Exploring System Challenges of Ultra-Low Latency Solid State Drives

Sungjoon Koh
Changrim Lee, Miryeong Kwon, and Myoungsoo Jung

Computer Architecture and Memory systems Lab
Executive Summary

Motivation. Ultra-low latency (ULL) is emerging, but not characterized by far.

Contributions.
- Characterizing the performance behaviors of ULL SSD.
- Studying several system-level challenges of the current storage stack.

Key Observations.
- ULL SSD minimizes the I/O interferences (interleaving reads and writes).
- NVMe queue mechanisms are required to be optimized for ULL SSDs.
- Polling-based I/O completion routine isn’t effective for current NVMe SSDs.
Architectural Change of SSD

- **CPU**
- **MCH (North Bridge)**
  - **PCI Express**
  - **Direct Access**
  - **DRAM**
- **ICH (South Bridge)**
  - **SATA SSD**
- **NVMe SSD**
- **High bandwidth**

**Notes:**
- PCI Express
- Direct Access
- DRAM
- SATA SSD
Evolution of SSDs

Bandwidth almost reaches the maximum performance.

Still, long latency (far from DRAM)

New flash memory, called “Z-NAND”
New Flash Memory

Existing 3D NAND

Read: 45-120 $\mu$s
Write: 660-5000 $\mu$s

Z-NAND [1]

Read: $3\mu s$ (15~20x)
Write: $100\mu s$ (6~7x)

Z-NAND based archives “Z-SSD”

Z-NAND [1]

<table>
<thead>
<tr>
<th>Technology</th>
<th>SLC based 3D NAND 48 stacked word-line layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>64Gb</td>
</tr>
<tr>
<td>Page Size</td>
<td>2kB/Page</td>
</tr>
</tbody>
</table>
Characterization Categories

Performance Analysis.
- Average latency.
- Long-tail latency.
- Bandwidth.
- I/O interference impact.

Polling vs. Interrupt
- Overall latency comparison.
- CPU utilization analysis.
- Memory requirement.
- Five-nines latency.
Evaluation Settings

**OS:** Linux 4.14.10

**CPU:** Intel® Core™ i7-4790K (4-core, 4.00GHz)

**Memory:** DDR4 DRAM (16GB)

**SSD**
- **ULL SSD:** Z-SSD Prototype (800GB)
- **NVMe SSD:** Intel® SSD 750 Series (400GB)

**Benchmark:** Flexible I/O Tester (FIO v2.99)
Performance Analysis
Overview

Host

Request Queue

Increase queue depth

Rd Wr Rd Wr Rd Wr Rd Wr

NVMe Driver

NVMe Controller

SSD

① **Average** latency & **Long-tail** latency

② **Bandwidth**

③ **Read latency** under **Read & Write intermixed** workload
Average Latency of ULL SSD

- Sequential Write
- Sequential Read
- Random Read
- Random Write

Sequential Read
Random Read
Sequential Write
Random Write

4KB DMA = 8μs ($t_R = 3μs$)

"Split-DMA & Super-Channel"

1.8x
5.1x

$t_R + t_{DMA}$

$11μs$
Split-DMA & Super-Channel

Reference: Cheong, Woosung et al., “A flash memory controller for 15μs ultra-low-latency SSD using high-speed 3D NAND flash with 3μs read time”, ISSCC, 2018
Long-tail Latency of ULL SSD

Resource conflict
Insufficient internal buffer, Internal tasks

“Split DMA” & “Suspend/Resume”
Suspend/Resume DMA Technique

Way 1

DMA (for write request)

Way 2

Wait

$t_R$ CMD

Data Out

Reduce read latency & Increase QoS

Suspend/Resume [1]

Suspend
Resume

Way 1

Read

DMA (for write request)

Way 2

$t_R$ CMD

Data Out

Reference: Cheong, Woosung et al., “A flash memory controller for 15μs ultra-low-latency SSD using high-speed 3D NAND flash with 3μs read time”, ISSCC, 2018
Flush operation / meta data writes in file system are intermixed with user requests. I/O Interference

Great **performance bottleneck** of conventional SSDs.

How about ULL SSD?

ULL SSD can be applied to **real-life storage stack** w/o performance degradation.

Remains almost constant → “Suspend/resume”, … [1]
Queue Analysis

Only 50% of Max BW
Almost Max BW

I/O request *rescheduling* within queue.

Light queue mechanisms (ex. NCQ) are *not sufficient.*
→ Requires *rich queue mechanism*

Well-aligned with light queue mechanisms (ex. NCQ).
NVMe needs to be *lightened*
Polling vs. Interrupt

Two different I/O completion methods
Interrupt / Polling

Systems with **short waiting time** adopts polling-based waiting strategy. (even though it incurs **lots of overheads**)

For example, **“spin lock”, “network message passing”** applies polling-based waiting strategy.

Polling is currently implemented to NVMe storage stack.

**Does it really need for current NVMe SSDs?**
Interrupt / Polling

Interrupt.

Submit request → Sleep → Complete request

CS

Shorter

Larger portion

Low latency

ISR

① Finishes

NVMe Controller

Gain

Polling.

Submit request → Polling → Complete request

CS

Done??

Gain

SSD

Command Execution

② Raise IRQ

③ Wake

① Finishes
Overall Performance

Future *lower latency SSD* can achieve remarkable performance improvement with polling-based I/O completion routine.
**System Challenges**

### Memory Bound

- **Memory bound** = Fraction of slots where pipeline could be stalled due to load/store.

- **High** memory bound = **Frequent** memory access

### Polling-based I/O Services

- **Significant system overhead**
- Needs to be addressed

### CPU Utilization

- High CPU utilization

###Latency (msec)**

- ULL Write
  - Interrupt
  - Polling

<table>
<thead>
<tr>
<th>4KB</th>
<th>8KB</th>
<th>16KB</th>
<th>32KB</th>
<th>4.2</th>
<th>4.3</th>
<th>4.4</th>
<th>4.5</th>
<th>4.6</th>
<th>4.7</th>
<th>4.8</th>
<th>4.9</th>
<th>5.0</th>
<th>99.999%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

Motivation. Ultra-low latency (ULL) is emerging, but not characterized by far.

Contributions.
- Characterizing the performance behaviors of ULL SSD.
- Studying several system-level challenges of the current storage stack.

Key Insights.
- ULL SSDs can be effectively applied to real-life storage stack. (RW mixed)
- NVMe queue mechanisms are required to be optimized for ULL SSDs.
- Polling-based I/O completion routine isn’t effective for current NVMe SSDs.
Thank you

Q&A