Is SSD with CXL Interfaces Brilliantly Stupid or Stupidly Brilliant?

YANG SEOK KI, Ph.D.
Vice President, Memory Solutions Lab., Samsung Electronics
Stupid or Brilliant?

Draisine (Running machine), 1817: the 1st bicycle in record

The Jazz Singer, 1927: the 1st movie with an audio track
SSD with CXL Interfaces

- Storage with memory and/or storage interfaces
Technical Needs
Memory Hierarchy

- Keep hot data close to CPU using data locality

Diagram:

- Core
- Cache
- Main Memory
- Flash Memory
- Hard Disk
- Remote Storage

Cost, Bandwidth vs. Latency vs. Capacity

Hot vs. Cold

Traditional Workload
Needs (1): Persistent Memory

- Discontinuation of the leading technology
Needs (2): Secondary Memory

- High overhead of virtual memory implementation

**<OS-level>**
Swap for memory extension on disk

**<User-level>**
Redis Auto Tiering for memory extension on SSD
Needs (3): Fast Small IO

• High overhead of IOs smaller than 4KB
CXL-based SSD
CXL-based SSD

- A Hybrid device of DRAM and NAND with CXL interfaces
CXL (Compute Express Link)

- Asynchronous blocking memory interface with optional coherency
CXL Device Types

- Device types based on protocols, not functions

Type1 Device

Type2 Device

Type3 Device
CXL-based SSD as Persistent Memory

• Type-3 device similar to NVDIMM

<table>
<thead>
<tr>
<th>Dump Size</th>
<th>Dump Time(s)</th>
<th>Dump Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>16GB</td>
<td>3.2</td>
<td>58J</td>
</tr>
<tr>
<td>32GB</td>
<td>6.4</td>
<td>115J</td>
</tr>
<tr>
<td>64GB</td>
<td>12.8</td>
<td>229J</td>
</tr>
<tr>
<td>128GB</td>
<td>25.6</td>
<td>457J</td>
</tr>
<tr>
<td>256GB</td>
<td>51.2</td>
<td>913J</td>
</tr>
<tr>
<td>512GB</td>
<td>102.4</td>
<td>1,825J</td>
</tr>
</tbody>
</table>

Load/store access to DRAM
Persistency via flushes to NAND
Power Failure Protection

- Host
- Controller
- CXL Root Port
- CXL BUS
- Memory-Semantic SSD™ with Persistence Feature
  - DRAM
  - NAND Flash Backing Store
- Power Supply
- Battery
- Power Failure Protection
Key Features & Benefits

- Battery-backed DRAM with speed comparable to DDR5
- Persistence achieved with data dumps to NAND flash
- Supports flush-on-fail with CXL 2.0 GPF feature
CXL-based SSD as Secondary Memory

**CXL-based SSD with built-in DRAM**

**Built-in DRAM**
- Processing AI and ML applications, usually need relatively small-sized data chunks
- Applications can write data to the DRAM cache at DRAM speed

**CXL Technology**
- Low latency enabled by CXL.memory protocol

**Tiered Memory Solution**

- CPU
- CXL-based SSD as Secondary Memory
Secondary Memory Options

- Example of Memory Configuration with TM Mode

<2TB Main Memory (Case-1)>

L4 $ 128GB DDR5 DIMM

L4 $ 128GB DDR5 DIMM

2TB SSD Capacity

<1-Tier Host-managed Device Memory (Case-2)>

CPU

Local DRAM 128GB DDR5 DIMM

Local DRAM 128GB DDR5 DIMM

HDM-T2 2TB

<2-Tier Host-managed Device Memory (Case-3)>

CPU

Local DRAM 128GB DDR5 DIMM

Local DRAM 128GB DDR5 DIMM

HDM-T2 2TB
Option 1 Performance

Key Features & Benefits

- Small granularity data access enable performance scales with cache hits
- Direct memory access advantage; no software cache overhead
- Large memory capacity at lower TCO

---

**Compared to PCIe Gen4 NVMe SSD**
CXL-based SSD as Fast Small IO Storage

File system-based access supports legacy NVMe

Load/store access For memory-mapped files

CXL interface

Normal I/O  Small I/O

Cache CTRL

NAND

DRAM Cache

<table>
<thead>
<tr>
<th>CXL Mem Read</th>
<th>Random Perf (128B)</th>
<th>Cache hit 0%</th>
<th>0.8 MIOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cache hit 50%</td>
<td>1.5 MIOPS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cache hit 100%</td>
<td>35.0 MIOPS</td>
<td></td>
</tr>
</tbody>
</table>

Latency

| Cache hit/miss | <1us/ 70us |

CXL.io

Seq. perf (128KB) Read: 5,500 MB/s
Write: 2,000 MB/s

Random Perf (4KB) Read: 800 KIOPS
Write: 85 KIOPS

Small I/O

4 KB

64 Bytes
128 Bytes
Fine-grain Access to Storage Data

Method 1
- Load/Store
- Byte Addressable CXL_mem
- HDM Region (500GB)
- Sector
- Main Memory
- Sector
- NVMe PCI CFG/BAR
- System Memory Space

Memory-Semantic SSD™

Method 2
- NVMe Driver
- read(), write()
- Sector/Block CXL.io
- Sector
- Initiate DMA
- Data Transfer

Fine-grain Access to Storage Data

Application
- Inference Engine Execution
- Model Update: Embedding Table Write

DLRM Benchmark
- Fine Grained Read
- Bulk Data Write

Mmap

VMA

A CXL File

mapped VMA

Virtual Memory

File System

A CXL File

DRAM Memory

CXL Memory

NVMe Device

Memory-Semantic SSD™
DLRM Performance with Fast Small IOs

DLRM** performance (Meta)

Inferences Per Second*

<table>
<thead>
<tr>
<th></th>
<th>I/O Based</th>
<th>Host Software Cache Based</th>
<th>Hardware Device Cache Based</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Overhead</td>
<td>4,450</td>
<td>16,824</td>
<td>30,445</td>
</tr>
<tr>
<td>Low DRAM data reuse</td>
<td>High Software Overhead</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Results based on publicly available DLRM workload traces from Meta and FPGA based PoC Memory-Semantic SSD™

** DLRM : Deep Learning Recommendation Model
Movie Recommendation System Demo
Challenges and Opportunities
Standard and Eco

• No definition and spec
Latency Tolerance

- Impact of long latency on CPU performance
Cache Management

• Managing in-device DRAM is the key!

Key Features & Benefits

• Close to DRAM end-to-end performance at a lower TCO*
• Up to ~10x better end-to-end performance with FPGA-based PoC**
Wrap Up

• SSD with CXL interfaces for
  • Persistent memory
  • Performant secondary memory
  • Storage for AI and HPC
  • Near data processing platform

• Community efforts
  • Standard for SSD with CXL interfaces (+cache management)
  • Software ecosystem
  • CPU architecture to tolerate long latency
Thank You