NVDIMM-N Cookbook:
A Soup-to-Nuts Primer on Using NVDIMM-Ns to Improve Your Storage Performance

Jeff Chang
VP Marketing and Business Development, AgigA Tech

Arthur Sainio
Director Marketing, SMART Modular
SNIA Legal Notice

- The material contained in this tutorial is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA Education Committee.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.
Abstract

Non-Volatile DIMMs, or NVDIMMs, have emerged as a go-to technology for boosting performance for next generation storage platforms. The standardization efforts around NVDIMMs have paved the way to simple, plug-n-play adoption. If you're a storage developer who hasn't yet realized the benefits of NVDIMMs in your products, then this tutorial is for you! We will walk you through a soup-to-nuts description of integrating NVDIMMs into your system, from hardware to BIOS to application software. We'll highlight some of the "knobs" to turn to optimize use in your application as well as some of the "gotchas" encountered along the way.

Learning Objectives

- Understand what an NVDIMM is
- Understand why an NVDIMM can improve your system performance
- Understand how to integrate an NVDIMM into your system
NVDIMM Cookbook
A User Guide that describes the building blocks and the interactions needed to integrate a NVDIMM into a system

- Part I
  - NVDIMM
- Part II
  - BIOS
- Part III
  - OS (Linux)
- Part IV
  - System Implementations & Use Cases
The “Ingredients”

CPU/Memory Controller

NVDIMM Cookbook

Approved SNIA Tutorial © 2015 Storage Networking Industry Association. All Rights Reserved.

NVDIMM-N

Power Supply

SMB (System Management Bus)

MRC + BIOS
(Memory Reference Code)

SAVE Trigger

DIMM Interface

Application

User Space

Load / Store

Kernel Space

OS

BIOS

NVDIMM

Platform Hardware

Software

Hardware

Block Mode

Block Driver

Byte Addressable

NVDIMM Driver

Energy Module

Kernel Space

Software

Hardware

Machine Readable Code

Load / Store

User Space

Application

System Management Bus

NVDIMM-N

Energy Module

Platform Hardware

CPU/Memory Controller

Power Supply
Part 1
NVDIMM
## JEDEC NVDIMM Taxonomy

Single letter designator - combines the media technology (NAND, etc) and the access mechanism (byte, block, etc.)

| NVDIMM-N | Memory mapped DRAM. Flash is not system mapped.  
| Memory mapped DRAM. Flash is not system mapped. |
| Access Methods -> direct byte- or block-oriented access to DRAM |
| Capacity = DRAM DIMM (1’s – 10’s GB) |
| Latency = DRAM (10’s of nanoseconds) |
| Energy source for backup |

| NVDIMM-F | Memory mapped Flash. DRAM is not system mapped.  
| Memory mapped Flash. DRAM is not system mapped. |
| Access Method -> block-oriented access through a shared command buffer, i.e. a mounted drive. |
| Capacity = NAND (100’s GB – 1’s TB) |
| Latency = NAND (10’s of microseconds) |

| NVDIMM-P | Memory mapped Flash and memory mapped DRAM |
| Memory mapped Flash and memory mapped DRAM |
| Supported -> Load/Store, Emulated Block |
| Two access mechanisms: persistent DRAM (–N) and also block-oriented drive access (–F) |
| Capacity = NVM (100’s GB – 1’s TB) |
| Latency = NVM (100’s of nanoseconds) |
NVDIMM-N How It Works

- Plugs into JEDEC Standard DIMM Socket
- Appears as standard RDIMM to host during normal operation
- Supercaps charge on power up
NVDIMM-N How It Works

• When health checks clear, NVDIMM can be armed for backup
• NVDIMM can be used as persistent memory space by the host
NVDIMM-N How It Works

- During unexpected power loss event, DRAM contents are moved to NAND Flash using Supercaps for backup power.
NVDIMM-N How It Works

- *When backup is complete, NVDIMM goes to zero power state*
- *Data retention = NAND Flash spec (typically years)*
NVDIMM-N How It Works

• *When power is returned, DRAM contents are restored from NAND Flash*
• *Supercaps re-charge in minutes*
NVDIMM-N How It Works

- **DRAM handed back to host in restored state prior to power loss**
- **Rinse and repeat**

Supercaps

Adapted from SNIA presentations by AgigA Tech
NVDIMM Entry Process using ADR (Asynchronous DRAM Re-fresh)

- Letters correspond to the timing diagram on the next page.
SAVE Operation

AC → System Power Supply → CPLD → Processor/Chipset → NVDIMM

A: AC
B: System DC
C: Powergood
D: SAVE_Trigger
E: SAVE_Trigger_n
F: Processor
G: NVDIMM Mode
H: SAVE

DC Holdup Time (1-12 msec)
NVDIMM-N DDR4 Platform HW Support/JEDEC Standardization

- DDR4 12V Power Pins (1, 145) standardized
- DDR4 SAVE_n Pin (230) standardized
  - Bi-directional SAVE_n to indicate SAVE completion
- EVENT_n asynchronous event notification
- I2C Device Addressing
- 12V in DDR4 simplifies NVDIMM power circuitry and cable routing
  - One cable needed between NVDIMM and BPM (Backup Power Module)
  - No cable needed if Host provides 12V backup power via DDR4 12V
# DDR4 Legacy vs. JEDEC Comparison

<table>
<thead>
<tr>
<th>Type</th>
<th>Features</th>
<th>1st Gen Legacy</th>
<th>2nd Gen JEDEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVDIMM/Firmware Hardware</td>
<td>NV controller registers controlled by Host via i2c</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DDR4 12V Power Pins (1,145)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>DDR4 SAVE_n Pin (230)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>NVDIMM Controller EVENT# Pin (78)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>SPD for NVDIMM representation</td>
<td>In Part number</td>
<td>JEDEC SPD</td>
</tr>
<tr>
<td></td>
<td>NV Controller registers</td>
<td>DDR3 compatible</td>
<td>JEDEC Registers</td>
</tr>
<tr>
<td></td>
<td>Memory Interface to Host</td>
<td>RDIMM</td>
<td>RDIMM/LRDIMM</td>
</tr>
<tr>
<td></td>
<td>JEDEC Raw Cards</td>
<td>None</td>
<td>LRDIMM</td>
</tr>
<tr>
<td>System/OS/BIOS/MRC</td>
<td>OS Driver (Block and Load/Store)</td>
<td>DDR3/4 compatible</td>
<td>New ACPI 6.0 and PMEM library compatible – Hardware Agnostic</td>
</tr>
<tr>
<td></td>
<td>NVDIMM Aware Kernel (Direct Access support)</td>
<td>Intel patch for 3.14</td>
<td>4.20 or higher – Hardware Agnostic</td>
</tr>
<tr>
<td></td>
<td>Intel MRC Changes to support NV Vendor</td>
<td>Yes - uses DDR3/4 MRC on Haswell</td>
<td>New MRC is required Hardware Agnostic</td>
</tr>
<tr>
<td></td>
<td>BIOS to support NV Vendor</td>
<td>Yes - Insyde/AMI support Intel MRC</td>
<td>New BIOS is required Hardware Agnostic</td>
</tr>
<tr>
<td></td>
<td>Direct Access (DAX) support for NVDIMM-N modules in Ext4</td>
<td>Yes</td>
<td>Yes - eliminates the page cache layer completely, Hardware Agnostic</td>
</tr>
<tr>
<td></td>
<td>OS NVDIMM Detection</td>
<td>E820 table type 12</td>
<td>ACPI 6.0 or higher/E820 table type 7</td>
</tr>
<tr>
<td></td>
<td>ADR support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>EVENT support – Output</td>
<td>Supplier dependent</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>SAVE_n support - Input</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>I2V support to connector - Input</td>
<td>Via Auxiliary</td>
<td>Yes</td>
</tr>
</tbody>
</table>
|                                           | I2V support Type                              | • Supercap input power | • Supercap input power • Backup power source
NVDIMM-N BIOS Support Functions

NVDIMMs rely on the BIOS/MRC (Memory Reference Code)

1. Detect NVDIMMs
2. Setup Memory Map
3. ARM for Backup
4. Detect AC Power Loss
5. Flush Write Buffers
6. RESTORE Data
   On Boot
7. Enable I2C R/W Access
Memory Reference Code (MRC) module provides the memory initialization procedure. This module is maintained by Intel (for Intel-based platforms of course) and released to all BIOS vendors.
NVDIMM Supported BIOS Flow

NVDIMM support: Major change in MRC module, minor change in E820 module
NVDIMM Restore/Recovery MRC Flow

1. **MRC Start**
   - Yes → **Save in Progress?**
   - No → **Train all DIMMs (including NVDIMMs) like standard DIMM**

2. **Save in Progress?**
   - Yes → **Configure Memory Controller to drive CKE low**
   - No → **Configure NVDIMM to Start Restore**

3. **Configure NVDIMM to Start Restore**
   - Yes → **Is Restore Complete?**
   - No → **Note Restore Status in MRC Output Structure**

4. **Is Restore Complete?**
   - Yes → **Note Restore Status in MRC Output Structure**
   - No → **Rewrite RC registers**

5. **Rewrite RC registers**
   - Yes → **Assert CKE**
   - No → **Rewrite MRS Registers**

6. **End of Restore Process**
   - Yes → **Run MemTest and MemInit on all DIMMs that didn’t have a successful restore**
   - No → **Did the NVDIMM Log an error?**

7. **Did the NVDIMM Log an error?**
   - Yes → **Note Error in output Structure**
   - No → **Does NVDIMM contain valid Data?**

8. **Does NVDIMM contain valid Data?**
   - Yes → **End of Restore Process**
   - No → **Did the NVDIMM Log an error?**
E820 Table Example

- E820 is shorthand to refer to the facility by which the BIOS of x86-based computer systems reports the memory map to the operating system or boot loader.

```
[root@localhost Desktop]# dmesg | grep e820
BIOS-e820: 0000000000000000 - 0000000000000000 (usable)
BIOS-e820: 0000000000000000 - 0000000000000000 (reserved)
BIOS-e820: 0000000000000000 - 0000000000000000 (reserved)
BIOS-e820: 0000000000000000 - 0000000000000000 (usable)
BIOS-e820: 0000000000000000 - 0000000000000000 (reserved)
BIOS-e820: 0000000000000000 - 0000000000000000 (reserved)
BIOS-e820: 0000000000000000 - 0000000000000000 (ACPI data)
BIOS-e820: 0000000000000000 - 0000000000000000 (ACPI NVS)
BIOS-e820: 0000000000000000 - 0000000000000000 (reserved)
BIOS-e820: 0000000000000000 - 0000000000000000 (usable)
BIOS-e820: 0000000000000000 - 0000000000000000 (ACPI NVS)
BIOS-e820: 0000000000000000 - 0000000000000000 (usable)
BIOS-e820: 0000000000000000 - 0000000000000000 (reserved)
BIOS-e820: 0000000000000000 - 0000000000000000 (reserved)
BIOS-e820: 0000000000000000 - 0000000000000000 (reserved)
BIOS-e820: 0000000000000000 - 0000000000000000 (reserved)
BIOS-e820: 0000000000000000 - 0000000000000000 (reserved)
```

Note: ACPI 6.0 defines Type 7 for Persistent Memory and NFIT
Additional BIOS Considerations

- BIOS also presents various menu options to setup NVDIMM operation
- Examples:
  - Enable ADR
  - Enable RESTORE
  - Enable ARM in BIOS
  - Write Cache options
Legacy vs JEDEC I2C Register Implementation

- BIOS implementations for DDR3 platforms and prior were specific to an NVDIMM vendor’s command set (although high level commands were common)
- Early DDR4 platforms also followed this same basic method. BIOS with MRC 1.10 to 1.14 all have Vendor Specific I2C support
- JEDEC I2C release date and MRC version not determined
- MRC with JEDEC I2C Register Support will most likely also include BIOS support for ACPI 6.0, NFIT (NVDIMM Firmware Interface Table), and DSM (Driver Specific Method), cf. http://pmem.io
Part 3
OS (Linux)
Memory Mapped File Programming Model

With Disks

User
- Application

Kernel
- File System
- Driver
- File system cache

HW
- Disk

With PM

User
- Application

Kernel
- File System
- Driver

HW
- Persistent Memory

Source: SNIA NVM Programming TWG
Linux NVDIMM Software Architecture

User Space
- Management UI
- Management Library

Application
- Standard Raw Device Access
- Standard File API

Block
- BTT (optional)
- File System

Kernel Space
- NFIT Core
- Block Window Driver
- PMEM Block Driver

Kernel Space
- DAX Enabled FS
- MMU

Commands
- Block I/O

NFIT Compatible NVDIMM

ACPI NFIT

4.2 Kernel

Intel® GIT Hub

ACPI 6.0 Compatible

Existing File Systems
What’s available in Linux 4.2 Kernel?

DAX
Enabled FS

EXT4 with "-o dax" support

BTT (Block Translation Table)

Built in Kernel driver nd_btt.ko.
Source: drivers/nvdimm/btt.c

Block Window Driver

Built in Kernel driver nd_blk.ko.
Source: drivers/nvdimm/blk.c

PMEM Block Driver

Built in Kernel driver nd_pmem.ko.
Source: drivers/nvdimm/pmem.c

NFIT Core

Built in Kernel driver core.ko.
Source: drivers/nvdimm/core.c
Linux Work in progress

NVM Library

Management Library

Under Intel’s initiative, being worked on. ([http://pmem.io](http://pmem.io)).

Support for block, object, log, virtual memory logic. Supports Load/Store primitives.

Release vehicle not known.

References

- [https://www.kernel.org/](https://www.kernel.org/)

- [http://pmem.io](http://pmem.io)


NVDIMM System Examples

• Intel NVDIMM support not POR but HW is enabled and BIOS available via SOW
NVDIMM-N DDR4 Platform

Energy Source Options

- JEDEC JC45.6 Byte Addressable Energy Backed Interface

Figure 1: NVDIMM overview
Population Rules

- There are no NVDIMM specific population rules
  - Normal DIMM population rules still apply (ex RDIMMs and LRDIMMs can’t be mixed)
  - NVDIMMs and normal DIMMs may be mixed in the same channel
  - NVDIMMs from different vendors may be mixed in the same system and even the same channel.
- How the DIMMs are installed in a system will affect performance, so thought should be put into how DIMMs are populated
- NVDIMM population tips
  - Interleaving DIMMs within a channel provides a very small performance benefit
  - Interleaving DIMMS across a channel provides a very large performance benefit
  - Two DIMMs of the same type should not be installed in the same channel unless all other channels in the system have at least one of that type DIMM.
Example Optimal Interleaves

- Has a 4-way Interleave between normal DIMMs, and optionally a 2-way interleave between the NVDIMMs

- Has a 4-way Interleave between normal DIMMs, and optionally a 4-way interleave between the NVDIMMs

- Has a 2-way Interleave between normal DIMMs, and optionally a 2-way interleave between the NVDIMMs

Source: Intel
Use Cases

• **In Memory Database:** Journaling, reduced recovery time, Ex-large tables

• **Traditional Database:** Log acceleration by write combining and caching

• **Enterprise Storage:** Tiering, caching, write buffering and meta data storage without an auxiliary power source

• **Virtualization:** Higher VM consolidation with greater memory density

• **High-Performance Computing:** Check point acceleration and/or elimination

• **NVRAM Replacement:** Higher performance enabled by removing the DMA setup/teardown

• **Other:** Object stores, unstructured data, financial & real-time transactions
Application Example: Storage Bridge Bay (SBB)

Shadow Writes Required for Failover

Adapted from SNIA presentations by AgigA Tech
SBB: A Simpler/Better/Faster Way

Also a better alternative to Cache-to-Flash implementations:
- Separate failure domain
- No battery maintenance
- System hold-up requirements significantly less severe
- 4x write latency performance improvement

Adapted from SNIA presentations by AgigA Tech
Advantages of NVDIMMs for Applications

Legacy HDD/SSD Solution
- Persistent data stored in HDD or SSD tiers
- Slow & unpredictable software stack

NVDIMM Solution
- Persistent data stored in fast DRAM tier
- Removes software stack from data-path

Accelerates SW-Apps!
- DRAM class latency & thru-put for persistent data
  - 1000X lower latency
  - 10X+ throughput increase

• The value is in application acceleration
Thank You!
The SNIA Education Committee thanks the following Individuals for their contributions to this Tutorial.

Authorship History
Jeff Chang/Arthur Sainio - June 2015

Additional Contributors
Mario Martinez - July 2015

Please send any questions or comments regarding this SNIA Tutorial to tracktutorials@snia.org