Application Access to Persistent Memory – The State of the Nation(s)!

Stephen Bates, Paul Grun, Tom Talpey, Doug Voigt

Microsemi, Cray, Microsoft, HPE
The Suspects

Stephen Bates
Microsemi

Paul Grun
Cray

Tom Talpey
Microsoft

Doug Voigt
HPE
We’ve Come a Long Way, Baby!
Persistent Memory (PM)

Low Latency  Memory Semantics  Storage Features
Taxonomy

- **NVM** – Non-Volatile Memory. All types, including those that are not byte-addressable
- **PM** – Persistent Memory. Sometimes PMEM is used but we use PM in this talk
- **NVMe** – NVM Express. A block protocol to run over PCIe, RDMA or Fibre Channel. A SATA/SAS replacement.
- **NVMP** – NVM Programming Model. Application-visible NVM behavior
- **NVMf** – NVMe over Fabrics. NVMe extended over fabrics

“This taxonomy is Jim Pappas approved!”
## Low Latency

<table>
<thead>
<tr>
<th></th>
<th>Latency</th>
<th>Relative</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 Cache Read</td>
<td>0.5ns</td>
<td>1</td>
</tr>
<tr>
<td>L2 Cache Read</td>
<td>7ns</td>
<td>14</td>
</tr>
<tr>
<td>DRAM Read</td>
<td>100ns</td>
<td>200</td>
</tr>
<tr>
<td><strong>The PM Opportunity</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NVMe DRAM SSD Read</td>
<td>10us</td>
<td>20,000</td>
</tr>
<tr>
<td>NVMe NAND SSD Read</td>
<td>150us</td>
<td>300,000</td>
</tr>
<tr>
<td>SAS HDD</td>
<td>500us</td>
<td>1,000,000</td>
</tr>
</tbody>
</table>

NVMe SSDs are (relatively) high latency!

PM provides persistence at memory-like speeds and semantics
Where Are We?
What is Needed?

- Media & Form-Factors
- OS Support
- Apps
- Protocols & Interconnect
- Libraries & Toolchain
Lots of Moving Parts

- Apps
  - User space apps
  - Kernel space apps
  - Communications middleware

- Protocols & Interconnect
  - APIs

- Protocols & Interconnect
  - Drivers
  - Communications Infrastructure

- Media & Form-Factors
  - Non-volatile Media, Form Factors

Consumers

Infrastructure

Media, Form Factors
Where does PM sit?
(Answer – anywhere it wants to)
Rationalizing the Problem Space

NVM consumers

- Memory
  - byte-addressable

- Storage
  - object, file, block…

- APIs
  - OFI, Verbs, NVMF, ND, NDK

- network

- I/O bus

- memory bus

- Non-volatile Media,
  - Form Factors

- SNIA, OSVs …

- OpenFabrics Alliance,
  - NVMe Inc,
  - Linux,
  - Windows…

- IETF, IBTA, PCIe SIG,
  - OS drivers…

- Vendors, JEDEC…
Start with Consumers of NVM Services

- NVM devices (SSDs...)
- storage client
- user app
- file system
- virtual switch
- provider
- NIC
- load/store, memcopy...
- POSIX read or write

Remote NVM device (storage, memory)
Application View

- SNIA NVMP
  - Describe application visible behaviors
  - APIs align with OSs

- PM File System Actions
  - Map – expose PM directly to applications
  - Optimized Flush – make data persistent

Middleware features e.g. RAID

PM data structure libraries

PM Aware File Systems

User

Kernel

File APIs

PM Aware Apps

Ld/St

PM Device

MMU Mappings
Possible Stack for NVM Access

- VFS / Block Layer
  - SCSI
  - NVMe
    - ulp
- local I/O
  - PCIe
  - HBA
  - SSD
  - NVDIMM
- remote I/O
  - remote byte addressable
  - iSCSI
  - sockets
  - NIC, RNIC, HCA, NIC
  - libfabric
  - provider
  - kfabric
  - kverbs
  - fabric-specific device
  - NIC
  - RNIC
  - HCA

APIs

- VFS / Block I/O / Network FS / LNET
  - SRP, iSER, NVMe/F, NFSoRDMA, SMB Direct, LNET/LND, ...
  - iSCSI
  - sockets

kernel application

user app

local byte addressable
Optimizing Fabrics for NVM

- Add persistence semantics to RDMA protocols
  - Register persistent memory regions
  - Completion semantics to ensure persistence, consistency
  - Client control of persistence
- Solve the “write-hole” problem
- Lots of Initiatives underway!

Can we make this work for NVDIMMs? NVMe SSDs with CMBs?
Simple Math

\[ \text{NVMe} + \text{RDMA} = \text{AWESOME} \]

\[ \text{PM} + \text{RDMA} = \text{AWESOME}^2 \]

\[ 2 + 2 = 4 \]
...continuing down the stack

NVM consumers

Memory byte-addressable

Storage object, file, block...

APIs -OFI, Verbs, NVMe/F...

Network

I/O bus

Memory bus

Non-volatile Media, Form Factors
- NVMe over Fabrics – Present an NVMe block device to a client over RDMA or Fibre Channel

- NVMe Controller Memory Buffers – Standardize (persistent) PCIe memory on NVMe devices. NVDIMM-N on PCIe bus?

- LightNVM – A low-level SSD interface that is more aligned to the underlying media (NAND)
<table>
<thead>
<tr>
<th>Category</th>
<th>Vendors</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM Drop-In</td>
<td>Everspin, Micron, Toshiba, SK Hynix</td>
<td>DRAM like latency, Super-Cap Replacement, Not for bulk storage, Memory Interface</td>
</tr>
<tr>
<td>Storage Class Memory</td>
<td>Micron-Inlet, SanDisk, Toshiba, Crossbar, Nantero</td>
<td>Faster than NAND, Cheaper/Slower than DRAM, Byte Addressable, Block and Memory Interfaces</td>
</tr>
<tr>
<td>NAND</td>
<td>Micron, Toshiba, SanDisk, SK Hynix, Samsung</td>
<td>Lowest cost, Slow (for NVM), Not byte addressable, cheap and plentiful, Block Interface</td>
</tr>
</tbody>
</table>
PM Form Factors

NVDIMM-N

NVDIMM-P

Not-NAND NVMe

NAND NVMe
## PM Form Factors

<table>
<thead>
<tr>
<th>Form-Factor</th>
<th>Media</th>
<th>Latency</th>
<th>Memory Semantics</th>
<th>Storage Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVDIMM-N</td>
<td>DRAM/MRAM</td>
<td>🎈</td>
<td>🎈</td>
<td>😞</td>
</tr>
<tr>
<td>NVDIMM-P</td>
<td>NAND/PM</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
</tr>
<tr>
<td>Non-NAND NVMe</td>
<td>DRAM/PM</td>
<td>😞</td>
<td>😞</td>
<td>🎈</td>
</tr>
<tr>
<td>NAND NVMe</td>
<td>NAND</td>
<td>😞</td>
<td>😞</td>
<td>🎈</td>
</tr>
</tbody>
</table>

Form factors impact Features
(No DMA engines on a DIMM!)
PM Scenarios

- PM region as a block device (a la persistent ram disk).
- Filesystems support: direct access to the memory (e.g. DAX), PM aware FS (e.g. m1fs).
- You can put your files, databases etc. on top.
- Remember we are crawling right now!
- Soon: Shared persistent memory
Libraries and Toolchains

EASY

```c
int main(int argc, char *argv)
{
    printf("Hello, PMEM World!\n");
    return 0;
}
```

HARD

```assembly
section .text
global _start
_start:
    mov edx,len
    mov ecx,msg
    mov ebx,1
    mov eax,4
    int 0x80

    mov eax,1
    int 0x80

section .data
msg db 'Hello, PMEM World!',0xa
len equ $ - msg
```

Make it easy for applications to utilize PM, regardless of OS and ARCH!
Call to Arms
Call to Arms

- **Libraries and Toolchains**: NVML for non-x86, integration into glibc/gcc etc.
- **Media & Form Factors**: Production PM, appropriate PM form factors.
- **Protocols and Interconnect**: Enhancements to NVMe and RDMA, PM over Fabrics, standardization of memory channels.
- **OS Support**: ISA updates, DAX devices, Other OS, new OSes?
Conclusions

- We are almost walking! Help out if you can
- If you want sub 10us access to persistent data then PM may be for you
- The CPU vendors have a lot of say in the interconnect but some open options exist too
- Toolchains, libraries and OSes are adapting
- New applications will complete the jigsaw and lead to revenue