Asynchronous I/O Stack: A Low-latency Kernel I/O Stack for Ultra-Low Latency SSDs

Gyusun Lee¹, Seokha Shin¹, Wonsuk Song¹, Tae Jun Ham², Jae W. Lee² and Jinkyu Jeong¹

1 Sungkyunkwan University (SKKU)
2 Seoul National University (SNU)
Storage Performance Trends

- Emerging ultra low-latency SSDs deliver I/Os in a few $\mu s$

Overhead of Kernel I/O Stack

- Low-latency SSDs expose the overhead of kernel I/O stack

<table>
<thead>
<tr>
<th>Device</th>
<th>Kernel</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>SATA SSD</td>
<td>24.0%</td>
<td>76.0%</td>
</tr>
<tr>
<td>NVMe SSD</td>
<td>37.6%</td>
<td>62.4%</td>
</tr>
<tr>
<td>Z-SSD</td>
<td>35.4%</td>
<td>64.6%</td>
</tr>
<tr>
<td>Optane SSD</td>
<td>35.5%</td>
<td>64.5%</td>
</tr>
</tbody>
</table>

Read

Write (+fsync)
Synchronous I/O vs. Asynchronous I/O

Our Idea: apply asynchronous I/O concept to the I/O stack itself
Read Path in Asynchronous I/O Stack

```
sys_read()
```

CPU  | I/O stack operations  | Device I/O  | Return to user
--- | --- | --- | ---
Device

Vanilla I/O Stack

```
sys_read()
```

CPU  | Async. operations  | Device I/O  | Return to user
--- | --- | --- | ---
Device

Asynchronous I/O Stack

Latency reduction
Write(+fsync) Path in Asynchronous I/O Stack

**Vanilla I/O Stack**

1. CPU invokes `sys_fsync()`
2. I/O stack ops.
3. Device
   - Device I/O
4. Return to user

**Asynchronous I/O Stack**

1. CPU invokes `sys_fsync()`
2. I/O stack ops.
3. Flush
   - Device I/O
   - Device I/O
   - Device I/O
4. Return to user
   - Latency reduction
FIO Performance (on Optane SSD)

- **Random read**
  - Vanilla
  - AIOS
  - AIOS-poll

- **Random write (+fsync)**
  - Vanilla

Up to 23% latency reduction

Up to 26% latency reduction

Latency (µs)

Block size

Latency (µs)

Block size

7.6µs
Main Talk of Asynchronous I/O Stack

- Detailed analysis of read/write I/O stack
- Asynchronous I/O stack
  - Proposed read and write(+fsync) paths in detail
- Lightweight block I/O layer
  - Low-latency block I/O service for ultra-low latency SSDs
- More performance measurements with various workloads

USENIX ATC 2019, Thursday July 11
Track I Exotic Kernel Features, 3:50pm
Real-world Workload Performance

- **RocksDB DBbench readrandom**
  - Up to 27% throughput improvement
  - Graph showing OP/s (k) vs Threads for Vanilla and AIOS.

- **Filebench varmail**
  - Up to 29% throughput improvement
  - Graph showing OP/s (k) vs Threads for Vanilla and AIOS.