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

1. Observe failure symptom
2. Report problem
3. Reproduce failure
4. Bug-triggering workload
5. Pinpoint root cause
6. Fix bug
Typical Steps to Fix the Issue

1. Observe failure symptom
2. Report problem
3. Reproduce failure
4. Bug-triggering workload
5. Pinpoint root cause
6. Fix bug

When Solid State Drives are not that solid

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

1. **Observe failure symptom**
2. **Report problem**
3. **Reproduce failure**
4. **Bug-triggering workload**
5. **Pinpoint root cause**
6. **Fix bug**
Typical Steps to Fix the Issue

1. Observe failure symptom
2. Report problem
3. Reproduce failure
4. Bug-triggering workload
5. Pinpoint root cause
6. Fix bug

Failure may be non-deterministic.
Typical Steps to Fix the Issue

1. Observe failure symptom
2. Report problem
3. Reproduce failure
4. Pinpoint root cause
5. Fix bug

Bug-triggering workload

Trimitster

This repository contains the source code of the testing script we used to investigate this trim issue.
Typical Steps to Fix the Issue

1. Observe failure symptom
2. Report problem
3. Reproduce failure
4. Pinpoint root cause
5. Bug-triggering workload
6. Fix bug
Typical Steps to Fix the Issue

1. **Observe failure symptom**
2. **Report problem**
3. **Reproduce failure**
4. **Bug-triggering workload**
5. **Pinpoint root cause**
6. **Fix bug**

---

**[PATCH] raid0: data corruption when using trim**

- **To:** neilb@xxxxxx
- **Subject:** [PATCH] raid0: data corruption when using trim
- **From:** Seunguk Shin <seunguk.shin@xxxxxxxxxx>
- **Date:** Sun, 19 Jul 2015 12:28:16 +0900
- **Cc:** linux-raid@xxxxxxxxxx
- **Dlp-filter:** Pass
- **Thread-index:** AdDB0p+sh1bj7soiSea9KclweqiZPg==

---

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

The raid driver lets split bias 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_cmd) because the scsi/ata needs some payloads that include start address and size of device to trim. Because split bias 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
drivers/md/raid0.c | 6       include/linux/bio.h | 6 +++++++
3 files changed, 19 insertions(+), 6 deletions(-)
Typical Steps to Fix the Issue

1. Observe failure symptom
2. Report problem
3. Reproduce failure
4. Bug-triggering workload
5. Pinpoint root cause
6. Fix bug

Focus of this work

Data Storage Lab
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
Existing Tools

- Software tools
  - Perf
  - Strace
  - Kprobe
  - Dtrace
  - Systemtap
  - Ftrace
  - LTTng
  - KGTP

- 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

A cross layer dependency graph & suspicious events

Reference execution

User specified

Data Storage Lab
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

<table>
<thead>
<tr>
<th>Failure</th>
<th>Seen?</th>
<th>Devices exhibiting that failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Corruption</td>
<td>Y</td>
<td>SSD#11, SSD#12, SSD#15</td>
</tr>
<tr>
<td>Flying Writes</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>Short Writes</td>
<td>Y</td>
<td>SSD#5, SSD#14, SSD#15</td>
</tr>
<tr>
<td>Unserializable Writes</td>
<td>Y</td>
<td>SSD#2, SSD#4, SSD#7, SSD#8, SSD#9, SSD#11, SSD#12, SSD#13, HDD#1</td>
</tr>
<tr>
<td>Metadata Corruption</td>
<td>Y</td>
<td>SSD#3</td>
</tr>
<tr>
<td>Dead Device</td>
<td>Y</td>
<td>SSD#1</td>
</tr>
<tr>
<td>None</td>
<td>Y</td>
<td>SSD#6, SSD#10, HDD#2</td>
</tr>
</tbody>
</table>

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

No 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

<table>
<thead>
<tr>
<th>Failure</th>
<th>Seen?</th>
<th>Devices exhibiting that failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Corruption</td>
<td>Y</td>
<td>SSD#11, SSD#12, SSD#15</td>
</tr>
<tr>
<td>Flying Writes</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Short Writes</td>
<td>Y</td>
<td>SSD#5, SSD#14, SSD#15</td>
</tr>
<tr>
<td>Unserializable Writes</td>
<td>Y</td>
<td>SSD#2, SSD#4, SSD#7, SSD#8, SSD#9, SSD#11, SSD#12, SSD#13, HDD#1</td>
</tr>
<tr>
<td>Metadata Corruption</td>
<td>Y</td>
<td>SSD#3</td>
</tr>
<tr>
<td>Dead Device</td>
<td>Y</td>
<td>SSD#1</td>
</tr>
<tr>
<td>None</td>
<td>Y</td>
<td>SSD#6, SSD#10, HDD#2</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Failure</th>
<th>Seen?</th>
<th>Devices exhibiting that failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Corruption</td>
<td>Y</td>
<td>SSD#11, SSD#12, SSD#15</td>
</tr>
<tr>
<td>Flying Writes</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Shorn Writes</td>
<td>Y</td>
<td>SSD#5, SSD#14, SSD#15</td>
</tr>
<tr>
<td>Unserializable Writes</td>
<td>Y</td>
<td>SSD#2, SSD#4, SSD#7, SSD#8, SSD#9, SSD#11, SSD#12, SSD#13, HDD#1</td>
</tr>
<tr>
<td>Metadata Corruption</td>
<td>Y</td>
<td>SSD#3</td>
</tr>
<tr>
<td>Dead Device</td>
<td>Y</td>
<td>SSD#1</td>
</tr>
<tr>
<td>None</td>
<td>Y</td>
<td>SSD#6, SSD#10, HDD#2</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Kernel</th>
<th>SSD1</th>
<th>SSD2</th>
<th>SSD3</th>
<th>SSD4</th>
<th>SSD5</th>
</tr>
</thead>
<tbody>
<tr>
<td>v2.6.32</td>
<td>992</td>
<td>317</td>
<td>26</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>v3.16.0</td>
<td>0</td>
<td>88</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

**syscall: fsync**

- **blkdev fsync**
  - _filemap_fda tawait_range
  - io_schedule
  - schedule
    - CMD: WRITE
    - CMD: WRITE

- **syscall: fsync**
  - **blkdev fsync**
    - _filemap_fda tawait_range
    - io_schedule
      - CMD: WRITE
      - CMD: WRITE
      - CMD: SYNC CACHE
Thanks !
Questions ?