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)**
- **ICH (South Bridge)**
- **DRAM**
- **PCI Express**
- **SATA SSD**

Direct Access

**NVMe SSD**

*High bandwidth*
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 µs
Write: 660-5000 µs

Z-NAND [1]

Read: 3 µs (15~20x)
Write: 100 µs (6~7x)

Z-NAND [1]

Technology: SLC based 3D NAND 48 stacked word-line layer
Capacity: 64Gb
Page Size: 2kB/Page

Z-NAND based archives “Z-SSD”
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)

---

<Our testbed w/ Z-SSDs>
Performance Analysis
Overview

Increase queue depth

Rd  Wr  Rd  Wr  Rd  Wr  Rd  Wr

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

② **Bandwidth**

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

- **Sequential Read (SeqRd)**
- **Random Read (RndRd)**
- **Sequential Write (SeqWr)**
- **Random Write (RndWr)**

**Key Observations**

- **ULL**
- **NVMe**

**Graph Details**

- **Average Latency (μsec)**
- **I/O Depth**

**Notes**

- **Split-DMA & Super-Channel**
- **4KB DMA = 8 μs (t_R = 3 μs)**

**Annotations**

- **5.1x**
- **1.8x**
- **11 μs**

**Legend**

- **SeqRd**
- **RndRd**
- **SeqWr**
- **RndWr**
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

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

Way 1

DMA (for write request)

Way 2

Wait

$t_R$ CMD

Data Out

Reduce read latency & Increase QoS

Suspend/Resume [1]

Way 1

Read

DMA (for write request)

Way 2

$t_R$ CMD

Data Out
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

- **NVMe SSD**
  - Only 50% of Max BW

- **ULL SSD**
  - Almost Max BW

**I/O request rescheduling** within queue.

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

**Short** write latency

**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

Low latency: Command Execution

Polling. Submit request → Polling → Complete request

NVMe Controller: Gain

ISR: CS

Raise IRQ: ②

Wake: ③

Finishes: ①

SSD: Command Execution

Done??
Overall Performance

---

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

Polling-based I/O services incur significant system level overheads → Needs to be addressed.

High CPU utilization
= Frequent memory access

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

High memory bound
= Frequent memory access
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