Automated Linux Kernel Crash Infrastructure – Eye in the Digital Sky

Igor Ljubuncic
OS Architect, Intel Corporation
Co authors: Raphael Sack

IT@Intel
Legal Notices

This presentation is for informational purposes only. INTEL MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

For more complete information about performance and benchmark results, visit www.intel.com/benchmarks

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

* Other names and brands may be claimed as the property of others.

Copyright © 2014, Intel Corporation. All rights reserved.
Agenda

• Overview of Intel chip design
• A day in life
• Kernel crash mechanism
• Architecture
• Monitoring & reporting
• Crash data & classification

• Examples
• Analysis
• Contribution
• Challenges
• Benefits
• Summary
Igor Ljubuncic

• OS Architect, Engineering Computing
• Technical and non-technical books, open-source projects, patents
• Petrol head
2014 Intel IT Vital Statistics

>6,300 IT employees
  59 global IT sites

>98,000 Intel employees¹
  168 Intel sites in 65 Countries

64 Data Centers
  (91 Data Centers in 2010)
  80% of servers virtualized
  (42% virtualized in 2010)

>147,000+ Devices
  100% of laptops encrypted
  100% of laptops with SSDs
  >43,200 handheld devices
  57 mobile applications developed

Source: Information provided by Intel IT as of Jan 2014
¹Total employee count does not include wholly owned subsidiaries that Intel IT does not directly support
Overview of Intel chip design env

• 40 sites globally
• 600,000 cores
• 5 PB memory
• 24 PB distributed storage
• Batch environment
• 40M+ regressions/month
A day in life

• Thousands of servers, homogenous environment
• One goes down with panic
• You start to panic
Linux Kernel Crash mechanism

- Kdump (http://lse.sourceforge.net/kdump/)
- Collect memory core on oops or panic
- Analyze & resolve
- Automation?
Our architecture

• Kdump setup (via configuration management)
• Crash analysis init script (via configuration management)
• Central repository (NFS)
• Crash database (SQL)
• Crash monitoring (alerts, trends)
Architecture

- Panic Kdump
  - Reboot
  - Crash init script

- NFS crash repository (auto cleanup)

- Crash database

- Data center 1
  - Monitoring console (email, 24/7)
  - Reporting (email, Web)

- Data center 2
- Data center X
## Monitoring & reporting

<table>
<thead>
<tr>
<th>Date</th>
<th>Source</th>
<th>IP</th>
<th>Hostname</th>
<th>Message</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>05/06/14</td>
<td>HW Health</td>
<td>10.0.4.22</td>
<td>pdc15</td>
<td>RAID controller predictive failure detected</td>
<td>2.6.33</td>
</tr>
<tr>
<td>05/06/14</td>
<td>Crash Collection</td>
<td>10.100.3.105</td>
<td>testhost1</td>
<td>Unknown kernel crash exception [cache_alloc_refill+231]</td>
<td>2.6.33</td>
</tr>
<tr>
<td>05/06/14</td>
<td>Disk space</td>
<td>10.101.22.16</td>
<td>nissrv3</td>
<td>Low disk space on /tmp [&lt;3% free]</td>
<td>2.6.18</td>
</tr>
</tbody>
</table>
Monitoring & reporting

Kernel crashes WW33

Number of kernel crashes

Kernel version

- 2.6.16
- 2.6.18
- 2.6.32
- 3.0.13
Crash data

• Backtrace (bt)
• Process tree (ps)
• Kernel buffer log (log)
Interesting information

Exception RIP (unique identifier) is the most important piece

PID: 0    TASK: ffffffff80300ae0    CPU: 0    COMMAND: "swapper"
#0 [fffffff80440f20] crash_nmi_callback at ffffffff8007a68e
#1 [fffffff80440f40] do_nmi at ffffffff8006585a *
#2 [fffffff80440f50] nmi at ffffffff80064ebf *
   [exception RIP: default_idle+61]
RIP: ffffffff8006b301    RSP: ffffffff803f3f90    RFLAGS: 00000246
RAX: 0000000000000000    RBX: ffffffff8006b2d8    RCX: 0000000000000000
RDX: 0000000000000000    RSI: 0000000000000001    RDI: ffffffff80302698
RBP: 0000000000090000    R8: ffffffff803f2000    R9: 000000000000003e
R10: ffff810107154038    R11: 0000000000000246    R12: 0000000000000000
R13: 0000000000000000    R14: 0000000000000000    R15: 0000000000000000
ORIG_RAX: ffffffff80000000    CS: 0010    SS: 0018
--- <exception stack> ---
#3 [fffffff803f3f90] default_idle at ffffffff8006b301 *
#4 [fffffff803f3f90] cpu_idle at ffffffff8004943c
## Crash reason classification

<table>
<thead>
<tr>
<th>Single host</th>
<th>Multiple hosts</th>
<th>Single exception</th>
<th>Multiple exceptions</th>
<th>Multiple crashes</th>
<th>Same platform model?</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>HW</td>
</tr>
<tr>
<td>-</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>HW/BIOS</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>BUG</td>
</tr>
</tbody>
</table>
Example, hardware problem

HARDWARE ERROR
CPU 6: Machine Check Exception:
5 Bank 3:
b62000070002010a
RIP !INEXACT! 33:<00002aaac1de6224>
TSC 33241ebfff96 ADDR 7641b7080
This is not a software problem!
Run through mcelog --ascii to decode and contact your hardware vendor
Example, kernel bug

[ 349.339841] BUG: unable to handle kernel NULL pointer dereference at (null)
[ 349.339844] IP: [<fffffffffa0402029>] init_module+0x19/0xff0
[null_pointer]
[ 349.339849] PGD 80b9e5067 PUD 80bc6d067 PMD 0
[ 349.339852] Oops: 0002 [#1] SMP
[ 349.339861] CPU 0
Analysis

• Automated analysis by crash script
• Alert on special cases (new exceptions, multiple hosts, etc)
• Engagement with vendor (critical!!)
In the past four years, we have reported more than 100 unique cases of kernel crashes to operating system vendors, unknown in the IT industry beforehand.
Challenges

• High level of expertise needed
• Hardware issues can be tricky
• Open source != 100% access to kernel source code
• Behind the scenes, little visibility
ROI

• Why not vendors do all the hard work?
• Monetary return?
• Impact outside the company?
ROI

Unique environment setup
First to discover many of the problems
Huge impact on time-critical business environment

Scenario: 30 crash reasons/year, 0.05 incidence rate, 10K hosts, 1 hour productivity loss per crash

15,000 crashes in a single year
15,000 hours of productivity lost
Summary

• A must in any big environment
• We saw a x50-100 reduction in crash rate since adoption
• Full situational awareness
• Improve stability, maximize uptime
• Community & upstream contribution
Automated Linux Kernel Crash Infrastructure—Eye in the Digital Sky

Despite popular myths, Linux systems can crash, a situation known as oops or panic. When this happens at home, you are inconvenienced. When a critical bug in the kernel causes a production server to stop working, the importance of environment stability and control gains more focus. Linux kernel crashes quickly can escalate from single host events into widespread outages. We want to identify issues in the Linux kernel quickly and contain and resolve them without any adverse impact or downtime for our customers—and we have a solution.

IGOR LJUBUNCIĆ and RAPHAEL SACK
Questions?