A Study of Application Performance with Non-Volatile Main Memory

Yiying Zhang,
Steven Swanson
Memory

Fast Volatile In bytes

Storage

Slow Persistent In blocks

Next-Generation Non-Volatile Memory (NVM)
The Landscape of Memory and Storage

Latency

$\begin{array}{c|c|c|c|c|c|c|c}
\text{Disk} & \text{Flash} & \text{DRAM} & \text{Next-Gen NVM} & \text{Phase Change Memory} & \text{Memristor} & \text{STT-RAM} & \ldots \\
1\text{ns} & 10\text{ns} & 100\text{ns} & 1\text{us} & 10\text{us} & 100\text{us} & 1\text{ms} & 10\text{ms}
\end{array}$

Is Changing!
Next-Generation Non-Volatile Memory

- Byte Addressable
- Low Latency
- Persistence
- Capacity
Outline

Introduction

Application performance with NVMM

NVMM in data centers

Conclusion
Intel NVMM Emulator

Use DRAM to emulate different NVMMs

Delay read latency by increasing CPU stalls

Read and write bandwidth throttling by limiting DDR transfer rate

Emulate data persistence by flushing CPU caches and adding software delay
NVMM Data Persistence

Flush CPU cache
- \textit{clflush}: flush one cache line at a time
- \textit{clflushopt}: reduces ordering guarantee
- \textit{clwb}: does not force to throw away cache lines
- WOF: read followed by non-temporal write

Ensure data durability in NVMM and ordering
- \textit{pcommit}
- \textit{sfence, mfence}
NVMM Data Persistence Cost

Update 1 byte, followed by \textit{msync} to a range of increasing size.

Making cache flush fast reduces data persistence cost.
Still costly when syncing a lot of data.
Selective Persistent Flushing (SPF)

Goal: only flush modified data

Use page table entry dirty bit
  - Only flush cache lines in dirty pages

Sub-page data flushing
  - Change system call granularity (e.g., msync)

Alternative: application modification
Reduce NVMM Data Persistence Cost

- default
- clflushopt
- HDD
- WOF
- clwb
- SSD
- SPF

msync latency (msec) vs msync size (MB)
Evaluation Configurations

Start with DRAM configuration
Adding PCM configurations one at a time

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Read Latency (ns)</th>
<th>Throughput Ratio to DRAM</th>
<th>Write Barrier Latency (μs)</th>
<th>Cache Flush Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDRAM</td>
<td>150</td>
<td>1</td>
<td>0</td>
<td>clflush</td>
</tr>
<tr>
<td>PM-Lat</td>
<td>300</td>
<td>1</td>
<td>0</td>
<td>clflush</td>
</tr>
<tr>
<td>PM-BW</td>
<td>300</td>
<td>1/8</td>
<td>0</td>
<td>clflush</td>
</tr>
<tr>
<td>NVMM-Raw</td>
<td>300</td>
<td>1/8</td>
<td>1</td>
<td>clflush</td>
</tr>
<tr>
<td>NVMM-WOF</td>
<td>300</td>
<td>1/8</td>
<td>1</td>
<td>WOF</td>
</tr>
<tr>
<td>NVMM-Opt</td>
<td>300</td>
<td>1/8</td>
<td>1</td>
<td>SPF</td>
</tr>
</tbody>
</table>
Evaluated Applications

File system applications: FileBench

Key-value store: MongoDB

Relational database: MySQL

Using NVMM as big memory: Memcache
FileBench – Varmail Performance

Throughput Ratio to PDRAM

- PDRAM
- PM-Lat
- PM-BW
- NVMM-Raw
- NVMM-WOF
- NVMM-Opt
- SSD
- HDD
- DRAM
MongoDB Key-Value Store Performance

MongoDB fsync_safe mode, YCSB workloads
Read latency = 300ns, read&write bandwidth = 1/8 DRAM bandwidth

Throughput Ratio to PDRAM

50% read, 50% write

95% read, 5% write

- PDRAM
- PM-Lat
- PM-BW
- NVMM-Raw
- NVMM-WOF
- SSD
- NVMM-Opt
- HDD
- DRAM
Summary of NVMM Performance

Net-gen NVM can offer fast, persistent storage

App performance with NVMM can be close to DRAM

Making data persistent can be costly

Techniques to reduce the cost of data persistence and the amount of data to make persistent
Outline

Introduction

Application performance with NVMM

NVMM in data centers

Conclusion
NVMM in Data Center

Data Center

NVMM

Reliability

Availability

Storage Data Replication

Low Latency

Consistency Protocol

Networking Stack

Software Overhead
Existing data replication is too slow for NVMM!
Mojim: Replicated, Reliable, Highly-Available NVMM with Good Performance
Mojim Approach

Replication Protocol: efficient 2-Tier

OS: optimized, generic layer

Networking: efficient RDMA

Interface: memory-based

Architecture: cache-flush optimization
Mojim Interface

Good performance

Easy-to-use, generic interface

Applications

load store

open

atomic persist

Sync Point

Mojim

NVMM
Mojim Architecture

Primary Node

Applications

load
store

Mojim Master

atomic
persist

Mirror Node

Applications

load
open

Mojim Mirror

RDMA

NVMM

NVMM
Async Mode

Primary Node

Applications

Mojim Master

NVMM

RDMA

Mirror Node

Applications

Mojim Mirror

NVMM

Weak Consistency

Good performance
Sync Mode

Primary Node

- Applications
- Mojim Master
- NVMM

Mirror Node

- Applications
- Mojim Mirror
- NVMM

Strong Consistency

Good performance
Sync-2tier Mode

Primary Node
- Applications
- Mojiim Master

Mirror Node
- Applications
- Mojiim Mirror

Mojim Backup
- Flash/Disk

More Redundancy

2 strongly-consistent copies + weakly-consistent copies
Good performance
MongoDB Key-Value Pair Load

Testbed: DRAM as NVMM proxy, 40Gbps Infiniband

Workload: key-value pair insert

Latency (ms)

- MongoDB: 3.0
- Mojim Async: ~1.0
- Mojim Sync-2tier: ~1.0

Replication: 2.8X
Networking: 4.9X
And more reliable
Data Persistence Latency

Testbed: Hardware NVMM emulator, 40Gbps Infiniband
Workload: Persist random 4KB regions in a 4GB mmap’d file
1 RDMA roundtrip < \textit{clflush} the data

Ensures cache coherence without clflush

Strongly ordered, flush one cache line at a time
Summary of NVMM Replication

Non-volatile memory in data center

Efficient data replication

Flexible modes

Replication even faster than no replication
Conclusion

NVMM can potentially fill the gap between memory and storage

Making data persistent can be costly

Mojim: first step of NVMM in data centers
Thank you!
Questions?

yiyiingzhang@cs.ucsd.edu
NVMDelight: http://nvmdel.uchsd.edu