SOS : Software-based Out-of-Order Scheduling for High-Performance NAND Flash-Based SSDs

Sangwook Shane Hahn, Sungjin Lee and Jihong Kim

Computer Architecture & Embedded Systems Laboratory
School of Computer Science and Engineering
Seoul National University

Massive Storage Systems and Technologies 2013 (MSST 2013)
May 9th, 2013
Introduction

- NAND flash memory based devices
  - Become more popular because of their performance
  - Consist of multiple flash chips
    - Each chip can perform only one flash operation at a time
- In order to increase the performance of NAND-based devices
  - Exploiting multichip parallelism is a key
  - Out-of-order execution model is ideal for multichip parallelism

![Diagram showing performance improvement through out-of-order execution model.](image)

MSST’13
Out-of-order Support in SSDs

- **Hardware-based Out-of-Order Scheduling (HOS)**
  - Receive requests with *only physical address* information translated by a flash translation layer (FTL)
  - Execute requests in an out-of-order manner

Logical address information

File System

- W (LBA0)
- E (LBA1)
- W (LBA2)
- R (LBA4)
- W (LBA5)
- W (LBA0)

Both Logical & Physical address information

- LBA
- FTL
- PBA

W (LBA0 -> PBA0)

NAND flash device

- Physical address information
  - W (PBA2)
  - E (PBA1)
  - W (PBA0)
  - W (PBA5)
  - R (PBA4)
  - W (PBA7)

HOS

Data buffer

Flash Memory

MSST’13
HOS Weakness #1: *Skewed Queue Problem*

- Data locality & different operation latencies **induce** the skewed queue problem.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Bonnie++</th>
<th>Postmark</th>
<th>Financial1</th>
<th>Financial2</th>
<th>Websearch</th>
</tr>
</thead>
<tbody>
<tr>
<td># of reallocatable writes</td>
<td>29%</td>
<td>32%</td>
<td>18%</td>
<td>11%</td>
<td>9%</td>
</tr>
<tr>
<td># of total writes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In order to **reallocate requests**, mapping table update process is inevitable.

**How to balance skewed queues?**

Modifying mapping table is **hard** to hardware-based scheduler and **easy** to software-based one.

When at least one of chips is idle.
HOS Weakness #2: **Useless Write Problem**

- **Useless Writes** means *overwrites* at the data buffer.
- HOS can’t recognize useless writes without logical address.

### Benchmarks:

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Bonnie++</th>
<th>Postmark</th>
<th>Financial1</th>
<th>Financial2</th>
<th>Websearch</th>
</tr>
</thead>
<tbody>
<tr>
<td># of overwrites/# of total writes</td>
<td>11.7%</td>
<td>14.3%</td>
<td>17.6%</td>
<td>9.2%</td>
<td>7.1%</td>
</tr>
</tbody>
</table>

In order to *cancel useless writes*,

logical address information of requests is essential.

*Access logical address information of request is hard to hardware-based scheduler and easy to software-based one.*
Our Contributions

- Propose software-based out-of-order scheduling (SOS)
  - SOS can overcome the skewed queue problem & useless write problem without additional hardware resources and high design cost

- SOS was implemented at a prototype SSD, BlueSSD
  - SOS improves the average I/O response time by up to 42% over HOS
Overview of SOS

- SOS handles requests at the software queues with logical & physical address information
  
  - Queue size leveler: detect the skewed queues and then rearranges requests
  
  - Write hit manager: eliminate useless writes by canceling unnecessary writes
Queue Size Leveler (QSL)

- Balance the size of multiple I/O queues by reallocating write requests to idle chips
  - Consider different latencies of each flash operations
  - Triggered when one of chips become idle
Write Hit Manager (WHM)

- Detect overwrites and cancel them to eliminate unnecessary writes and invalidations
  - Additional flag at mapping table implemented for detection
  - Detect useless writes without full search

Flag 0 means “Previous write request still exists at data buffer”

Overwrite occurs & it triggers WHM

MSST’13
Experimental Settings

- We implemented the SOS in SSD prototype, BlueSSD
  - BlueSSD supports 4 buses and 4 ways (Total 16 chips)
  - PowerPC 405 processor (@100Mhz) on BlueSSD runs Linux 2.6.25.3 kernel

- Realize HOS by rearranging the sequence of requests according to the out-of-order scheduling algorithm
  - The rearranged I/O traces were replayed, using the in-order scheduling algorithm
Experimental Results

- Characteristics of benchmarks

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Bonnie++</th>
<th>Postmark</th>
<th>Financial1</th>
<th>Financial2</th>
<th>Websearch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Ratio</td>
<td>52.1%</td>
<td>50.0%</td>
<td>32.8%</td>
<td>82.4%</td>
<td>91.1%</td>
</tr>
<tr>
<td>Write Ratio</td>
<td>47.9%</td>
<td>50.0%</td>
<td>67.2%</td>
<td>17.6%</td>
<td>8.9%</td>
</tr>
</tbody>
</table>

- SOS improves I/O response times by 15% to 42% over HOS
Conclusion & Future Work

- Software-based out-of-order scheduling
  - Exploits the multichip parallelism more effectively than hardware-based one
    - Queue size leveler addresses skewed queue problem
    - Write hit manager addresses useless write problem
  - Improves I/O response times by up to 42% over HOS

- Future work
  - More flexible request scheduling techniques
    - Reflect user-priority of requests from upper layer, etc.
Thank you