Planning for the Next Decade of NVM Programming

Andy Rudoff
Principal Engineer
Intel Corporation
Planning for the Next Decade of NVM Programming

Andy Rudoff
Principal Engineer
Intel Corporation

Member: SNIA NVM Programming TWG
The Story So Far:

In the beginning the Universe was created.

This has made a lot of people very angry and been widely regarded as a bad move.
The Story So Far:

In the beginning the Universe was created.

This has made a lot of people very angry and been widely regarded as a bad move.

- Douglas Adams, *The Restaurant at the End of the Universe*
Table 1. Comparison of data storage technologies. (Data drawn from public sources and HP internal research).

<table>
<thead>
<tr>
<th></th>
<th>Memristor</th>
<th>PCM</th>
<th>STT-RAM</th>
<th>DRAM</th>
<th>Flash</th>
<th>HD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chip area per bit ((F^2))</td>
<td>4</td>
<td>8–16</td>
<td>14–64</td>
<td>6–8</td>
<td>4–8</td>
<td>n/a</td>
</tr>
<tr>
<td>Energy per bit ((\text{pJ})^2)</td>
<td>0.1–3</td>
<td>2–100</td>
<td>0.1–1</td>
<td>2–4</td>
<td>(10^4–10^4)</td>
<td>(10^6–10^7)</td>
</tr>
<tr>
<td>Read time (ns)</td>
<td>&lt;10</td>
<td>20–70</td>
<td>10–30</td>
<td>10–50</td>
<td>25,000</td>
<td>5–8\times10^6</td>
</tr>
<tr>
<td>Write time (ns)</td>
<td>20–30</td>
<td>50–500</td>
<td>13–95</td>
<td>10–50</td>
<td>200,000</td>
<td>5–8\times10^6</td>
</tr>
<tr>
<td>Retention</td>
<td>&gt;10 years</td>
<td>&lt;10 years</td>
<td>Weeks</td>
<td>&lt;Second</td>
<td>~10 years</td>
<td>~10 years</td>
</tr>
<tr>
<td>Endurance (cycles)</td>
<td>~10^{12}</td>
<td>(10^7–10^8)</td>
<td>(10^{15})</td>
<td>(&gt;10^{17})</td>
<td>(10^3–10^6)</td>
<td>(10^{15}) ?</td>
</tr>
<tr>
<td>3D capability</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Source:
Moving the Focus to SW Latency

App to SSD IO Read Latency (QD=1, 4KB)

- NAND MLC SATA 3 ONFI2
- NAND MLC SATA 3 ONFI3
- NAND MLC PCIe x4 Gen3 ONFI3
- Future NVM PCIe* x4 Gen3

- NVM Tread
- NVM xfer
- Misc SSD
- Link Xfer
- Platform + adapter
- Software

us: 0 20 40 60 80 100 120
Memory or Storage?
Persistence:

Storage
- Block I/O only
- Sync or Async
  - DMA master
- High Capacity?
- Drive-serviceability?
- NAND, NVMe, PCIe

pmem
- Byte addressable
- Sync (probably)
  - DMA slave
- Growing Capacity?
- NVDIMM
- Not NAND, NVMe
  - PCIe?
pmem: The New Tier

- Byte-addressable memory, but persistent
- Must be *reasonable* to stall a CPU waiting for a load to finish
  - So, not NAND NVM based
- Can do small I/O
  - DIMMs are 64B cache line accessible
- Can DMA to it
  - Receive data from network directly to persistence!
SNIA Technical Working Group
Initially defined 4 programming modes required by developers

Spec 1.0 developed, approved by SNIA voting members and published

Defining the NVM Programming Model
Recent Announcements

- Intel® 3D XPoint™ Technology
- The Intel DIMM
1000X faster
THAN NAND

1000X endurance
OF NAND

10X denser
THAN CONVENTIONAL MEMORY
The Memory Timeline

- 1947: RAM
- 1956: PROM
- 1961: SRAM
- 1966: DRAM
- 1971: EPROM
- 1988: NOR Flash Memory
- 1989: NAND Flash Memory
- 2015: 3D XPoint™
(Announced) State of SW Ecosystem

- Detecting pmem
  - BIOS creates ACPI-style information for OS
  - Defined in ACPI 6.0
- Linux support upstream
  - Exposing pmem as block storage
    - Generic NVDIMM driver for Linux released
  - Exposing pmem for direct access
    - Linux DAX upstream
- Naming pmem areas
  - Linux ext4+DAX support upstream
- KVM Changes upstream
- Support in other operating systems emerging
  - Neal’s talk Yesterday
    - Storage Class Memory Support in the Windows Operating System
  - Heavy OSV Involvement in TWG
The Next Decade…
Transparency Levels

- application
- middleware
- libraries
- file systems
- in-kernel usages
- block storage

Increasing barrier to adoption

NVDIMM

User Space

Kernel Space

Application

File System

Driver

2015 Storage Developer Conference. © Intel Corporation. All Rights Reserved.
Transparency Levels

Increasing leverage

application
middleware
libraries
file systems
in-kernel usages
block storage

Application
File System
Driver
NVDIMM

User Space
Kernel Space
One Transparent Example: *pmem Paging*

- Application
- User Space
- Middleware
- Libraries
- File Systems
- Kernel Space
- In-kernel usages
- Block storage
- Driver
- NVDIMM
Paging from the OS Page Cache

User Space

Kernel Space

Application

Application

Application

NVDIMM

NVDIMM

DRAM
Attributes of Paging
(and why everyone avoids it)

- Major page faults
  - Block I/O (page I/O) on demand
  - Context switch – there and back again
  - Latency of block stack
- Available memory looks much larger
  - But penalty of fault is significant
- Page in must pick a victim
  - Based on simplistic R/M metric
  - Can surprise an application
- Many enterprise apps opt-out
  - Managing page cache themselves
  - Using intimate date knowledge for paging decisions
- Interesting example: Java GC
Paging to pmem

User Space

Application

Kernel Space

Application

Application

User Space

NVDIMM

NVDIMM

NVDIMM

NVDIMM

DRAM

NVDIMM

…
When Will pmem Paging be Cost Effective?

- When pmem costs less than (or close to) DRAM
- When pmem performance approaches DRAM
- When pmem capacity becomes significant

- When pmem is as reliable as memory
  - Probably needs to exceed memory reliability due to the fact it is persistent
Not just for pmem…

- High-bandwidth memory
- NUMA localities
- Different NVM technologies
Extending into User Space

- application
- middleware
- libraries
- file systems
- in-kernel usages
- block storage
- NVDIMM
- Driver
- File System
- Kernel Space
- User Space
NVM Library: pmem.io
64-bit Linux Alpha Release

- Open Source
  - [http://pmem.io](http://pmem.io)
- `libpmem`
- `libpmemobj`
- `libpmemblk`
- `libpmemlog`
- `libvmem`

Libraries:
- `libpmem`
- `libpmemobj`
- `libpmemblk`
- `libpmemlog`
- `libvmem`
Replication Challenge of pmem

- Application
  - Standard Raw Device Access
- Application
  - Standard File API
- Application
  - Standard File API
- NVDIMM Driver
- File System
- PM-Aware File System
- MMU Mappings
- NVDIMM

No Kernel Interception Point
RDMA to pmem
Non-Transparent pmem Use Cases

- Volatile caching
  - Due to capacity, relative simplicity

- In-memory database

- Storage appliance write cache
  - Also for large structures like dedup tables
  - Leverage RDMA capability

- Large, byte-addressable data structures
  - Example: HBASE hash table

- HPC
  - Example: checkpoint
  - Example: distributed versioned object store
Prediction: The Sweet Spots

- Application
- File System
- Driver
- NVDIMM

User Space

- User Space
- Middleware
- Libraries

Kernel Space

- Kernel Space
- File Systems
- In-kernel usages
- Block storage

Increasing barrier to adoption
Prediction: The Big Challenge

application

middleware
libraries

file systems
in-kernel usages
block storage

Application
File System
Driver
NVDIMM

User Space
Kernel Space

Worth the complexity?
Summary…
Summary

- Building a SW ecosystem for pmem
  - Won’t overcome Enterprise time-to-adoption, but…
  - Linux support upstream
  - Other operating systems progressing

- Cost versus Benefit Challenge
  - Cost of Emerging NVM
  - Cost of application complexity
  - Fall back to transparency at various levels

- What you can do to prepare
  - Learn NVM programming model
  - Map use cases to pmem
  - Contribute to libraries, SW ecosystem
Links to More Information

- SNIA NVM Programming Model
  - [http://www.snia.org/forums/ssi/nvmp](http://www.snia.org/forums/ssi/nvmp)

- Intel® Architecture Instruction Set Extensions Programming Reference

- Open Source NVM Library work
  - [http://pmem.io](http://pmem.io)

- Linux kernel support & instructions
  - [https://github.com/01org/prd](https://github.com/01org/prd)

- ACPI 6.0 NFIT definition (used by BIOS to expose NVDIMMs to OS)

- Open specs providing NVDIMM implementation examples, layout, BIOS calls:

- Google group for pmem programming discussion:
  - [http://groups.google.com/group/pmem](http://groups.google.com/group/pmem)