What You can Do with NVDIMMs

Rob Peglar
President, Advanced Computation and Storage LLC
A Fundamental Change Requires An Ecosystem

- Windows Server 2016
- Windows 10 Pro for Workstations
- Linux Kernel 4.2 and later
- VMware, Oracle, SAP HANA early enablement programs

- Multiple vendors shipping NVDIMMs
- SNIA NVDIMM Special Interest Group (formed Jan’14)
- Successful demonstrations of interoperability among vendors

- JEDEC JESD245B.01: Byte Addressable Energy Backed Interface (released Jul’17)
- JEDEC JESD248A: NVDIMM-N Design Standard (released Mar’18)
- SNIA NVM Programming Model (v1.2 released Jun’17)
- unfit ACPI NVDIMM Firmware Interface Table (v6.2 released May’17)

- All major OEMs shipping platforms with NVDIMM support
- Requires hardware and BIOS mods

© 2019 Storage Networking Industry Association. All Rights Reserved.
JEDEC-Defined NVDIMM Types

NVDIMM-N
- Host has direct access to DRAM
- NAND flash is only used for backup
- Capacity = DRAM (10’s - 100’s GB)
- Latency = DRAM (10’s of nanoseconds)
- Endurance = DRAM (effectively infinite)
- No impact to memory bus performance
- Low cost controller can be implemented
- Specifications completed and released
- Ecosystem moving into mature stage

NVDIMM-P
- Host is decoupled from the media (agnostic to PM type)
- New protocol to “hide” non-deterministic access
  - Capacity = PM (100’s GB+)
  - Latency = PM (>> 10’s of nanoseconds)
  - Endurance = PM (finite)
- Likely to impact memory bus performance
- Complex controller & buffer scheme likely required
- Specifications still under definition (2H’19 release?)
- No ecosystem yet, likely DDR5 timeframe

NVDIMM Types Are Complementary, Not Competing
NVDIMM Target Application Areas

Databases
- Log Acceleration
- In-Memory Commit

Storage
- Filesystems
- Fast Caching
- SSD Wear-Out

Virtualization
- Higher VM Consolidation
- More Virtual Users/System

Big Data
- Fast IOPs Workloads
- In-Memory Processing
- Metadata Store
- Low Latency Look-Up & Processing

Cloud Computing/ IoT
- Byte-Level Data Processing

Artificial Intelligence

USE CASES

The same factors driving NAND Flash adoption apply to NVDIMMs: IOPS, Latency, Performance

NVDIMM addressing is exactly like DRAM
PMEM and NVDIMM-P

Wendy Elsasser
Distinguished Engineer, Arm
PMEM potential and options

- **Database applications**
  - Journaling, efficient logging
- **HPC**
  - Faster checkpointing
- **Big Data**
  - Dense memory, higher performance
- **Scale-out storage**
  - Meta-data storage, write buffering, caching
- **Multi-tenant, cloud services**
  - DRAM + PMEM for low cost, high capacity

---

A wide variety of technologies with varied characteristics
Leveraging the DDR bus

- Re-use DDR bus,
  - Share command/address, data pins
- Add feedback channel
  - Enable non-deterministic timing
- Extended address for higher capacity
  - Up to 8TB per rank currently
- New Opcodes
  - Media agnostic interface
- Meta-data sent on ‘ECC’ bus
  - Enable flow control
  - Guarantee transmission, data valid
  - Support out of order responses
Adding non-determinism DDR

New command to receive data at deterministic delay equivalent to CAS latency

New handshake signal

Extended address

Time Break

Non-Deterministic Delay

Deterministic Delay
NVDIMM-P meta-data

- Transferred with write data and returned with read data
  - ECC to verify correct transmission
  - Poison flag
  - User defined meta-data

- Returned with read data
  - Read ID to enable out of order completion
    - 8-bits for DDR4, 10-bits for DDR5 currently
    - Host matches to Read ID sent with command
  - Write credits to ensure write buffers don’t overflow
    - Separate credits for write and persistent write commands
# Write commands and persistence

Managing a point of persistency (Pop)

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XWRITE</td>
<td>• Indefinitely buffered on-DIMM in volatile media</td>
</tr>
</tbody>
</table>
| PWRITE | • Will ultimately be committed to NV media  
         • Setting optional ‘Persist’ flag forces push to NV media  
         • Enables smaller granularity persist ops  
         • Optionally colored with a Write-Group-ID (WGID) |
| FLUSH  | • Previous XWRITE, PWRITE data pushed to NV media  
         • Attributes identify target for optional optimization  
         • PWRITEs with a specified WGID,  
         • all-PWRITEs,  
         • all-PWRITEs and all XWRITEs  
         • Final FLUSH sequence defined for power-fail event |

![Push to NVM Latency](image)

Sub-us latency (media dependent) with smaller persist operations for all media bandwidths
## Correlating industry terms

<table>
<thead>
<tr>
<th>System Behavior</th>
<th>Energy Backed NVDIMM-P</th>
<th>Non-Energy Backed NVDIMM-P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push to point of persistency</td>
<td>XWRITE and PWRITE (persistent write) data successfully transmitted across DIMM boundary.</td>
<td>Same as Energy Backed ”Deep FLUSH” Case</td>
</tr>
<tr>
<td></td>
<td>- Memory store potentially buffered in energy backed domain on DIMM</td>
<td></td>
</tr>
<tr>
<td>“Deep FLUSH”</td>
<td>Leverages NVDIMM-P commands:</td>
<td></td>
</tr>
</tbody>
</table>
| - Defined by SNIA NVM programming model | 1) PWRITE (Persist=1)  
2) FLUSH                                                                                                                                                                                                 |                                                                                            |
|                                  | Persistent memory store pushed to the NV media                                                                                                                                                                         |                                                                                            |
|                                  | - DIMM returns completion flag                                                                                                                                                                                           |                                                                                            |

NVDIMM-P protocol supports both point of persistency and ”Deep FLUSH” concepts regardless of energy backing source.
NVDIMM-P persist operation

- Host notified when persist operation completed
  - PWRITE(Persist = 1) or FLUSH command
- Optional small-granularity persistence with WGID
  - WGID management tracks pending and completed persist status
- Final FLUSH for power-fail event
What’s next

- Emerging NVM redefining the memory sub-system
  - Transformative capacity
  - Directly addressable persistent memory

- Persistent capability required across power-fail events
  - Linux PMEM drivers and ISA support currently available
  - NVDIMM-P natively supports persistence

- JEDEC actively defining NVDIMM-P protocol for standardized solution
  - Core items have been balloted to enable IP development
  - Will continue to flush out details to finalize v1.0 specification
  - Aligning firmware definition to be similar to BAEBI
  - Become more active in JEDEC for more details and to help steer the future!