Non-Volatile Memory Modules (NVDIMMs)

Bill Gervasi
Principal Systems Architect
bilge@Nantero.com
Demand Outpacing Capacity

In-Memory Computing
Artificial Intelligence
Machine Learning
Deep Learning

Memory Demand
DRAM Capacity
Chumming Up to the CPU

Lower latency, higher throughput

Mass storage moving closer to the CPU
DRAM Channel Protocol

Designed around direct connection to a DDR device
Fully deterministic operation required
Multiplexed address bus with:
- Chip selects (ranks)
- Rows
- Columns
- Commands

DDR4 limits:
- 16 Gb per chip
- 144 chips per module
- 256 GB per DIMM

DDR5 limits:
- 32 Gb per chip
- 288 chips per module
- 1 TB per DIMM
CPU communicates with DRAM only
On power fail, Controller copies contents to Flash
External energy source powers NVDIMM until backed up
NVDIMM-N Backup Protocol

**Power fail**
- Complete burst in process
- Save all pending operations
- Copy DRAM to NVM

**Power restore**
- Check save status
- Copy NVM to DRAM
- Run
Power failure is a key factor in server software design.

Checkpointing intermediate results to storage affects performance.

Data persistence near the CPU is a huge improvement in systems architecture.
Persistence in Main Memory

Old Process

Database Transaction

Checkpoint

Database Transaction

Checkpoint

Database Transaction

Add Persistence Memory

Persistent Memory (NVDIMM)

Transaction critical data

Temporary data

Application code

etc

Main Memory (DRAM)

New Process

Database Transaction

Database Transaction

Database Transaction

Database Transaction
Unfortunately, NVDIMM-N doesn’t solve the capacity demand… in fact makes it worse

NVDIMM-N capacity is half of the equivalent DRAM module capacity

Does add data persistence
Universe of Persistent Memories

Many technologies coming online to fill the gap between DRAM and Flash

- Hard Disk
- SSD
- NVMe
- Wasteland
- DDR DRAM
- Flash

Moderate speed
Moderate endurance
Capacity range

- Phase Change
  - 3D Xpoint
  - Resistive RAM
  - Magnetic RAM

Painfully slow
Lotsa cheap bits
Low endurance

PMs do not replace DRAM though

See my separate presentation on Memory Class Storage…
Virtualizing the DRAM Channel

Incorporating PM into the DRAM channel requires:

• Mapping devices into the DRAM address range
• Allowing for non-determinism for bookkeeping operations

Limited write endurance forces PMs to go offline for operations such as wear leveling

Media agnostic; any PM can be on the local bus
Virtualizing the DRAM Channel

### DDR4

<table>
<thead>
<tr>
<th>Function</th>
<th>Protocol / Current Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode Register Set</td>
<td>H L L L L BG BA V</td>
</tr>
<tr>
<td>Refresh</td>
<td>H L L L L L L L BG BA V</td>
</tr>
<tr>
<td>Self Refresh Entry</td>
<td>H L L L L L L L BG BA V</td>
</tr>
<tr>
<td>Self Refresh Exit</td>
<td>H L L L L L L L BG BA V</td>
</tr>
<tr>
<td>Single Bank Precharge</td>
<td>H L L L L L L BG BA V</td>
</tr>
<tr>
<td>Precharge all Banks</td>
<td>H L L L L L L BG BA V</td>
</tr>
<tr>
<td>Bank Activate</td>
<td>H L L L L L L BG BA V</td>
</tr>
<tr>
<td>Write (Fixed BL or BCD)</td>
<td>H L L L L L L BG BA V</td>
</tr>
<tr>
<td>Write (BC4, on the Fly)</td>
<td>H L L L L L L BG BA V</td>
</tr>
<tr>
<td>Write (BLU, on the Fly)</td>
<td>H L L L L L L BG BA V</td>
</tr>
<tr>
<td>Write with Auto Precharge</td>
<td>H L L L L L L BG BA V</td>
</tr>
<tr>
<td>Write with Auto Precharge (BC4, on the Fly)</td>
<td>H L L L L L L BG BA V</td>
</tr>
<tr>
<td>Read (Fixed BL or BCD)</td>
<td>H L L L L L L BG BA V</td>
</tr>
<tr>
<td>Read (BC4, on the Fly)</td>
<td>H L L L L L L BG BA V</td>
</tr>
<tr>
<td>Read (BLU, on the Fly)</td>
<td>H L L L L L L BG BA V</td>
</tr>
<tr>
<td>Read with Auto Precharge (Fixed BLU)</td>
<td>H L L L L L L BG BA V</td>
</tr>
<tr>
<td>Read with Auto Precharge (BC4, on the Fly)</td>
<td>H L L L L L L BG BA V</td>
</tr>
<tr>
<td>Read with Auto Precharge (BLU, on the Fly)</td>
<td>H L L L L L L BG BA V</td>
</tr>
<tr>
<td>No Operation</td>
<td>H L L L L L L L V</td>
</tr>
<tr>
<td>Device Deshibited</td>
<td>H L L L L L L L V</td>
</tr>
<tr>
<td>Power Down Entry</td>
<td>H L L L L L L L V</td>
</tr>
<tr>
<td>Power Down Exit</td>
<td>H L L L L L L L V</td>
</tr>
<tr>
<td>ZQ calibration Long</td>
<td>H L L L L L L L V</td>
</tr>
<tr>
<td>ZQ calibration Short</td>
<td>H L L L L L L L V</td>
</tr>
</tbody>
</table>

### NVDIMM-P

<table>
<thead>
<tr>
<th>Function (Reference)</th>
<th>Process / Current Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRS (Mode Register Set)</td>
<td>H L L L L L L L V</td>
</tr>
<tr>
<td>Write &amp; SREAD : RIFD</td>
<td>H L L L L L L L V</td>
</tr>
<tr>
<td>PWRITE : WGD[4:0]</td>
<td>H L L L L L L L V</td>
</tr>
<tr>
<td>SEND</td>
<td>H L L L L L H RIFU</td>
</tr>
<tr>
<td>SEND-W PER</td>
<td>H L L L L H RIFU</td>
</tr>
<tr>
<td>SREAD</td>
<td>H L L L L L RIFU</td>
</tr>
<tr>
<td>XREAD</td>
<td>H L L L L L RIFU</td>
</tr>
<tr>
<td>UNMAP</td>
<td>H L L L L L RIFU</td>
</tr>
<tr>
<td>FLUSH</td>
<td>H L L L L L RIFU</td>
</tr>
<tr>
<td>NOP</td>
<td>H L L L L L RIFU</td>
</tr>
<tr>
<td>DESELECT</td>
<td>H L L L L L RIFU</td>
</tr>
<tr>
<td>POWER DOWN ENTRY</td>
<td>H L L L L L RIFU</td>
</tr>
<tr>
<td>POWER DOWN EXIT</td>
<td>H L L L L L RIFU</td>
</tr>
<tr>
<td>ZQ Calibration Long</td>
<td>H L L L L L L V</td>
</tr>
<tr>
<td>ZQ Calibration Short</td>
<td>H L L L L L L V</td>
</tr>
</tbody>
</table>

Different protocols but can share the same wires

NVDIMM-P uses different protocols compared to DDR4, but they can share the same wires.
The NVDIMM-P Protocol

NVDIMM-P protocol invented to handle memory with low endurance

Non-deterministic credit based system allows time for bookkeeping

Out-of-order data returned with ID
Big Data Over -P Protocol

-P Protocol is NOT DRAM
Coexists by supporting DRAM timing and ODT decoding
NVDIMM-P Protocol extends the DDR interface to enable big data.

Out-of-order non-deterministic data allows for bookkeeping such as wear leveling.

Requires new CPU.

NVDIMM-P Capacity

Memory Demand

DRAM Capacity

DDR5 NVDIMM-P too
Software Issues

Symmetric solutions are simplest; no software changes, accept the performance you get.

Asymmetric solutions are more complicated, software partitioning required, many solution punt by mounting NVDIMM as an SSD.

- All NVDIMM-N: No problem, all memory persistent, all memory has same performance.
- All NVDIMM-P: No problem, all memory persistent, all memory has same performance.
- Mix of NVDIMM-N & DRAM: Complicates the solution. Software must separate persistent data from ephemeral data.
- Mix of NVDIMM-P & DRAM: NVDIMM-P can mount as extended memory with asymmetric performance or simply as SSD.
Advantage of Large Capacity PM

- **Persistent Main Memory**
  - Main Memory (e.g., DRAM)

- **Mass Storage** (SSD, etc)

- Much larger data sets align with increase in in-memory analysis memory requirements

- Al, data mining, etc

- Far fewer flushes to external mass storage

- Power fail safe
Operating systems have “new” hooks for persistent memory

Both disk mount and direct access enabled
Persistent memory has generated concerns about data security

Some systems prefer to encrypt in the CPU

NVDIMMs specifications adding on-DIMM encryption option

May be required for systems with DMA to the DRAM channel
Summary

Memory capacity demands exceeding DRAM roadmap

DRAM protocol limited to 16 Gb for DDR4, 32 Gb for DDR5

NVDIMM-N adds data persistence

NVDIMM-P allows media independent expansion

Software must deal with performance/feature asymmetry

Data encryption coming
Questions?

Bill Gervasi
Principal Systems Architect
bilge@Nantero.com