the Next

•Linux is a trademark of Linus Torvalds in the United States, other countries or both.

•UNIX is a registered trademark of The Open Group in the United States and other countries.

•Other company, product or service names may be trademarks or service mark of others.


HITACHI
Inspire the Next