Empowered by Innovation NEC

Maintaining Linux Long Term & Adding Specific Features in Telecom Systems

Keika Kobayashi
NEC Communication Systems
Sep 29, 2010@LinuxCon Japan2010

OUTLINE

1. Background

2. What we did
   1. Keep kernel stable.
   2. Update kernel without rebooting.
   3. Find buggy application affecting service.
   4. Avoid delay caused by heavy disk writing.
   5. Get debugging info when the problem occurs.

3. Conclusion

1. Background

Supporting Linux for Telecom System since 2003.

High availability is important for Telecom System.

6 nines (99.9999 percent) availability is required.

- Remove factors which cause stopping service.
- Regression must be avoided.
- Must find the root cause out and fix, when the problem occurs.

Providing Long-Term Support

- Our product would keep going more than 10 years.

CGL

We are using CGL distro

CGL (Carrier Grade Linux)

- Meet specific requirements such as availability, reliability and etc for use communication service.
- CGL Specs are defined by Linux Foundation CGL Working Group.
- Consist of 7 Categories.

- Availability	- Clustering	- Serviceability	- Performance
- Standards	- Hardware	- Security	

CGL Specs History

- Aug/2002 v1.0
- Oct/2003 v2.0
- Feb/2005 v3.0
- The latest specification is v4.0. (announced Feb/2007)

2. What we did

We added specific features and maintained kernel
for keeping service available.

Most requirements in CGL spec were implemented by CGL distro and community. However there are still gaps between the current implementations and the telecom requirements.

Item List

1. Keep kernel stable.
2. Update kernel without rebooting.
3. Find buggy application affecting service.
4. Avoid delay caused by heavy disk writing.
5. Get debugging info when the problem occurs.

2.1 Keep kernel stable

Issue

There was no long term maintained stable kernel. Kernel bugs must be fixed before system would hit known problems which may cause stopping service.

General Solution

- Use stable kernel.

  Nowadays stable kernel exists.

  The stable kernel contains bug fixes only and is maintained well.
- Apply service pack released by distro.

  Distro delivers minor change for security and bug fix.

Patch filtering

Action

For providing long term support, should apply necessary bug fix.
Our kernel doesn't have stable tree. When the problem occurred, distros would fix the bug. However they didn’t apply patches in community actively in those days.
So, we should search critical issues with patch filtering in mainline and the other distros reports.

- Filtering community patches including keyword such as “panic”, “crash”, “oops”, “race” and etc in changelog.
- Picking up high severity issues in the other distro.
- Searching similar issues which occur in the other version.

Result

- Applied around 100 patches expected to fix fatal problem.

2.2 Update kernel without rebooting

Issue

- Rebooting is needed to update kernel.
  In spite of our redundant system, rebooting is not acceptable, because rebooting causes…
  - During rebooting, system reliability is lower.
  - Fail over brings stopping service for several seconds at least.

CGL Availability Requirement

- AVL.27.0 Kernel Live Patching
  - Any distro hasn't supported when we released.

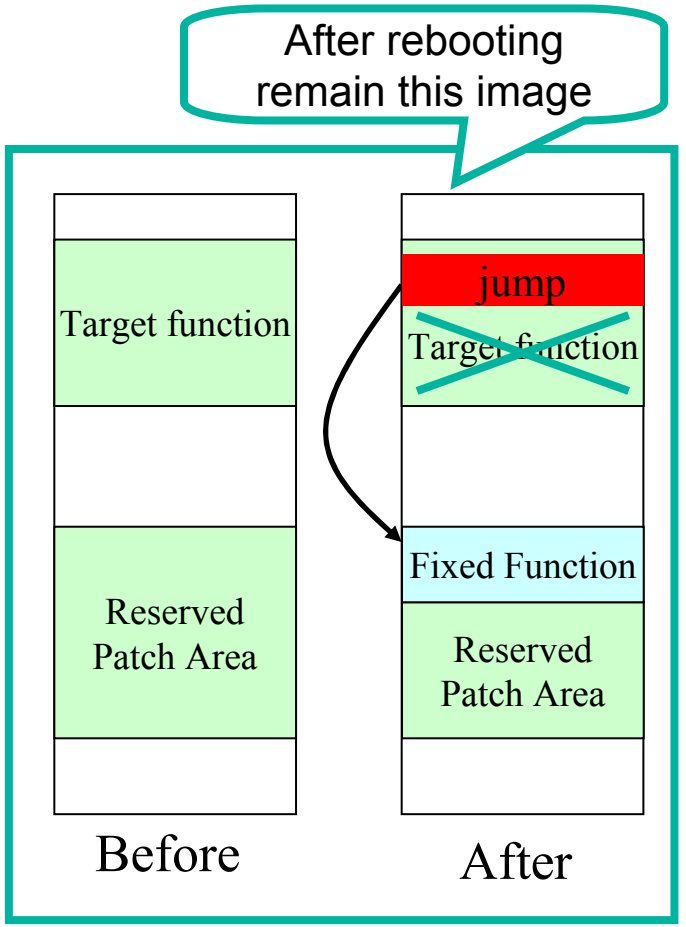
In community

- ksplice

Live Patch Mechanism

We implemented live patch system for rebootless update.
The live patching method is well known.

- Beforehand
  - There is patch area in the kernel.
- Making live patch
  - Make live patch to place in the patch area.
- Appling patch
  - Load and place that live patch in the patch area by our live patch driver.
  - Insert the jump instruction to the head of the target function.
- Replacing bzImage
  - Replace with bzImage which correspond to the fixed binary image.



Restriction & Result

Restriction

Unable to make live patch in the following cases.

- __init function
  There is no code in the live kernel.
- Any code jumps to first 5 bytes of target function.
  If any instruction jumping to the middle of the first 5 bytes exists, an invalid code might be executed. Because x86 is the CISC CPU and the jump instruction as 5bytes. Jumping to first 5 bytes may occur in our live patch system.
- Including infinite loop
  Kernel thread in infinite loop never jump to fixed function.
  Live patching can't change its behavior in such a loop.
- Add or delete member of structure.
  When fixed function receives structure data from non fixed function, data inconsistency occurs because of changing the size of structure.
- Kernel module
  No need to patch to module, we can use rmmod and insmod to replace to the fixed module.

Result

- Have updated live systems several times while they were running.

2.3 Find buggy application affecting service.

Issue

- Unable to find buggy userland process getting into busy loop.

  The process which is continuously consuming CPU than expected, may be buggy. We often saw the process causes stopping service. Should detect such a process and handle it properly as soon as possible.

CGL Availability Requirement

- AVL.14.0 Excessive CPU Cycle Usage Detection
  - Not in our kernel

In community

- Sched : SCHED_FIFO/SCHED_RR watchdog timer

  Recent kernel has RT watchdog.

Implementation

Implementation

- Accounting CPU-usage tick per process
  and the counter is reset when the process sleeps.
  e.g. sleep(),nanosleep(),select(),etc...
- If the counter reaches threshold, SIGXCPU is sent to the process
- Setting threshold with setrlimit
- The counter and threshold are carried over child process
  when calling fork().

2.4 Avoid delay caused by heavy disk writing

Issue

- Service delay occurs under heavy disk writing
  - service delay occurs by memory reclaim.
  - Increasing dirty pages causes lack of available memory until dirty pages become clean.
    - Dirty pages should be flushed to disk before they can be freed.
    - Clean pages will be free memory quickly.
  - They are increased by file writing.
    - When file writing is faster than physical disk's spec.
    - Because Linux use as much memory as possible.

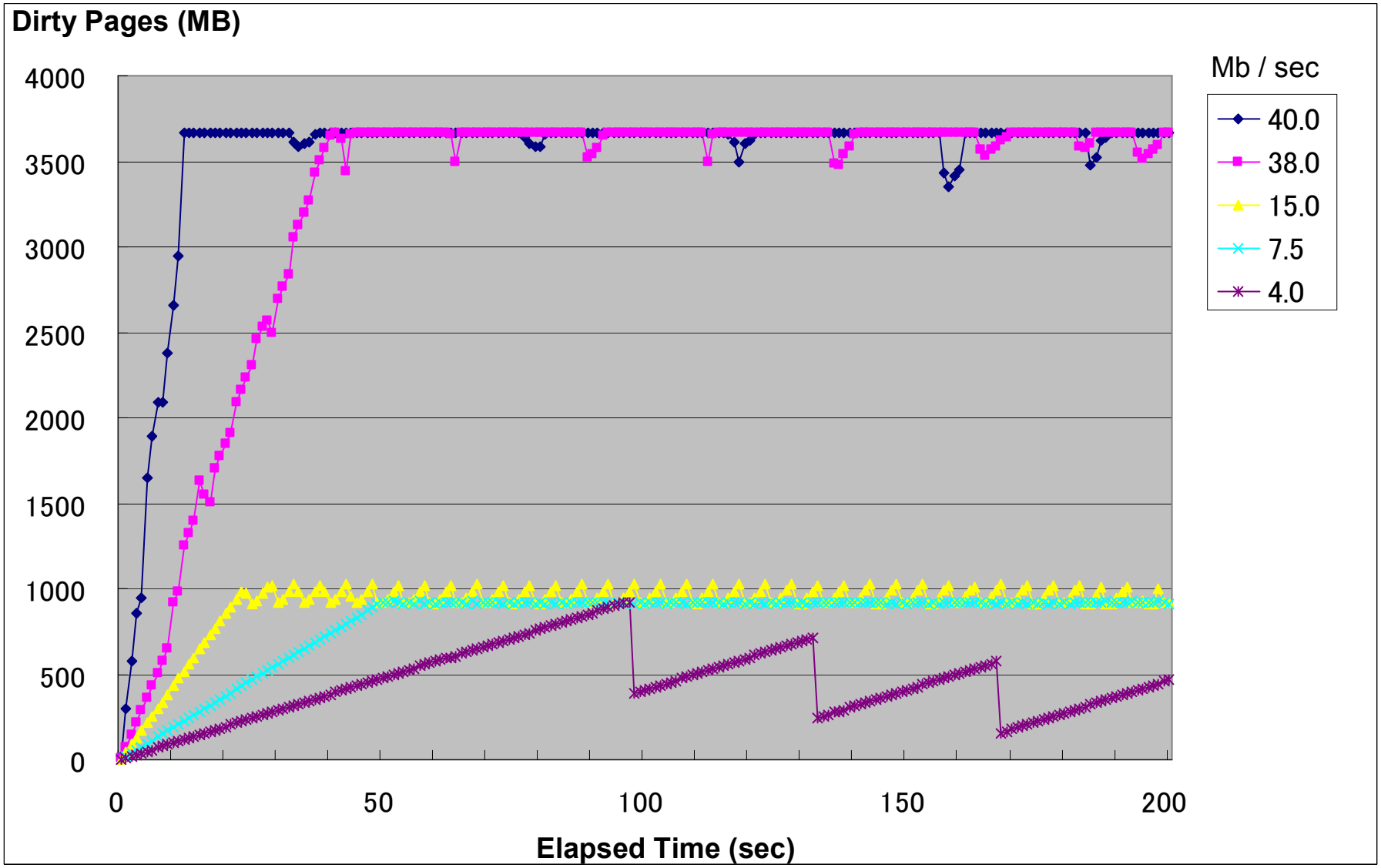
In community

- Not yet

Action

- Suppress disk writing and control amount of dirty pages for system.

Relation between file writing and dirty page



CPU : Xeon 2.13GHz x 4
Mem : 8G

2.5 Get debugging info when the problem occurs

Issue

- When taking crash dump fails, unable to find the cause of the problem.

  When system panic occurs, we need to analyze the problem.

  To fix the problem, it is necessary to find the root cause. If we don't fix it and the critical issue remains in system, availability will be lower.

CGL Serviceability Requirement

- SFA.1.0 Kernel Panic Handler Enhancements

  Some items as system panic behavior are required.

  - Forcing a crash dump
    - kdump since 2.6.13
    - LKCD before
  - Logging the panic event to the system event log

    Log for panic event is not enough for problem analyze.

Get debugging info when the problem occurs (cont.)

In community

- Crash Dump
  - kdump

    Kdump is reliable, because crash dump is captured from booted new kernel.
    Dumping process can avoid bad influence of the crushed kernel.
  - LKCD (Linux Kernel Crash Dump)

    LKCD is not more reliable than kdump.
    When system panic occurs, some kind of inconsistency occurs in running kernel and crash dump is captured from context of the crushed kernel.
- Oops message

  Linux provides oops message on serial or VGA console for problem analyze.

Motivation

- Our kernel doesn't include kdump

  =>Improved LKCD for getting more reliable

- Even if improved LKCD, crash dump sometimes fails.
  And the console is not often available in customer environment.

  =>Logging to System Event Log

Improvement of LKCD

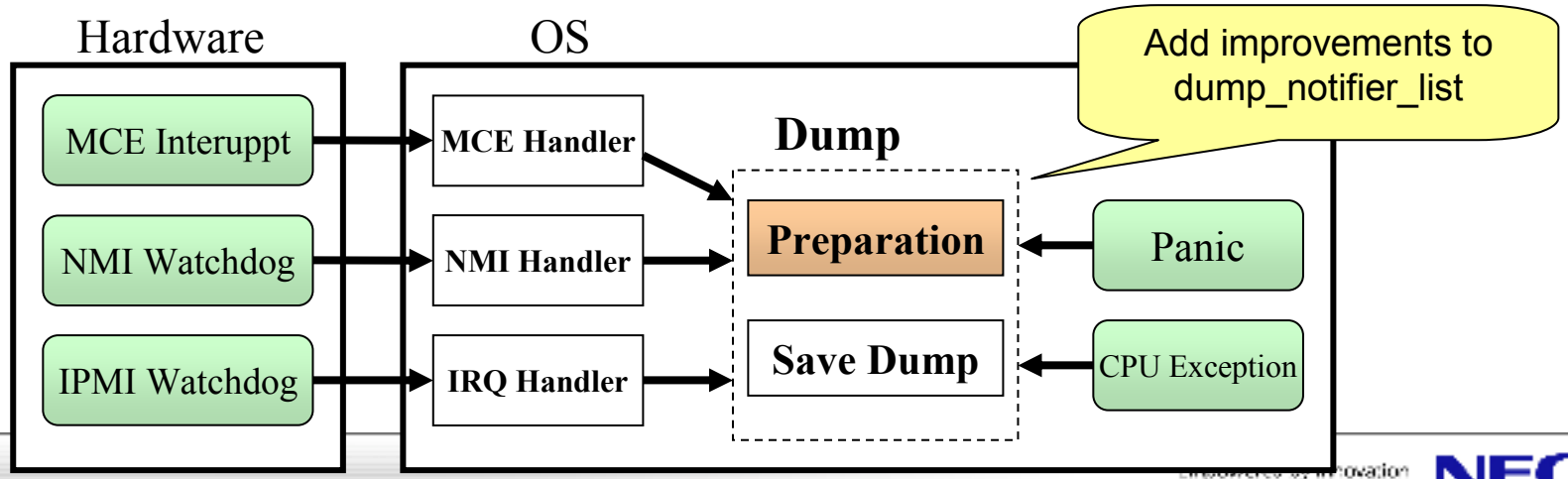
Improvement

- Force unlock the xtime_lock

  When making dump header, LKCD calls do_gettimeofday().
  If panic process gets lock of the xtime_lock, it will lead to deadlock.
  So, before dumping, force unlock the xtime_lock.

- Stop timer Interrupts

  LKCD doesn't know any spinlock status before kernel crash. During taking dump, timeout functions from timer interrupts called at same CPU. If the function requires spinlock which has already locked before crash, LKCD failed caused by deadlock.
  So, stop timer interrupts before taking dump and never raise running timer.



Logging to System Event Log

Implementation

Logging to Non-Volatile Memory.

Sytem Event Log(SEL) is available for Logging.

So, we store Oops message to SEL.

- SEL
  - Non-volatile memory of BMC for logging
  - Able to write records with IPMI from OS
  - SEL's capacity is small and stored Oops message should be compressed.

```
lock 0000: rq 0 tsk 6942@2 on Panic
CPU#2 Pid:6942[00000101c931d800]
comm:one_prcs
RIP:0010:[ffffffffa00f5f04]
RSP: 0000:00000100cfeafa38 EFLAGS: 00000046
RAX:0000000000000002 RBX:0000000000000002
RCX:0000000000000028
RDX:0000000000000000 RSI:00000000000000c8
RDI:00000100cfeafbe8
RBP:00000000000000c8 R08:0000000000000043
R09:0000000000000000
R10:0000000000000000 R11:0000000000000001
R12:0000000000000000
R13:0000000000000001 R14:0000000000000029
R15:0000000000000000

Over IRQ#11

<ffffffff802ea7f4>{start_next_msg+150}
<ffffffff80292d61>{vsprintf+20}
<ffffffff801b56d7>{notifier_call_chain+31}
<ffffffff801a7032>{panic+219}
```

Result

- Able to analyze the problem without crash.

3. Conclusion

- Solved many issues which we encountered in long term support
- Kernel live patch is needed for telecom service.
  We implemented and used live patch system.
- RT watchdog is helpful for keeping availability.
- Improvement of LKCD is effective in some cases
- By storing oops message to SEL, it is possible to analyze the problem without crash.
- OSS is customizable and we can provide flexible support.

Empowered by Innovation