A Cross-Layer Approach for Diagnosing Storage System Failures

Duo Zhang, Chander Gupta, Mai Zheng
Iowa State University

Adam Manzanares, Filip Blagojevic, Cyril Guyot
Western Digital Research

1 Motivation

Diagnosing storage system failures is challenging even for professionals. One example is the "When Solid State Drives Are Not That Solid" incident occurred at Algolia data center, where a random subset of SSD based servers crashed for unknown reasons [5]. After looking into almost all software deployed, the engineers finally (mistakenly) concluded that Samsung SSDs were to blame. Samsung SSDs were criticized and blacklisted, until one month later a Linux kernel bug was identified as the root cause [4].

Similarly, Zheng *et al.* studied the behavior of SSDs under power fault [12]. The testing framework relied on the block I/O layer to apply workloads and check the behavior of devices. The initial experiments were performed on Linux kernel v2.6.32, and eight SSDs exhibited serialization failures [12]. However, in their follow-up work [13] where similar experiments were conducted on a newer kernel (v3.16.0), they observed that the failures symptoms on some SSDs changed significantly. The authors analysed hundreds of suspicious kernel patches and eventually confirmed that the different symptoms were caused by a Linux kernel bug [13].

The two cases above share a common mistake: people try to explain the behavior of SSDs *indirectly* through the OS kernel, with the (wrong) assumption that the kernel is correct. This is natural in practice as applications have to access the device via system calls. Also, SSDs are relatively young, and seems less trustable. We name such common practice as a *top-down indirect* approach. Nevertheless, both cases show that the kernel may play a role in causing system failures, while the device may be innocent. In fact, similar confusing and debatable failures are not uncommon [2,3,6]. With the system complexity increasing, the challenge of failure diagnoisis will likely increase too.

2 Design & Implementation

To help diagnose storage system failures, we introduce a *cross-layer* approach called XDB. As shown in Figure 1a, XDB hosts the target storage software stack in a custom virtual machine (VM). Similar to a classic bus analyzer [1], XDB captures the commands (e.g., SCSI, NVMe) transferred between the OS kernel and the storage device directly (via *DevAgent*)to help understand the interaction between the kernel and the device. Different from the bus analyzer, XDB leaverages virtualization to eliminate the dependence on special hardware (e.g., snooping cables) and the associated cost/inconvenience. Moreover, to help understand the high-level semantics and causal relation, XDB collects host-side functin calls (via *HostAgent*) and correlates events across layers and traces. XDB helps in two ways: (1) Combine events from different layers (i.e., host and device) into a cross-layer trace for analyzing the end-to-end system behavior; (2) Automatically identify potential problematic regions in the trace based on reference rules, which can be specified based on either domain knowledge (e.g., an `fsync` call should generate 0x35 SYNC commands in SCSI) or non-failure execution traces (e.g., due to different software versions or non-determinism).

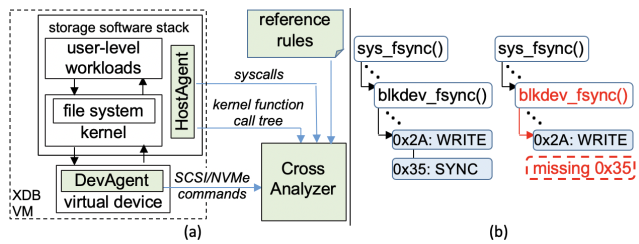


Figure 1: **(a) Design Overview; (b) Example Result.**

3 Preliminary Result

We have built a prototype of XDB based on QEMU [7] and applied it to diagnose 12 system failures caused by bugs in file systems, block layer, etc reported recently [8–11, 13]. We find that XDB is particularly helpful for understanding synchonrization issues where buggy functions lead to unexpected command sequences. Figure 1b shows the simplified result of diagnosing one of the aforementioned cases in section 1 . By using the cross-layer traces, it is easy to see that the `blkdev_fsync` in one execution (right-hand side) failed to generate a SYNC command at the device level.

References

[1] Bus analyzer wiki. https://en.wikipedia.org/wiki/Bus_analyzer.

[2] Discussion on data loss on mSATA SSD module and Ext4 . http://pcengines.ch/msata16a.htm, 2016.

[3] Failure on FreeBSD/SSD: Seeing data corruption with zfs trim functionality . https://lists.freebsd.org/pipermail/freebsd-fs/2013-April/017145.html, April 29, 2013.

[4] raid0: data corruption when using trim. https://www.spinics.net/lists/raid/msg49440.html, July 19, 2015.

[5] When solid state drives are not that solid. https://blog.algolia.com/when-solid-state-drives-are-not-that-solid/, June 15, 2015.

[6] Discussion on kernel TRIM support for SSDs: [1/3] libata: Whitelist SSDs that are known to properly return zeroes after TRIM . http://patchwork.ozlabs.org/patch/407967/, Nov 7 - Dec 8, 2014.

[7] Fabrice Bellard. Qemu, a fast and portable dynamic translator. In *USENIX Annual Technical Conference, FREENIX Track*, volume 41, page 46, 2005.

[8] Seulbae Kim, Meng Xu, Sanidhya Kashyap, Jungyeon Yoon, Wen Xu, and Taesoo Kim. Finding semantic bugs in file systems with an extensible fuzzing framework. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP'19)*, pages 147–161, 2019.

[9] Lanyue Lu, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, and Shan Lu. A study of linux file system evolution. In *Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST'13)*, pages 31–44, San Jose, CA, 2013.

[10] Jayashree Mohan, Ashlie Martinez, Soujanya Ponnapalli, Pandian Raju, and Vijay Chidambaram. Finding crash-consistency bugs with bounded black-box crash testing. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 33–50, 2018.

[11] Yiliang Shi, Danny V. Murillo, Simeng Wang, Jinrui Cao, and Mai Zheng. A command-level study of linux kernel bugs. In *2017 International Conference on Computing, Networking and Communications (ICNC'17)*, 2017.

[12] Mai Zheng, Joseph Tucek, Feng Qin, and Mark Lillibridge. Understanding the robustness of SSDs under power fault. In *Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST'13)*, 2013.

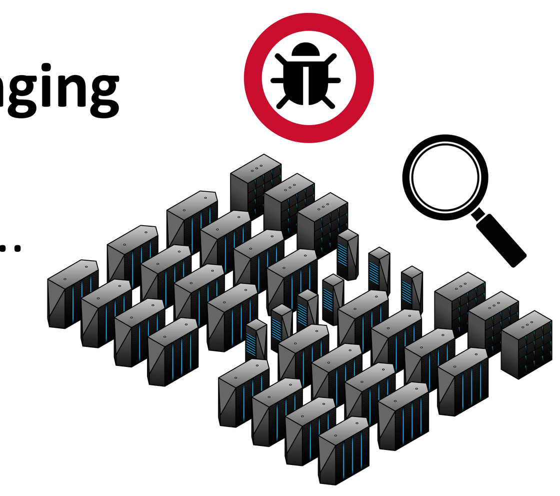
[13] Mai Zheng, Joseph Tucek, Feng Qin, Mark Lillibridge, Bill W Zhao, and Elizabeth S. Yang. Reliability Analysis of SSDs under Power Fault. In *Proceedings of the ACM Transactions on Computer Systems (TOCS'17)*, 2017.

A Cross-Layer Approach for Diagnosing Storage System Failures

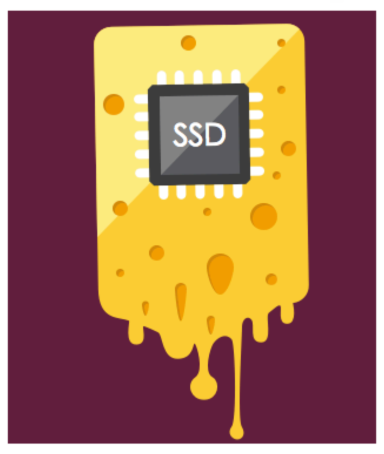
Duo Zhang, Chander Gupta, Mai Zheng
Iowa State University
Adam Manzanares, Filip Blagojevic, Cyril Guyot
Western Digital Research

FAST '20
18th USENIX Conference on File and Storage Technologies
FEBRUARY 24–27, 2020
SANTA CLARA, CA, USA
https://www.usenix.org/conference/fast20

Motivation



- **Diagnosing storage system failures is challenging**
  - Complex software layers
    - e.g., databases, file systems, software RAID, ...
  - Complex devices
    - e.g., flash-based solid state drives (SSDs),
  - Complex cross-layer interactions/dependencies
- **Example 1: ``When SSDs are not that solid'' [1]**
  - A random subset of SSD-based servers in Algolia datacenter crashed and corrupted files for unknown reasons
  - Developers "spent a big portion of two weeks just isolating machines and restoring them as quickly as possible"
  - Samsung SSDs were mistakenly blamed and blacklisted
  - A month later a ***Linux kernel bug*** was identified as root cause



- **Example 2: "Robustness of SSDs under Power Fault" [2][3]**
  - Zheng *et al.* studied behavior of SSDs under power fault
  - Bypassed file system, but still relied on the block I/O layer to apply workloads
  - Initial experiments were performed on Linux kernel v2.6.32, where 8 SSDs exhibited serialization errors
  - A follow-up work on kernel v3.16.0 shows that the number of errors on some SSDs has reduced significantly
  - A ***Linux kernel bug*** caused the different symptoms

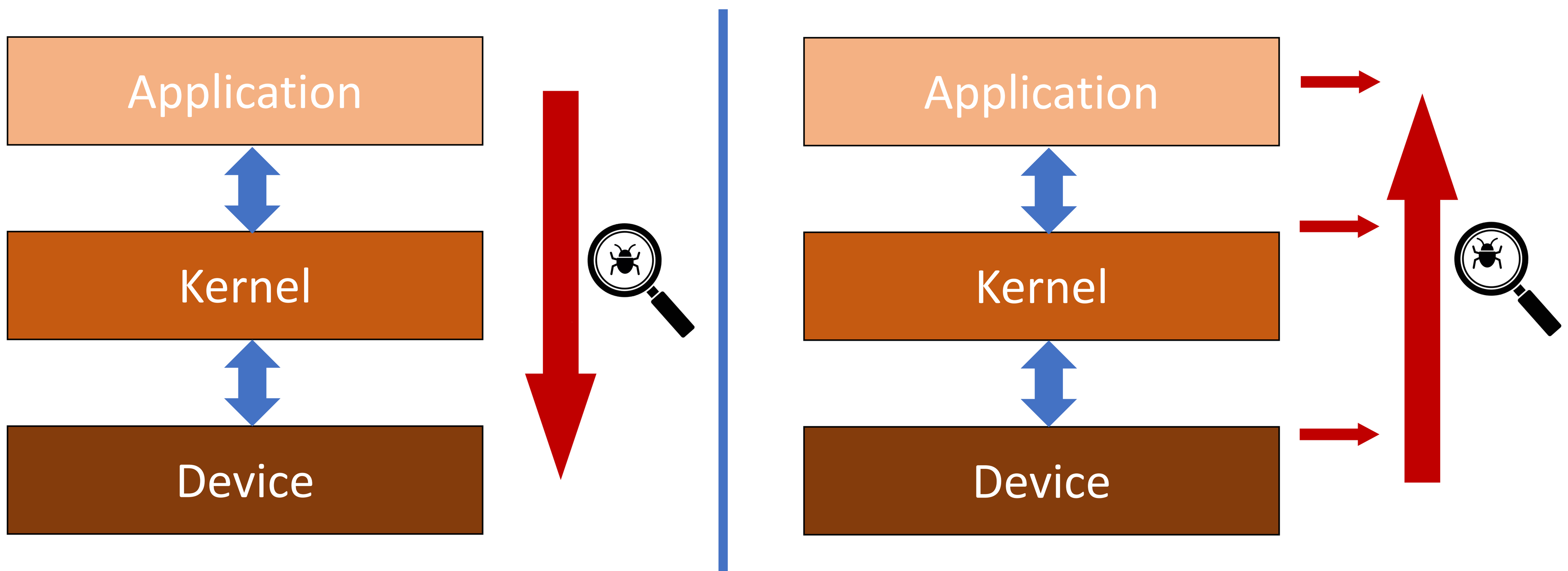
Kernel	SSD1	SSD2	SSD3	SSD4	SSD5
2.6.32	992	317	26	2	0
3.16.0	0	88	2	1	0

Table 1. SSDs behave differently under two kernels

COMMON MISTAKE

infer the behavior of devices ***indirectly*** by ***assuming that the kernel is correct***

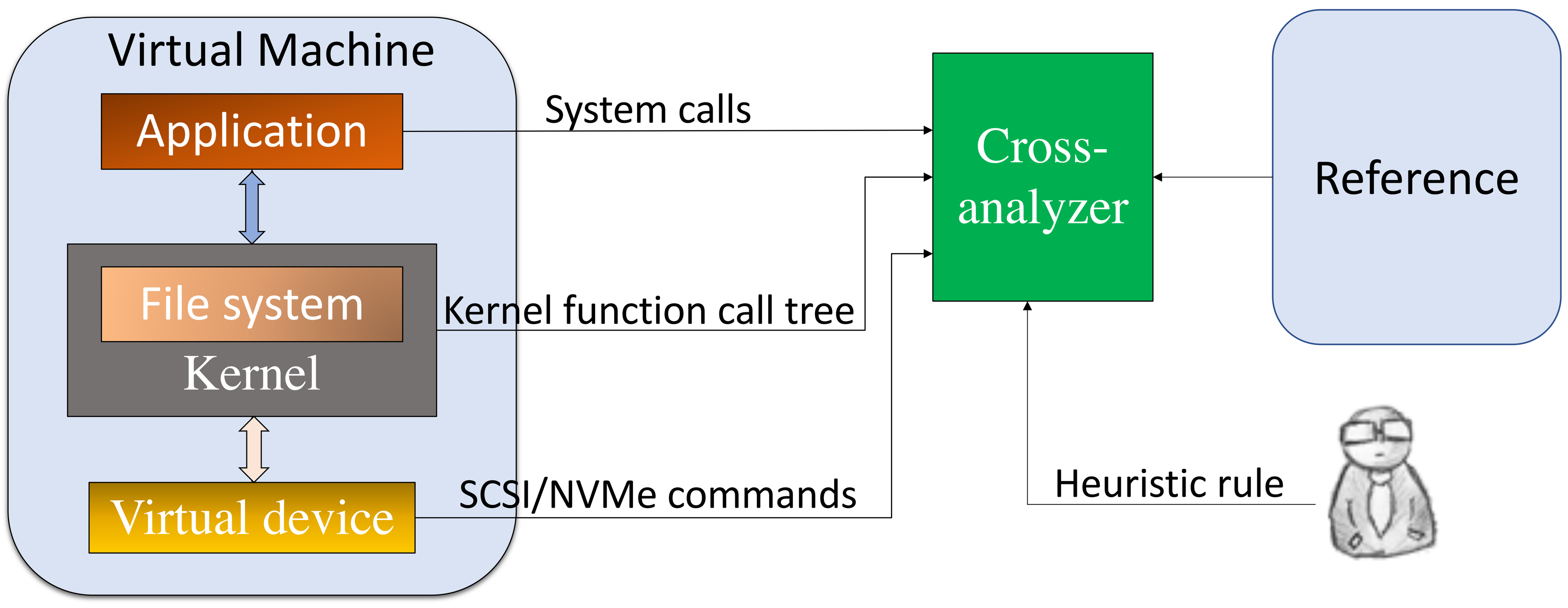
Methodology



(a) Traditional diagnosis (b) Our approach

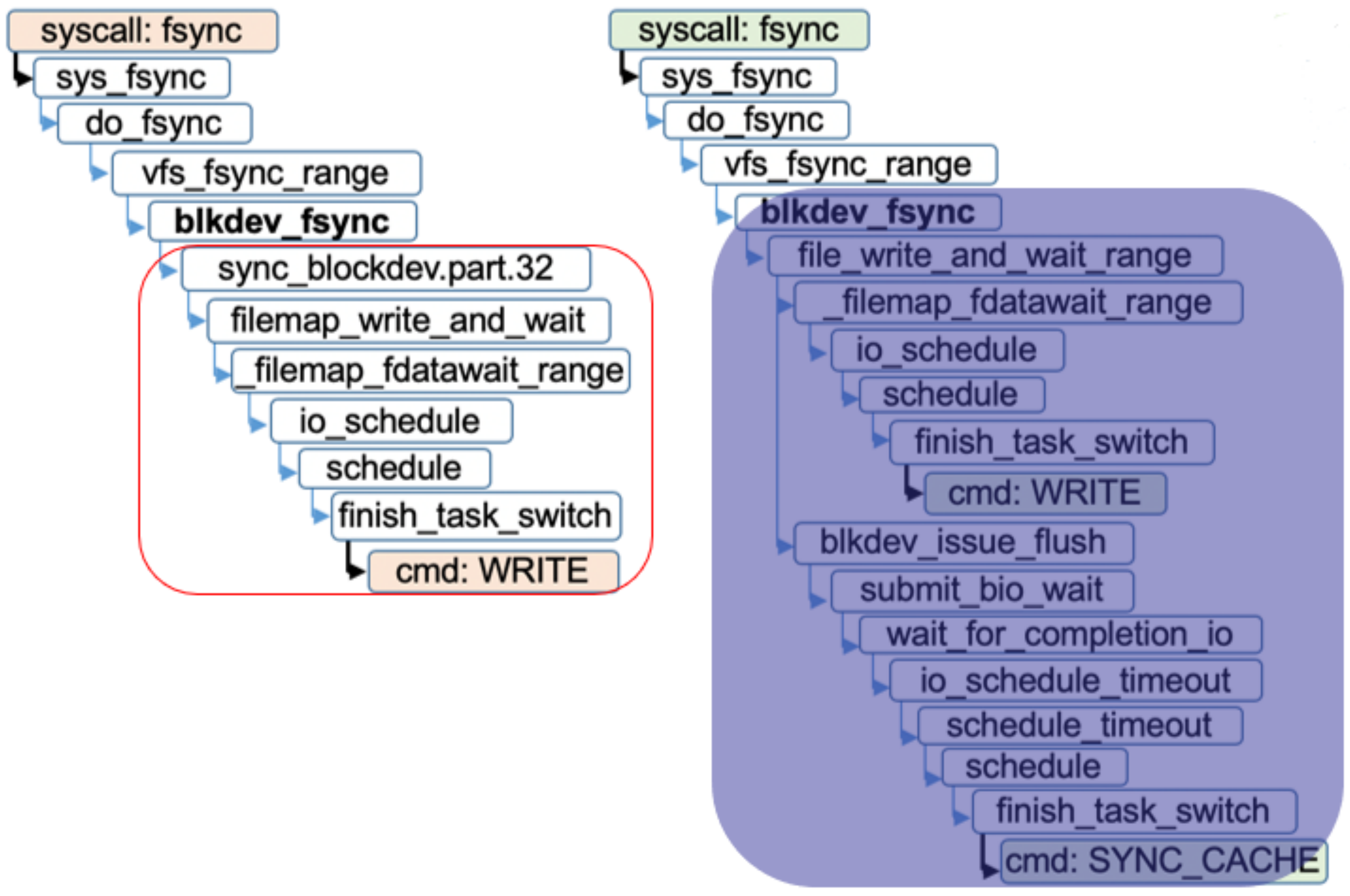
- Traditional diagnosis: a top-down approach
  - Infer the device behavior indirectly through kernel
  - trust the kernel more than the device
- **Our approach**
  - collect device-level information directly
  - leverage cross-layer differences to narrow down root cause

Prototype



- Based on QEMU virtual machine
- Cross-layer tracing & delta debugging

Preliminary Results



- Diagnose the kernel bug in [3]
  - Collect and corelate cross-layer trace
  - Pinpoint root cause by reference or heuristic rule

Acknowledgements

- This work was supported in part by NSF under grants CNS-1566554/1855565 & CCF-1717630/1853714, and by a gift from Western Digital. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of NSF

References

[1] Failures at Algolia data center: When Solid State Drives are not that solid. https://blog.algolia.com/when-solid-state-drives-are-not-that-solid/

[2] Mai Zheng, Joseph Tucek, Feng Qin, and Mark Lillibridge. Understanding the robustness of SSDs under power fault. In Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST), 2013.

[3] Mai Zheng, Joseph Tucek, Feng Qin, Mark Lillibridge, Bill W Zhao, and Elizabeth S. Yang. Reliability Analysis of SSDs under Power Fault. In Proceedings of the ACM Transactions on Computer Systems (TOCS), 2017.

IOWA STATE UNIVERSITY

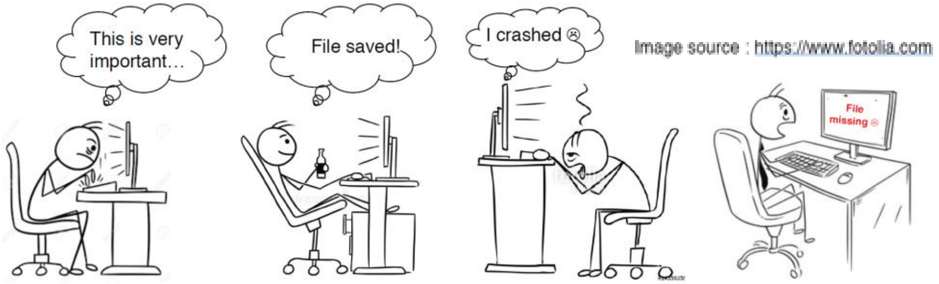
Data Storage Lab

A Cross-Layer Approach for Diagnosing Storage System Failures

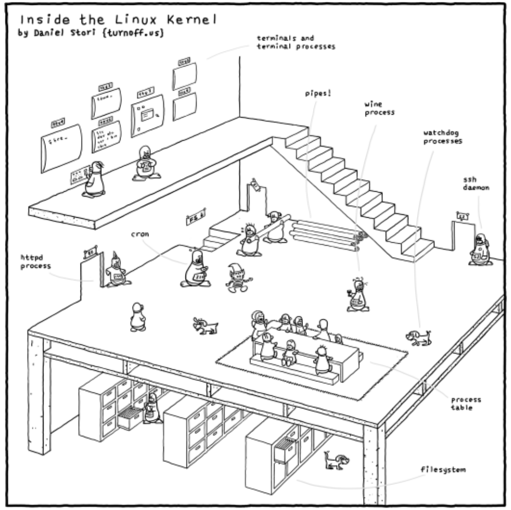
Duo Zhang, Chander Gupta, Mai Zheng
Iowa State University

Adam Manzanares, Filip Blagojevic, Cyril Guyot
Western Digital Research

Storage System Failures Are Damaging



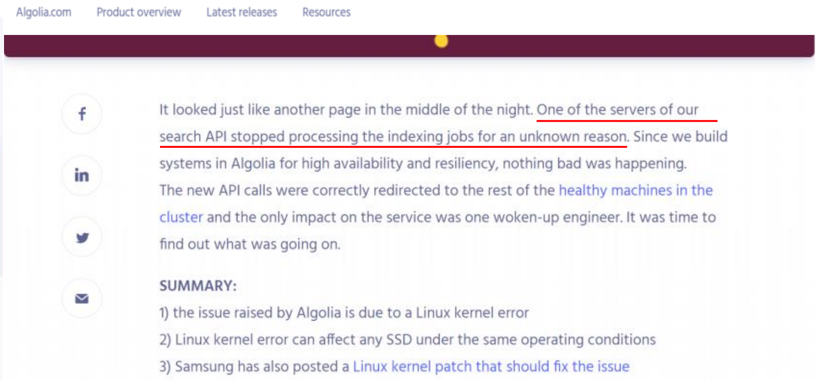
Typical Steps to Fix the Issue



From turnoff.us

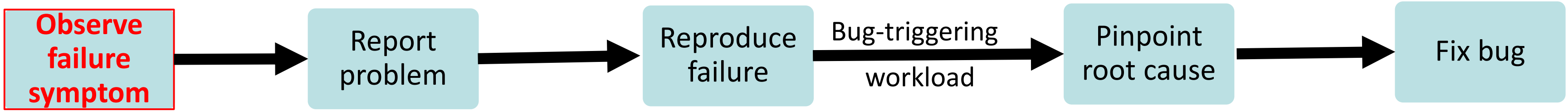
Observe failure symptom → Report problem → Reproduce failure → (Bug-triggering workload) → Pinpoint root cause → Fix bug

Typical Steps to Fix the Issue

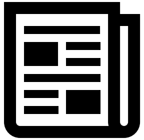

Algolia.com Product overview Latest releases Resources

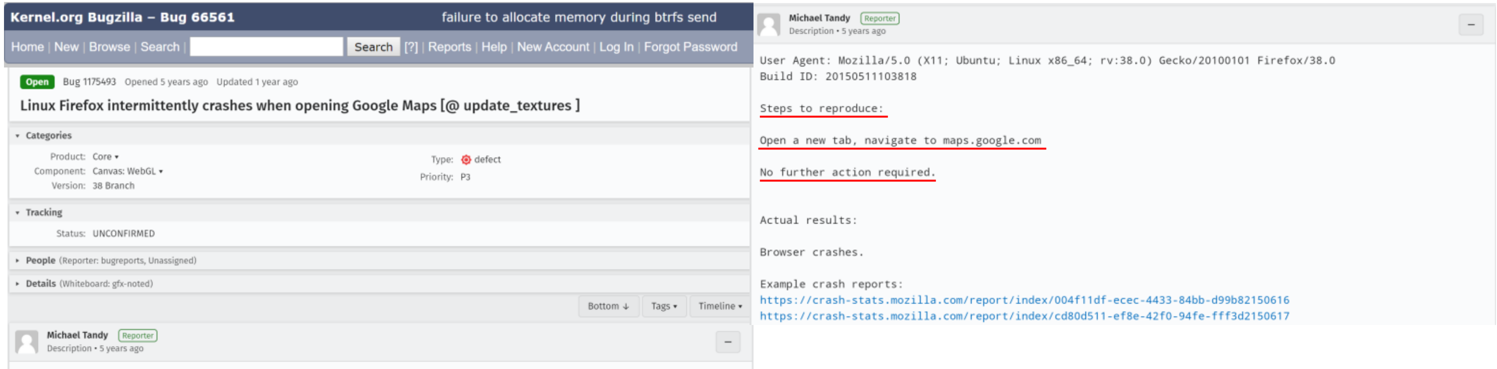
It looked just like another page in the middle of the night. One of the servers of our search API stopped processing the indexing jobs for an unknown reason. Since we build systems in Algolia for high availability and resiliency, nothing bad was happening. The new API calls were correctly redirected to the rest of the healthy machines in the cluster and the only impact on the service was one woken-up engineer. It was time to find out what was going on.

SUMMARY:
1) the issue raised by Algolia is due to a Linux kernel error
2) Linux kernel error can affect any SSD under the same operating conditions
3) Samsung has also posted a Linux kernel patch that should fix the issue



Typical Steps to Fix the Issue




Kernel.org Bugzilla – Bug 66561 failure to allocate memory during btrfs send

Home | New | Browse | Search | Search [?] | Reports | Help | New Account | Log In | Forgot Password

Open Bug 1175493 Opened 5 years ago Updated 1 year ago

Linux Firefox intermittently crashes when opening Google Maps [@ update_textures]

Categories

Product: Core
Component: Canvas: WebGL
Version: 38 Branch
Type: defect
Priority: P3

Tracking

Status: UNCONFIRMED

People (Reporter: bugreports, Unassigned)

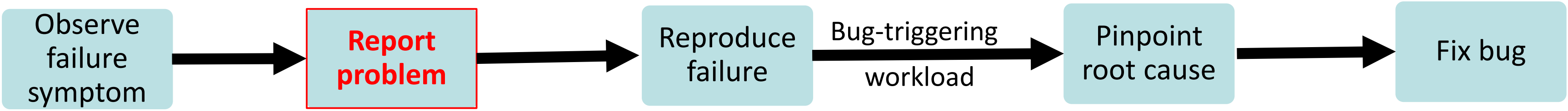
Details (Whiteboard: gfx-noted)

Bottom ↓ Tags Timeline

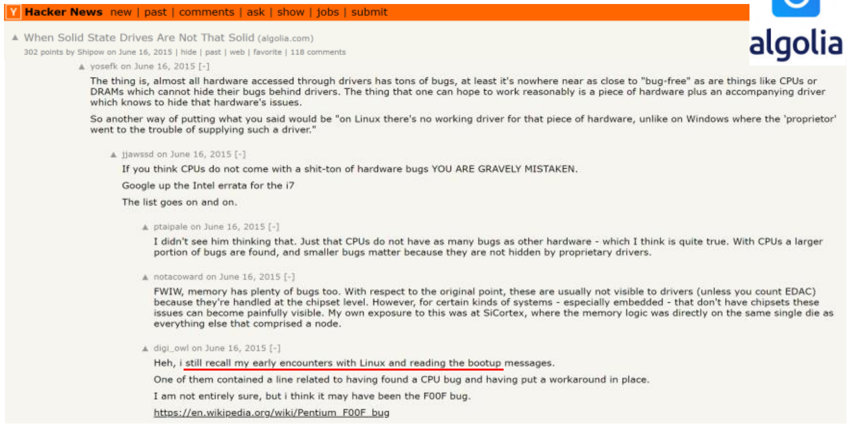
Michael Tandy Reporter
Description • 5 years ago

Michael Tandy Reporter
Description • 5 years ago

```
User Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:38.0) Gecko/20100101 Firefox/38.0
Build ID: 20150511103818

Steps to reproduce:

Open a new tab, navigate to maps.google.com

No further action required.


Actual results:

Browser crashes.

Example crash reports:
https://crash-stats.mozilla.com/report/index/004f11df-ecec-4433-84bb-d99b82150616
https://crash-stats.mozilla.com/report/index/cd80d511-ef8e-42f0-94fe-fff3d2150617
```



Typical Steps to Fix the Issue



Hacker News new | past | comments | ask | show | jobs | submit

▲ When Solid State Drives Are Not That Solid (algolia.com)
302 points by Shipow on June 16, 2015 | hide | past | web | favorite | 118 comments

▲ yosefk on June 16, 2015 [-]

The thing is, almost all hardware accessed through drivers has tons of bugs, at least it's nowhere near as close to "bug-free" as are things like CPUs or DRAMs which cannot hide their bugs behind drivers. The thing that one can hope to work reasonably is a piece of hardware plus an accompanying driver which knows to hide that hardware's issues.

So another way of putting what you said would be "on Linux there's no working driver for that piece of hardware, unlike on Windows where the 'proprietor' went to the trouble of supplying such a driver."

▲ jjawssd on June 16, 2015 [-]

If you think CPUs do not come with a shit-ton of hardware bugs YOU ARE GRAVELY MISTAKEN.

Google up the Intel errata for the i7

The list goes on and on.

▲ ptaipale on June 16, 2015 [-]

I didn't see him thinking that. Just that CPUs do not have as many bugs as other hardware - which I think is quite true. With CPUs a larger portion of bugs are found, and smaller bugs matter because they are not hidden by proprietary drivers.

▲ notacoward on June 16, 2015 [-]

FWIW, memory has plenty of bugs too. With respect to the original point, these are usually not visible to drivers (unless you count EDAC) because they're handled at the chipset level. However, for certain kinds of systems - especially embedded - that don't have chipsets these issues can become painfully visible. My own exposure to this was at SiCortex, where the memory logic was directly on the same single die as everything else that comprised a node.

▲ digi_owl on June 16, 2015 [-]

Heh, i still recall my early encounters with Linux and reading the bootup messages.

One of them contained a line related to having found a CPU bug and having put a workaround in place.

I am not entirely sure, but i think it may have been the F00F bug.

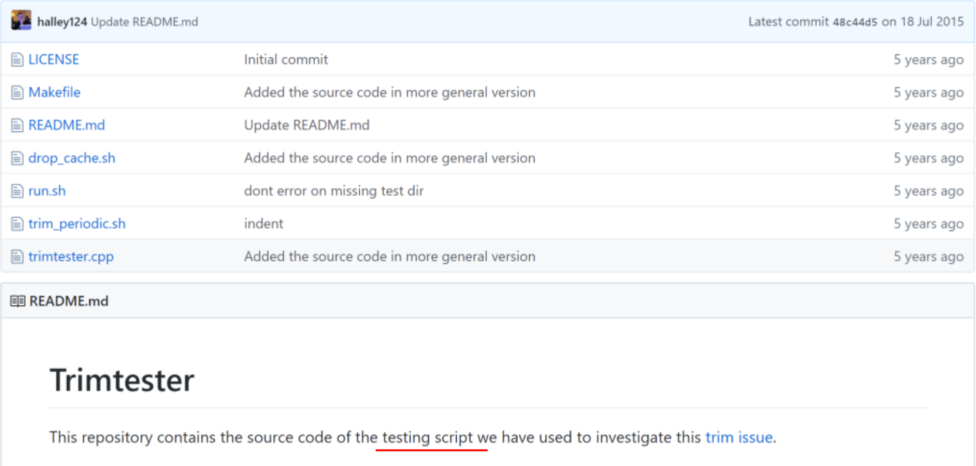
https://en.wikipedia.org/wiki/Pentium_F00F_bug

algolia

Failure may be non-deterministic.

Observe failure symptom → Report problem → **Reproduce failure** → Bug-triggering workload → Pinpoint root cause → Fix bug

Typical Steps to Fix the Issue



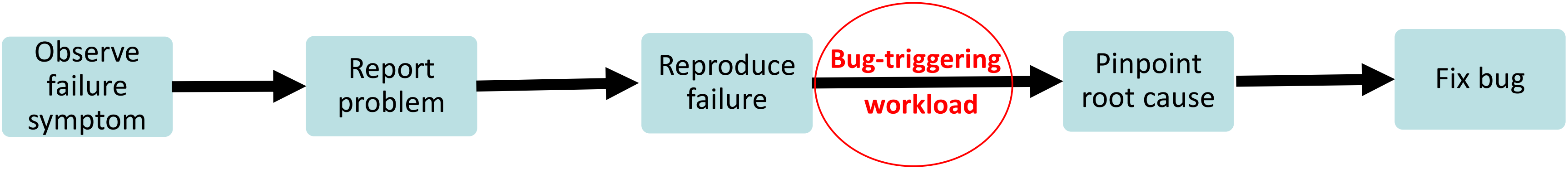
halley124 Update README.md — Latest commit 48c44d5 on 18 Jul 2015

LICENSE	Initial commit	5 years ago
Makefile	Added the source code in more general version	5 years ago
README.md	Update README.md	5 years ago
drop_cache.sh	Added the source code in more general version	5 years ago
run.sh	dont error on missing test dir	5 years ago
trim_periodic.sh	indent	5 years ago
trimtester.cpp	Added the source code in more general version	5 years ago

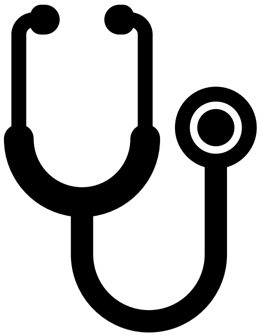
README.md

Trimtester

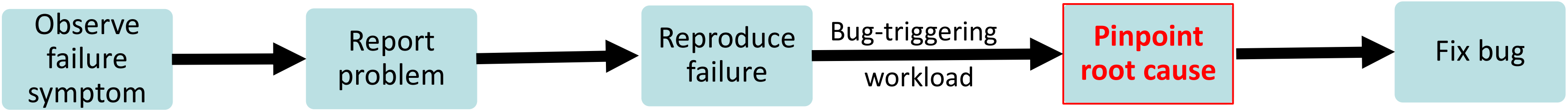
This repository contains the source code of the testing script we have used to investigate this trim issue.



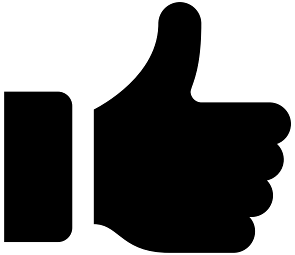
Typical Steps to Fix the Issue



```
[root@DELL-RnD-India tracing]# cat available_tracers
blk function_graph mmiotrace wakeup_rt wakeup irqsoff function sched_switch nop
[root@DELL-RnD-India tracing]# cat current_tracer
function_graph
[root@DELL-RnD-India tracing]# echo function > current_tracer
[root@DELL-RnD-India tracing]# cat current_tracer
function
[root@DELL-RnD-India tracing]# echo 1 > tracing_enabled
[root@DELL-RnD-India tracing]# cat trace > /tmp/trace.txt
[root@DELL-RnD-India tracing]# echo 0 > tracing_enabled
[root@DELL-RnD-India tracing]# cat /tmp/trace.txt | head -15
# tracer: function
#
#           TASK-PID    CPU#    TIMESTAMP  FUNCTION
#              | |       |          |         |
          <idle>-0     [002] 749252.349487: tick_program_event <-hrtimer_force_reprogram
          <idle>-0     [002] 749252.349487: tick_dev_program_event <-tick_program_event
          <idle>-0     [002] 749252.349488: ktime_get <-tick_dev_program_event
          <idle>-0     [002] 749252.349488: clockevents_program_event <-tick_dev_program_event
          <idle>-0     [002] 749252.349488: lapic_next_event <-clockevents_program_event
          <idle>-0     [002] 749252.349488: native_apic_mem_write <-lapic_next_event
          <idle>-0     [002] 749252.349489: enqueue_hrtimer <-__hrtimer_start_range_ns
          <idle>-0     [002] 749252.349489: _raw_spin_unlock_irqrestore <-__hrtimer_start_range_n
s
          <idle>-0     [002] 749252.349489: enter_idle <-cpu_idle
          <idle>-0     [002] 749252.349490: mwait_idle <-cpu_idle
          <idle>-0     [002] 749252.349490: trace_power_start.clone.0 <-mwait_idle
[root@DELL-RnD-India tracing]#
```



Typical Steps to Fix the Issue



[PATCH] raid0: data corruption when using trim

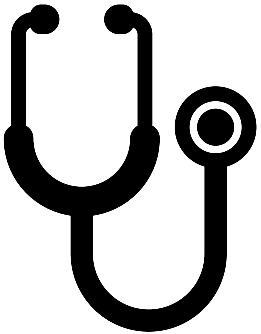
[Date Prev][Date Next][Thread Prev][Thread Next][Date Index][Thread Index]

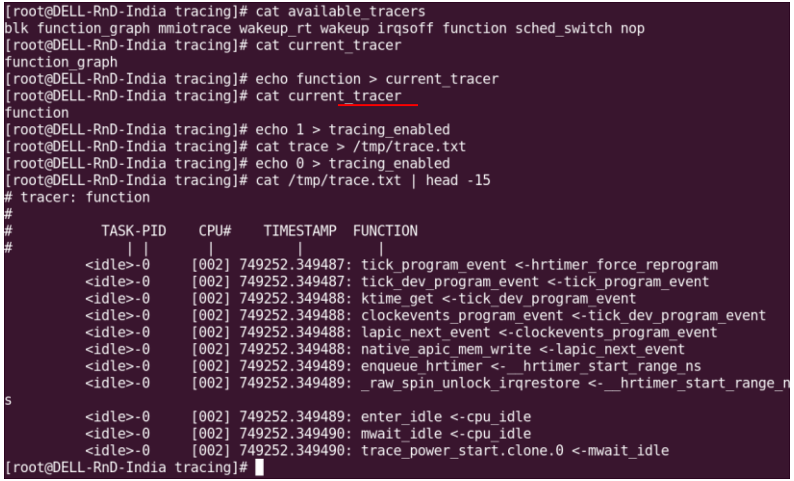
- *To*: neilb@xxxxxxx
- *Subject*: [PATCH] raid0: data corruption when using trim
- *From*: Seunguk Shin <seunguk.shin@xxxxxxxxxxx>
- *Date*: Sun, 19 Jul 2015 12:28:16 +0900
- *Cc*: linux-raid@xxxxxxxxxxxxxxx
- *Dlp-filter*: Pass
- *Thread-index*: AdDB0p+sh1bJ7soiSea9KclweqiZPg==

```
[?0001-PATCH-raid0-data-corruption-when-using-trim.patch?]

>From ca7dbe01fcd2ef2f8cea1a38de5aca5c866c585d Mon Sep 17 00:00:00 2001
From: Seunguk Shin <seunguk.shin@xxxxxxxxxxx>
Date: Sat, 18 Jul 2015 20:13:44 +0900
Subject: [PATCH] [PATCH] raid0: data corruption when using trim

When we are using software raid and tirm, there is data corruption.

The raid driver lets split bios share bio vector of source bio.
The scsi/ata driver allocates a page and stores that pointer on
bio->bi_io_vec->bv_page
(sd_setup_discard_cmnd) because the scsi/ata needs some payloads
that include start address and size of device to trim.
Because split bios share the source bio's bi_io_vec,
the pointer to the allocated page in scsi/ata driver is overwritten.

This patch splits bio vector if bio is discard.
---
block/bio.c              |  6 ------
drivers/md/raid0.c    | 13 +++++++++++++
include/linux/bio.h |  6 ++++++
3 files changed, 19 insertions(+), 6 deletions(-)
```

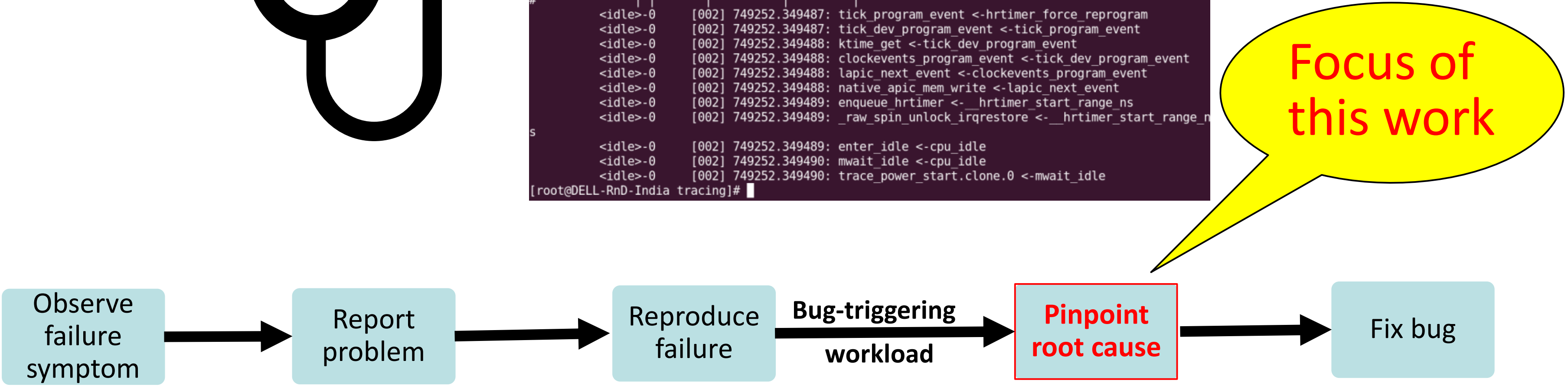
Observe failure symptom → Report problem → Reproduce failure → (Bug-triggering workload) → Pinpoint root cause → **Fix bug**

Typical Steps to Fix the Issue



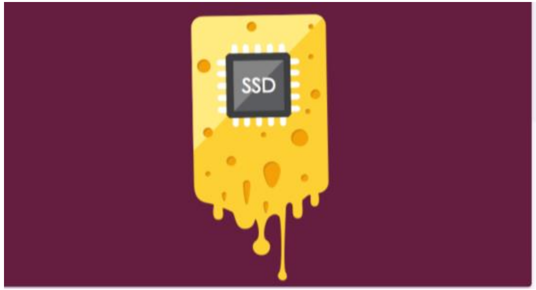


```
[root@DELL-RnD-India tracing]# cat available_tracers
blk function_graph mmiotrace wakeup_rt wakeup irqsoff function sched_switch nop
[root@DELL-RnD-India tracing]# cat current_tracer
function_graph
[root@DELL-RnD-India tracing]# echo function > current_tracer
[root@DELL-RnD-India tracing]# cat current_tracer
function
[root@DELL-RnD-India tracing]# echo 1 > tracing_enabled
[root@DELL-RnD-India tracing]# cat trace > /tmp/trace.txt
[root@DELL-RnD-India tracing]# echo 0 > tracing_enabled
[root@DELL-RnD-India tracing]# cat /tmp/trace.txt | head -15
# tracer: function
#
#           TASK-PID    CPU#    TIMESTAMP  FUNCTION
#              | |       |          |         |
          <idle>-0     [002] 749252.349487: tick_program_event <-hrtimer_force_reprogram
          <idle>-0     [002] 749252.349487: tick_dev_program_event <-tick_program_event
          <idle>-0     [002] 749252.349488: ktime_get <-tick_dev_program_event
          <idle>-0     [002] 749252.349488: clockevents_program_event <-tick_dev_program_event
          <idle>-0     [002] 749252.349488: lapic_next_event <-clockevents_program_event
          <idle>-0     [002] 749252.349488: native_apic_mem_write <-lapic_next_event
          <idle>-0     [002] 749252.349489: enqueue_hrtimer <-__hrtimer_start_range_ns
          <idle>-0     [002] 749252.349489: _raw_spin_unlock_irqrestore <-__hrtimer_start_range_n
s
          <idle>-0     [002] 749252.349489: enter_idle <-cpu_idle
          <idle>-0     [002] 749252.349490: mwait_idle <-cpu_idle
          <idle>-0     [002] 749252.349490: trace_power_start.clone.0 <-mwait_idle
[root@DELL-RnD-India tracing]#
```



Pinpointing Root Cause Is Difficult

- E.g., Algolia case:
  - The symptom **disappear** after changing to other brand SSDs
  - Samsung SSDs were **mistakenly** blamed and blacklisted
  - **A month later,** a Linux kernel bug was identified as root cause



When Solid State Drives are not that solid

Adam Surak | Jun 15th 2015 | 12 min read | Engineering

algolia BLOG Algolia.com Product overview Latest releases Resources

As a result, we informed our server provider about the affected SSDs and they informed the manufacturer. Our new deployments were switched to different SSD drives and we don't recommend anyone to use any SSD that is anyhow mentioned in a bad way by the Linux kernel. Also be careful, even when you don't enable the TRIM explicitly, at least since Ubuntu 14.04 the explicit FSTRIM runs in a cron once per week on all partitions – the freeze of your storage for a couple of seconds will be your smallest problem.

algolia BLOG Algolia.com Product overview Latest releases Resources

UPDATE July 17:

We have just finished a conference call with Samsung considering the failure analysis of this issue. Samsung engineering team has been able to successfully reproduce the issue with our latest provided binary.

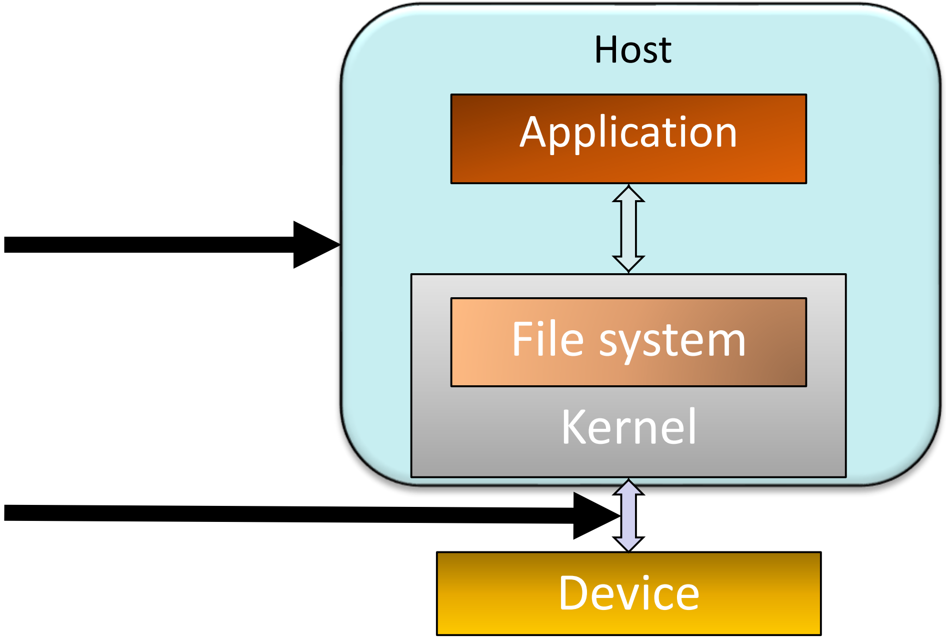
Samsung had a concrete conclusion that the issue is not related to Samsung SSD or Algolia software but is related to the Linux kernel.

Samsung has developed a kernel patch to resolve this issue and the official statement with details will be released tomorrow, July 18 on Linux community with the Linux patch guide. Our testing code is available on GitHub.

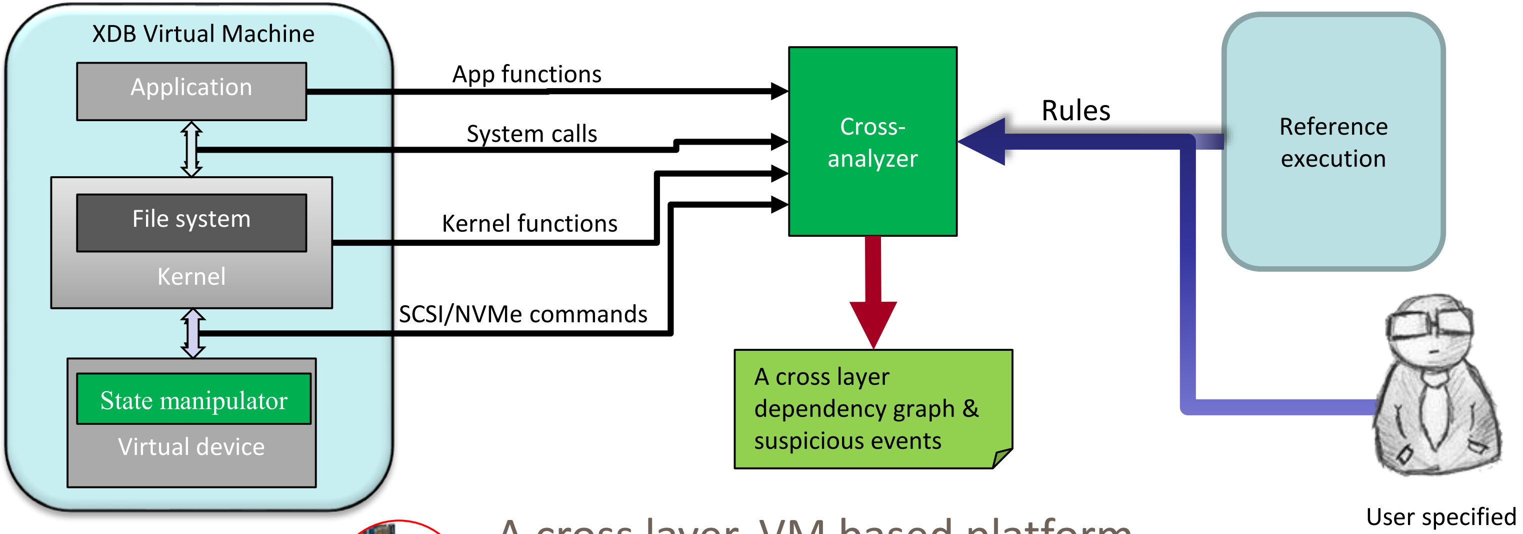
Existing Tools

- Software tools

- Hardware tools
  - E.g., the SCSI bus analyzer



Our Approach: XDB



A cross layer, VM based platform

- Support unmodified software stack
- Do not require special hardware
- Identify anomaly automatically

Key Challenges

1. How to synchronize events precisely across layers ?

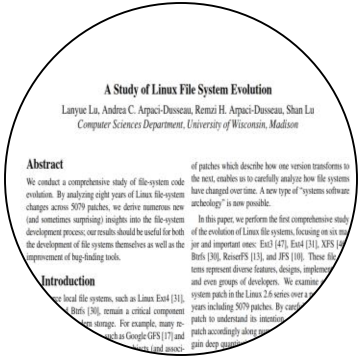
2. How to trace fine-grained functions ?

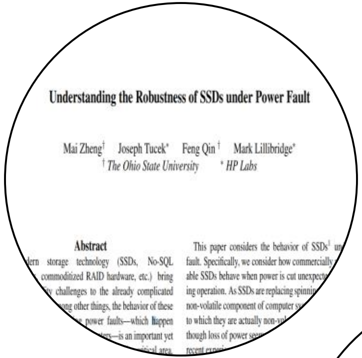
3. How to define rules ?

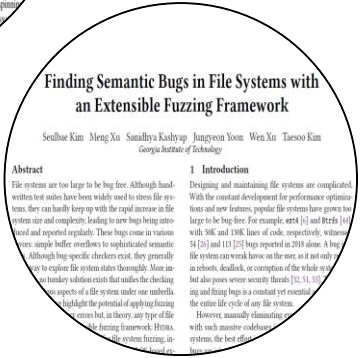
4. How to minimize disturbance and overhead ?

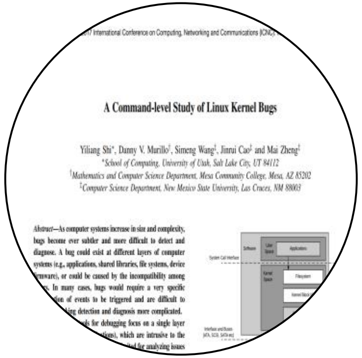
Preliminary Results

- A prototype based on QEMU
  - Trace system calls
  - Trace kernel functions (partial)
  - Trace NVMe and SCSI commands
- Reproduced failure cases from literature
- Experiment setting:
  - Intel Xeon 3.00GHz CPU
  - 16GB main memory
  - Two WD5000AAKS hard disks.
  - Ubuntu 16.04 LTS with kernel v4.4









Case Study

- Failure symptom
  - Zheng et. al. [1] studied the SSDs' behavior under power fault on Linux kernel v2.6.32
    - Applied workload & measured behavior via the block layer
  - Many serialization errors observed on different SSDs
  - The root cause is unclear

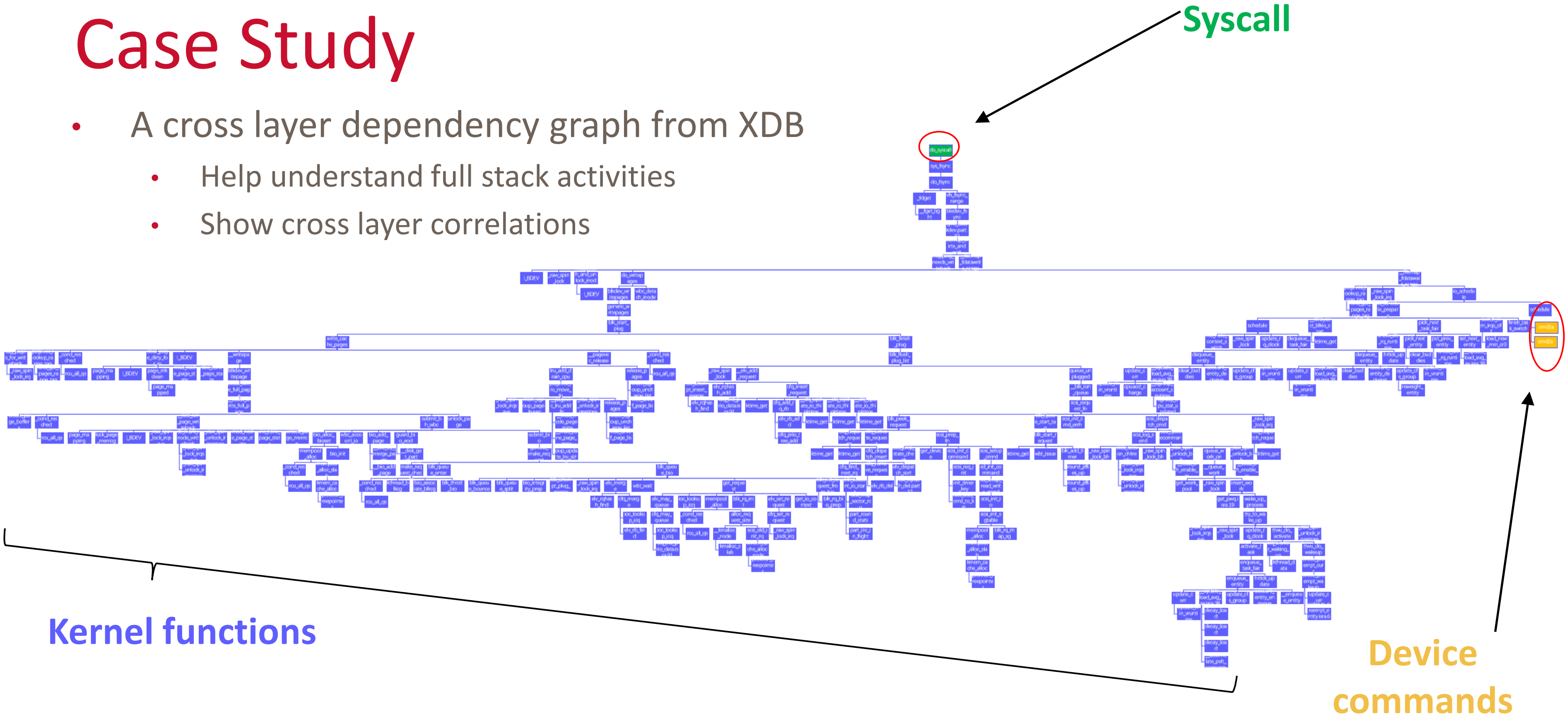
Failure	Seen?	Devices exhibiting that failure
Bit Corruption	Y	SSD#11, SSD#12, SSD#15
Flying Writes	N	-
Shorn Writes	Y	SSD#5, SSD#14, SSD#15
Unserializable Writes	Y	SSD#2, SSD#4, SSD#7, SSD#8, SSD#9, SSD#11, SSD#12, SSD#13, HDD#1
Metadata Corruption	Y	SSD#3
Dead Device	Y	SSD#1
None	Y	SSD#6, SSD#10, HDD#2

Table 5: Summary of observations. "Y" means the failure was observed with any device, while "N" means the failure was not observed.

[1] Understanding the Robustness of SSDs under Power Fault (FAST'13)

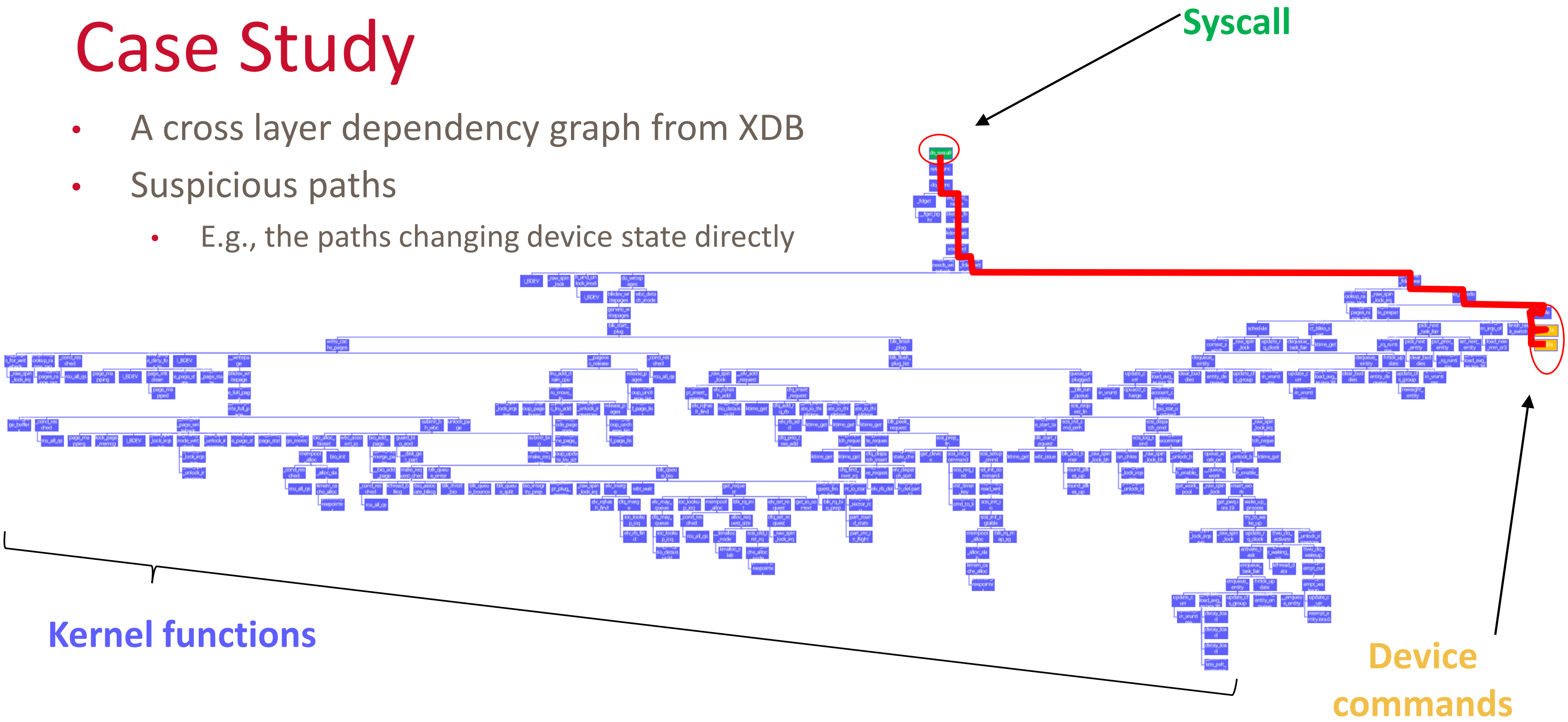
Case Study

- A cross layer dependency graph from XDB
  - Help understand full stack activities
  - Show cross layer correlations



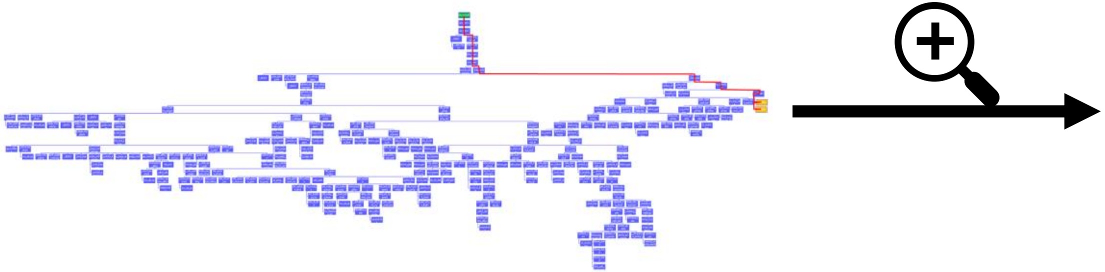
Case Study

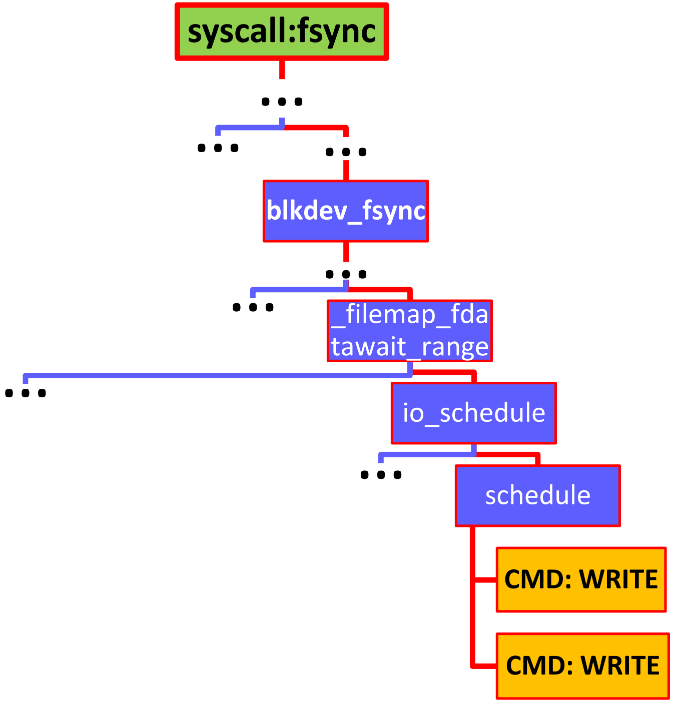
- A cross layer dependency graph from XDB
- Suspicious paths
  - E.g., the paths changing device state directly



Case Study

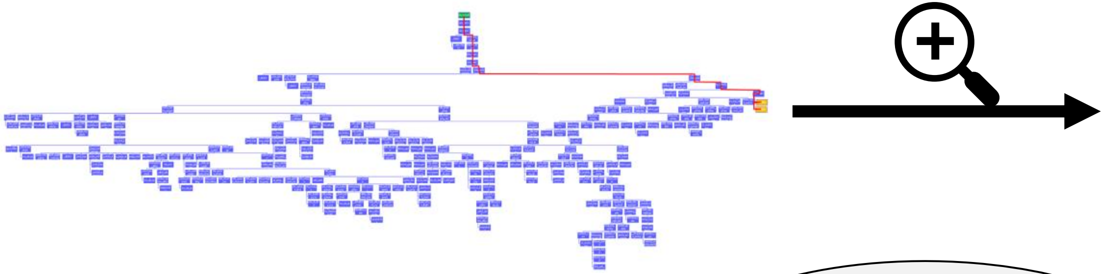
- A cross layer dependency graph from XDB
- Suspicious paths
  - E.g., the paths changing device state directly

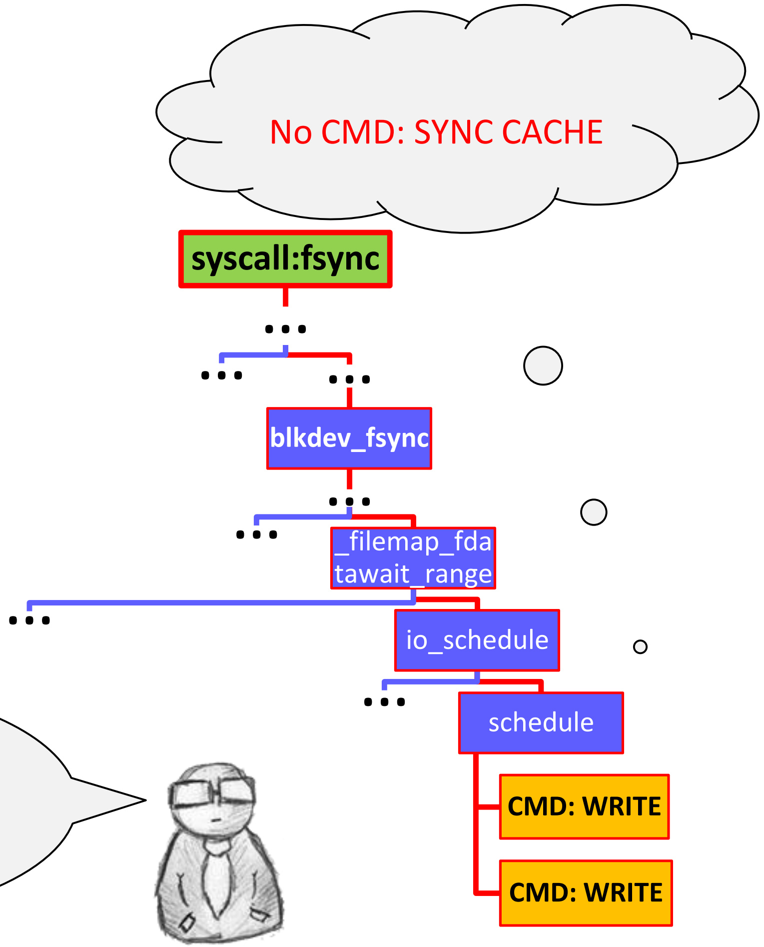




Case Study

- A cross layer dependency graph from XDB
- Suspicious paths
  - E.g., the paths changing device state directly





User specified rule:
fsync →
blkdev_fsync() →
CMD: SYNC CACHE

Case Study

- Recap failure symptom
  - Zheng et. al. [1] studied SSDs' behavior under power fault on kernel **v2.6.32**
    - Applied workload & measured behavior via the block layer
  - Many serialization errors observed on different SSDs
  - The root cause is unclear

Failure	Seen?	Devices exhibiting that failure
Bit Corruption	**Y**	SSD#11, SSD#12, SSD#15
Flying Writes	N	-
Shorn Writes	**Y**	SSD#5, SSD#14, SSD#15
Unserializable Writes	**Y**	SSD#2, SSD#4, SSD#7, SSD#8, SSD#9, SSD#11, SSD#12, SSD#13, HDD#1
Metadata Corruption	**Y**	SSD#3
Dead Device	**Y**	SSD#1
None	**Y**	SSD#6, SSD#10, HDD#2

Table 5: Summary of observations. "Y" means the failure was observed with any device, while "N" means the failure was not observed.

[1] Understanding the Robustness of SSDs under Power Fault (FAST'13)

Case Study

- Recap failure symptom
  - Zheng et. al. [1] studied SSDs' behavior under power fault on kernel **v2.6.32**
    - Applied workload & measured behavior via the block layer
  - Many serialization errors observed on different SSDs
  - The root cause is unclear

Failure	Seen?	Devices exhibiting that failure
Bit Corruption	Y	SSD#11, SSD#12, SSD#15
Flying Writes	N	-
Shorn Writes	Y	SSD#5, SSD#14, SSD#15
Unserializable Writes	Y	SSD#2, SSD#4, SSD#7, SSD#8, SSD#9, SSD#11, SSD#12, SSD#13, HDD#1
Metadata Corruption	Y	SSD#3
Dead Device	Y	SSD#1
None	Y	SSD#6, SSD#10, HDD#2

Table 5: Summary of observations. "Y" means the failure was observed with any device, while "N" means the failure was not observed.

[1] Understanding the Robustness of SSDs under Power Fault (FAST'13)

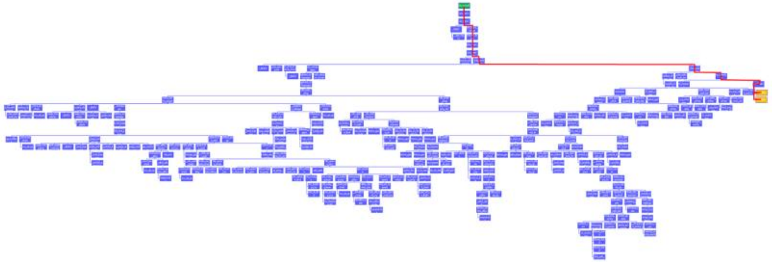
- New symptom
  - Zheng et. al. [2] repeated the experiment on kernel **v3.16.0**
  - The no. of serialization errors reduce on the new kernel

Kernel	Number of serialization errors				
	SSD1	SSD2	SSD3	SSD4	SSD5
v2.6.32	992	317	26	2	0
v3.16.0	0	88	2	1	0

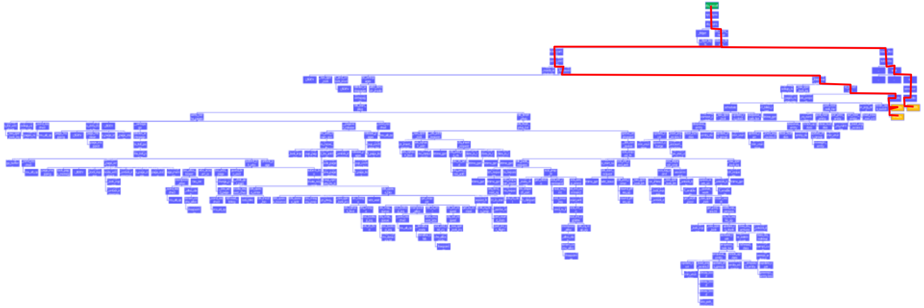
[2] Reliability Analysis of SSDs under Power Fault (TOCS'17)

Case Study

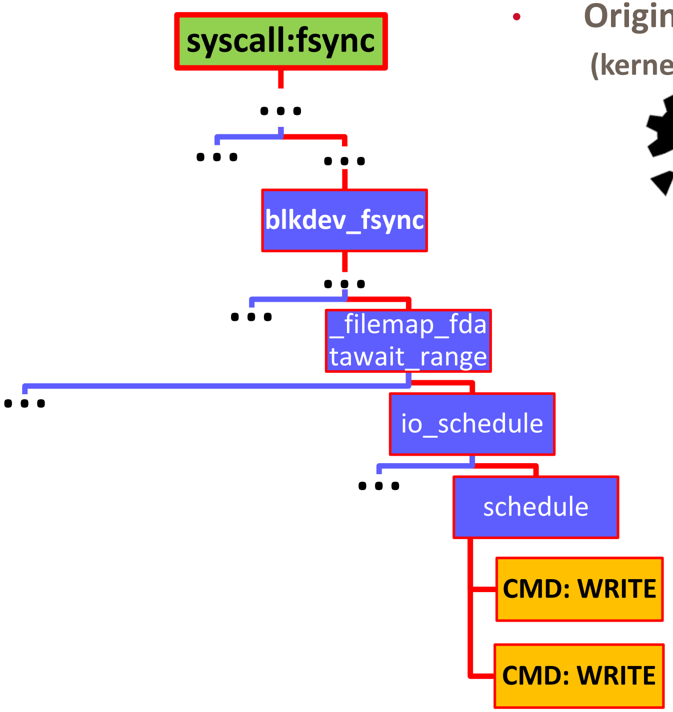
- **Original graph (kernel v2.6.32)**



- **Reference graph (kernel v3.16.0)**

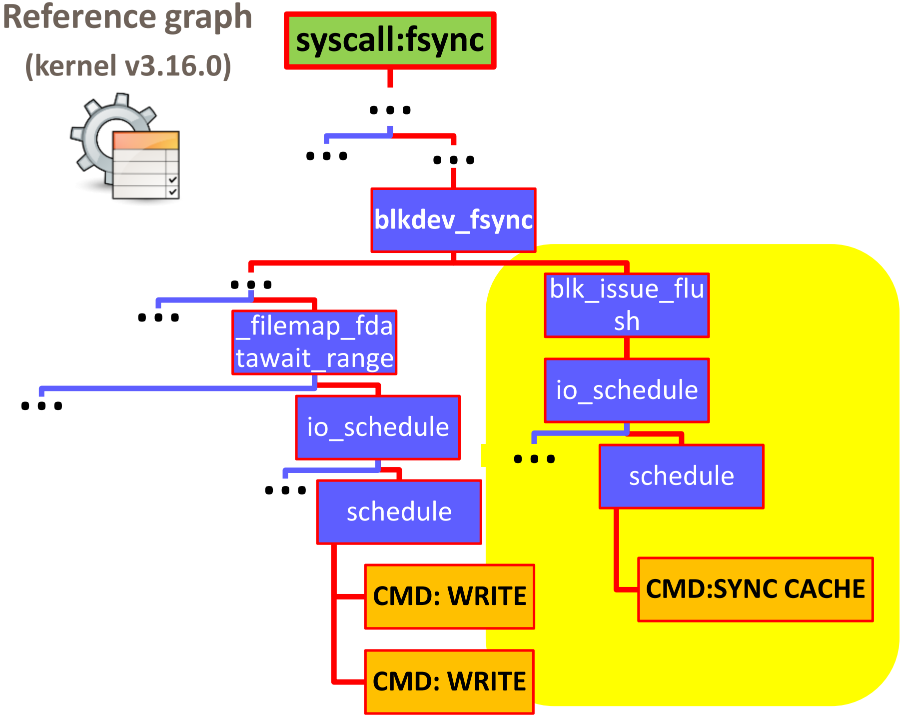


Case Study



- Original graph
(kernel v2.6.32)

- Reference graph
(kernel v3.16.0)



Thanks !

Questions ?