Fujitsu Corporate Profile
Japan’s largest IT services provider and No. 5 in the world. (based on vendor revenue 2015)*

We do everything in ICT. We use our experience and the power of ICT to shape the future of society with our customers.

156,000 Fujitsu people support customers in more than 100 countries.

FORTUNE named Fujitsu as one of ‘the World's Most Admired Companies’ for the fourth consecutive year.

* Source: Gartner, "Market Share: IT Services, 2015" 06 April 2016 (GJ16139)
"Through our constant pursuit of innovation, the Fujitsu Group aims to contribute to the creation of a networked society that is rewarding and secure, bringing about a prosperous future that fulfills the dreams of people throughout the world."

Excerpt from FUJITSU Way
A worldwide network to support our customers

‘shaping tomorrow with you’ wherever you are.

Approximately 155,000 Fujitsu colleagues working with customers in over 100 countries

As of March 2017

* EMEIA - Europe, Middle East, India and Africa
Technology
Memory Storage Hierarchy

- CPU
- CPU Caches
- DRAM (Memory)
- Flash (Storage)
- HDD (Storage)

- Byte
- Load, Store
- Block
- I/O

- Volatile
- Non Volatile

Capacity
Cost
Speed
Memory Storage Hierarchy – Paradigm Shift

- **CPU Caches**
- **DRAM (Memory)**
- **Persistent Memory**
- **Flash (Storage)**
- **HDD (Storage)**

- **CPU**
- **Capacity** (Cost)
- **Data Access**
  - Byte
  - Load, Store
- **Performance** (Speed)
- **Non Volatile**
- **Volatile**
PERSISTENT MEMORY PLATFORM SUPPORT

Direct Load/Store Access
Native Persistence
128, 256, 512GB
DDR4 Pin Compatible

- BIOS
- OPERATING SYSTEM
- SNIA NVM PROGRAMMING MODEL
- APPLICATION

LOW LATENCY SYSTEM ACCESS TO PERSISTENT MEMORY

IDLE AVERAGE RANDOM READ LATENCY¹

10μs FOR 4KB
~100 ns TO ~350 ns FOR 64B²

¹ Source: Intel-tested. Average read latency measured at queue depth 1 during 4k random write workload. Measured using FIO 3.1, comparing Intel Reference platform with Optane™ SSD DC P4800X 375GB and Intel® SSD DC P4600 1.6TB compared to SSDs commercially available as of July 1, 2018. Performance results are based on testing as of July 24, 2018 and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

² App Direct Mode, NeonCity, LBB 61 chipset, CLX 90z20 Core (QDF G0Y7Z), Memory Config 192GB DDR4 (per socket) DDR2666 MT/s, Optane DCPMM 128GB, 865G i5 660, SKC version WW48.8 BKC, Linux OS 4.18.8-100.fc27, Spectre/Meltdown Patched (1,2,3,4)
INTEL® OPTANE™ PERSISTENT MEMORY: APP DIRECT

App Direct Mode provides the persistent memory programming model
- Reported to OS by ACPI
- Linux and Windows expose via "DAX" file systems

Several use cases supported by OS & PMDK APIs
- Persistent memory, non-paged
  (no DRAM footprint when accessed)
- Volatile App Direct, an explicit pool of volatile memory
- Storage over App Direct, a very fast SSD built on persistent memory

Memory Mode provides familiar volatile memory programming model

- Additional layer of caching: DRAM as WB cache
- Hardware managed, software sees very high capacity memory (6 TB)

Range of use cases supported

- No software change – big memory
- Applications/Algorithms changes for new hierarchy/capacity

Developer placed data structures

“SAP HANA knows which data structures benefit most from persistent memory. SAP HANA automatically detects persistent memory hardware and adjusts itself by automatically placing these data structures on persistent memory, while all others remain in DRAM”

- Column Store Main in Persistent Memory
  - 90% of the data footprint
  - Nonvolatile – no initial load time
- High perf, volatile in DRAM
- SSDs still used for row store, column delta, replication, backups...

Source: “SAP HANA & Persistent Memory”
  - Andreas Schuster

---
3 App Direct Mode: XeonCity, Xeon-B3 chipset, CLX 85 26 Core (GDF QG2), Memory Conf 102GB DDR4 (per socket), DDR 2666 MT/s, Intel® Optane Data Center Persistent Memory 128GB, BIOS 561.D09, I0K version W44B.5 I0K, Linxus OS 4.18.8-100.fc27, Spectre/Meltdown Patched (1,2,3, 3a) © 2018 Storage Networking Industry Association. All Rights Reserved
PMEM attributes

- Enables a fundamental change in computing architecture
- Standardized through NFIT and JEDEC
- Linux 4.4+ kernels and Windows2016 have the software stack
- Dramatically increases system performance
- Bridges the gap between DRAM and Flash
- Apps, middleware and OSs are no longer bound by file system overhead in order to run persistent transactions
- NVDIMMs BIOS/MRC (Memory Reference Code)
- Open source library is available for applications
Use Cases

- Execute in place for *.ko, *.so, ...
- In Memory Database
  - Journaling, reduced recovery time, tables
- Traditional Database
  - Log acceleration by write combining, caching
- Enterprise Storage
  - Tiering, caching, write buffering and meta data storage
- High-Performance Computing
  - Check point acceleration and/or elimination
- Analytics, AI, HPC
Technology

Setup and Configuration
Linux 4.9+

*Courtesy of Dan Williams <dan.j.williams@intel.com>

Persistent Memory Tools

Configuration
- Pre-boot
  - ipmctl
- Linux
  - ipmctl
  - ndctl
- Windows
  - ipmctl
  - New-StoragePool
  - New-Volume

Benchmark
- Intel® Memory Latency Checker (MLC)
- FIO (Flexible IO Tester)
- pmembench

Analysis
- Intel® VTune Amplifier
  - Memory Analyzer
  - Storage Analyzer
- Intel® Persistent Inspector
- Intel® VTune Platform Profiler
- pmempool
- pmemcheck
- Valgrind
Provisioning Terms & Concepts

Non-Interleaved Regions

Regions are created within [non]interleaved sets. Interleaving can be 1 to n-way mapping.

Interleaved Regions

Creates contiguous physical address space and provides striped reads/writes for better throughput.

Namespaces

Similar to SSD, raw capacity of a region is partitioned into one or more logical devices called namespaces.

Exposing Persistent Memory to Applications

Filesystem DAX (FSDAX)

Persistent Memory Pool(s): persistent memory is exposed by the OS to the application as memory-mapped files when using PMDK.

Direct Access (DAX) Filesystem: For file mappings (mmap), the storage device is mapped directly into user space and bypasses page cache.

/dev/pmem: a device used to create a filesystem.
Persistent Memory Pools

- Intended for use on DAX File System
- Pools are tagged with a ‘layout’ name/substring for identification
- Support for multiple pools per Application
- Pools can be aggregated into ‘pool sets’ to provide a larger address space and replication
- Easy backup/restore

We use BIOS to detect memory modules.

- BIOS could be used to define the operation mode (AD, MM or mixed mode) – but we do not use BIOS for that purpose.

- instead: the operation mode (AD, MM or mixed mode) is implied by the way we define allocation goals using Linux `ipmctl`.

- we could use standalone UEFI shell for all configuration steps – but FJ does not use it at all.

---

**Configuration overview**

<table>
<thead>
<tr>
<th>app_direct_mode</th>
<th>Support for accessing Apache Pass DIMM persistent memory in App Direct mode. In App Direct mode, Apache Pass DIMMs and DDR act as independent memory resources under direct load/store control of an application. <em>(One-Level Memory, or 1LM)</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1LM = AD</td>
<td>BIOS could be used to define the operation mode (AD, MM or mixed mode) – but we do not use BIOS for that purpose. Instead: the operation mode (AD, MM or mixed mode) is implied by the way we define allocation goals using Linux <code>ipmctl</code>.</td>
</tr>
<tr>
<td>memory_mode</td>
<td>Support for accessing Apache Pass DIMM capacity in memory mode where Apache Pass DIMMs act as system memory under the control of the operating system. In Memory mode, any DDR in the platform will act as a cache working in conjunction with the Apache Pass DIMMs. In some earlier documentation, this mode was referred to as <em>Two-Level Memory, or 2LM</em>.</td>
</tr>
<tr>
<td>2LM = MM</td>
<td></td>
</tr>
</tbody>
</table>

**persistent memory concepts**

<table>
<thead>
<tr>
<th>BIOS detect</th>
<th>AEP memory modules</th>
<th>AD or MM or mixed mode</th>
<th>allocation goals</th>
<th>regions</th>
<th>namespaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>n/a</td>
<td>X</td>
<td>X</td>
<td>implied by allocation goal</td>
<td>define</td>
<td>administrate</td>
</tr>
<tr>
<td>n/a</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>n/a</td>
<td>administrate</td>
</tr>
<tr>
<td>n/a</td>
<td>X</td>
<td>X</td>
<td>n/a</td>
<td>n/a</td>
<td>define and administrate</td>
</tr>
</tbody>
</table>
Test system PRIMERGY

- DRAM 32 GB
- DCPMM 256 GB
- empty

rx24-1  RX2540-M4
CPU     Xeon 82xx, 24C, 2.6GHz
BIOS    R1.3.1 for D3384-B1x
Configuration – DIMM type and location

[root@rx24-1 ~]# ipmctl show -dimm

<table>
<thead>
<tr>
<th>DimmID</th>
<th>Capacity</th>
<th>HealthState</th>
<th>ActionRequired</th>
<th>LockState</th>
<th>FWVersion</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0001</td>
<td>252.4 GiB</td>
<td>Healthy</td>
<td>0</td>
<td>Disabled</td>
<td>01.01.00.5212</td>
</tr>
<tr>
<td>0x0101</td>
<td>252.4 GiB</td>
<td>Healthy</td>
<td>0</td>
<td>Disabled</td>
<td>01.01.00.5212</td>
</tr>
<tr>
<td>0x1001</td>
<td>252.4 GiB</td>
<td>Healthy</td>
<td>0</td>
<td>Disabled</td>
<td>01.01.00.5212</td>
</tr>
<tr>
<td>0x1101</td>
<td>0.0 GiB</td>
<td>Non-functional</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

[root@rx24-1 ~]# ipmctl show -topology

<table>
<thead>
<tr>
<th>DimmID</th>
<th>MemoryType</th>
<th>Capacity</th>
<th>PhysicalID</th>
<th>DeviceLocator</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0001</td>
<td>Logical Non-Volatile Device</td>
<td>252.4 GiB</td>
<td>0x0031</td>
<td>CPU1_DIMM_A2</td>
</tr>
<tr>
<td>0x0101</td>
<td>Logical Non-Volatile Device</td>
<td>252.4 GiB</td>
<td>0x003d</td>
<td>CPU1_DIMM_D2</td>
</tr>
<tr>
<td>0x1001</td>
<td>Logical Non-Volatile Device</td>
<td>252.4 GiB</td>
<td>0x0049</td>
<td>CPU2_DIMM_A2</td>
</tr>
<tr>
<td>0x1101</td>
<td>Logical Non-Volatile Device</td>
<td>0.0 GiB</td>
<td>0x0055</td>
<td>CPU2_DIMM_D2</td>
</tr>
<tr>
<td>N/A</td>
<td>DDR4</td>
<td>32.0 GiB</td>
<td>0x002f</td>
<td>CPU1_DIMM_A1</td>
</tr>
<tr>
<td>N/A</td>
<td>DDR4</td>
<td>32.0 GiB</td>
<td>0x0033</td>
<td>CPU1_DIMM_B1</td>
</tr>
<tr>
<td>N/A</td>
<td>DDR4</td>
<td>32.0 GiB</td>
<td>0x0036</td>
<td>CPU1_DIMM_C1</td>
</tr>
<tr>
<td>(...)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Configuration – DIMM location and nmem devices

```
[root@rx24-1 ~]# ndctl list --dimms -u
{
    "dev":"nmem1",
    "id":"8089-a2-1834-0000173b",
    "handle":"0x101",
    "phys_id":"0x3d"
},
{
    "dev":"nmem0",
    "id":"8089-a2-1834-000016b0",
    "handle":"0x1",
    "phys_id":"0x31"
},
{
    "dev":"nmem2",
    "id":"8089-a2-1834-00001a91",
    "handle":"0x1001",
    "phys_id":"0x49"
}
```

```
[root@rx24-1 ~]# ls -l /dev/nmem*
crw------- 1 root root 251, 0 Mar 20 09:52 /dev/nmem0
crw------- 1 root root 251, 1 Mar 20 09:52 /dev/nmem1
crw------- 1 root root 251, 2 Mar 20 09:52 /dev/nmem2
crw------- 1 root root 251, 3 Mar 20 09:52 /dev/nmem3
```

```
[root@rx24-1 ~]# grep 251 /proc/devices
251 dimmctl
```
Configuration – Create Goal (1)

- `ipmctl-create-goal` - Creates a memory allocation goal on one or more DCPMM

  Creates a memory allocation goal on one or more for the BIOS to read on the next reboot in order to map the DCPMM capacity into the system address space. Persistent memory can then be utilized by creating a namespace.

  `ipmctl create [-help|-h] [-force|-f] [-units|-u (B|M|MiB|GB|GiB|TB|TiB)] [-output|-o (text|nvmxml)] [-dimm [[(DimmIDs)]] -goal [-socket (SocketIDs)] [MemoryMode=(0|)] [PersistentMemoryType=(AppDirect|AppDirectNotInterleaved)] [Reserved=(0|)] [NamespaceLabelVersion=(1.1|1.2)]`

[root@rx24-1 ~]# ipmctl create -dimm 0x0000,0x0101 -goal PersistentMemoryType=AppDirect

The following configuration will be applied:

<table>
<thead>
<tr>
<th>SocketID</th>
<th>DimmID</th>
<th>MemorySize</th>
<th>AppDirect1Size</th>
<th>AppDirect2Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>0x0001</td>
<td>0.0 GiB</td>
<td>252.0 GiB</td>
<td>0.0 GiB</td>
</tr>
<tr>
<td>0x0000</td>
<td>0x0101</td>
<td>0.0 GiB</td>
<td>252.0 GiB</td>
<td>0.0 GiB</td>
</tr>
</tbody>
</table>

Do you want to continue? [y/n] y

Created following region configuration goal

<table>
<thead>
<tr>
<th>SocketID</th>
<th>DimmID</th>
<th>MemorySize</th>
<th>AppDirect1Size</th>
<th>AppDirect2Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>0x0001</td>
<td>0.0 GiB</td>
<td>252.0 GiB</td>
<td>0.0 GiB</td>
</tr>
<tr>
<td>0x0000</td>
<td>0x0101</td>
<td>0.0 GiB</td>
<td>252.0 GiB</td>
<td>0.0 GiB</td>
</tr>
</tbody>
</table>

A reboot is required to process new memory allocation goals.
## Configuration – Create Goal (2)

```
[root@rx24-1 ~]# ipmctl show -region
SocketID | ISetID | PersistentMemoryType | Capacity | FreeCapacity | HealthState
----------|-------|-----------------------|----------|--------------|-------------
0x0000    | 0x8baeeb8535e2444 | AppDirect           | 504.0 GiB | 504.0 GiB    | Healthy     
0x0001    | 0x7fe0da90c5328a22 | AppDirectNotInterleaved | 252.0 GiB | 252.0 GiB    | Healthy     
```

```
[root@rx24-1 ~]# ipmctl show -dimm 0x1001 -pcd Config | grep Dimm
---DimmID=0x0001---
NumOfDimmsInInterleaveSet : 0x2
DimmUniqueIdentifier : 8089-a2-1834-000016b0
DimmUniqueIdentifier : 8089-a2-1834-0000173b
(...)
---DimmID=0x0101---
NumOfDimmsInInterleaveSet : 0x2
DimmUniqueIdentifier : 8089-a2-1834-000016b0
DimmUniqueIdentifier : 8089-a2-1834-0000173b
(...)
---DimmID=0x1001---
NumOfDimmsInInterleaveSet : 0x1
DimmUniqueIdentifier : 8089-a2-1834-00001a91
```
ndctl create-namespace - Create a maximally sized pmem namespace in 'fsdax' mode

- **-r, --region <region-id>** limit namespace to a region with an id or name of <region-id>
- **-v, --verbose** emit extra debug messages to stderr
- **-e, --reconfig <reconfig namespace>** reconfigure existing namespace
- **-n, --name <name>** specify an optional free form name for the namespace
- **-s, --size <size>** specify the namespace size in bytes (default: available capacity)
- **-m, --mode <operation-mode>** specify a mode for the namespace, 'sector', 'fsdax', 'devdax' or 'raw'
- **-M, --map <memmap-location>** specify 'mem' or 'dev' for the location of the memmap
- **-l, --sector-size <lba-size>** specify the logical sector size in bytes
- **-t, --type <type>** specify the type of namespace to create 'pmem' or 'blk'

```
[root@rx24-1 ~]# ndctl create-namespace  -v -r region0 -m fsdax -t pmem  -s 479857737728
{
    "dev": "namespace0.0",
    "mode": "fsdax",
    "map": "dev",
    "size": 472356225024,
    "uuid": "8232e3e1-cafd-43d6-bdc9-fb6d9aa9b594",
    "sector_size": 512,
    "align": 2097152,
    "blockdev": "pmem0"
}
```
Configuration – List Namespaces (1)

```bash
[root@rx24 ~]# ndctl list -D -R -u
(...)
{
  "dev":"region1",
  "size":"252.00 GiB (270.58 GB)",
  "available_size":0,
  "max_available_extent":0,
  "type":"pmem",
  "iset_id":"0x7fe0da90c5328a22",
  "mappings":[
    {
      "dimm":"nmem2",
      "offset":"0x10000000",
      "length":"0x3f00000000",
      "position":0
    }
  ],
  "persistence_domain":"memory_controller"
},
{
  "dev":"region0",
  "size":"504.00 GiB (541.17 GB)",
  "available_size":"504.00 GiB (541.17 GB)",
  "max_available_extent":"504.00 GiB (541.17 GB)",
  "type":"pmem",
  "iset_id":"0x8baaeeb8535e2444",
  "mappings":[
    {
      "dimm":"nmem1",
      "offset":"0x10000000",
      "length":"0x3f00000000",
      "position":1
    },
    {
      "dimm":"nmem0",
      "offset":"0x10000000",
      "length":"0x3f00000000",
      "position":0
    }
  ],
  "persistence_domain":"memory_controller"
}
```
Configuration – List Namespaces (2)

```
[root@rx24-1 ~]# ndctl list -N -B -D -R -u
(...)
"dev":"region3",
"size":"50.00 GiB (53.69 GB)",
"type":"pmem",
"persistence_domain":"unknown",
"namespaces":[
  {
    "dev":"namespace3.0",
    "mode":"fsdax",
    "map":"mem",
    "size":"50.00 GiB (53.69 GB)",
    "sector_size":512,
    "blockdev":"pmem3"
  }
(...)
BOOT_IMAGE=/vmlinuz-4.20.16-200.fc29.x86_64 root=/dev/mapper/fedora-root resume=/dev/fedora/swap quiet showopts
```

```
crashkernel=202M,high
memmap=50G!4G memmap=50G!1176G
memmap=nn[KMG]!ss[KMG]

[KNL,X86] Mark specific memory as protected. Region of memory to be used, from ss to ss+nn. The memory region may be marked as e820 type 12 (0xc) and is NVDIMM or ADR memory.
```
Configuration – memmap= ... (1)

Dec 11 10:56:47 rx24-2 kernel: BIOS-provided physical RAM map:
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x0000000000000000-0x0000000009bff] usable
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x0000000009c00-0x000000000ffff] reserved
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x0000000e0000-0x00000000000ffff] reserved
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x00000000652ba000-0x0000000067c88fff] reserved
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x0000000067c89000-0x0000000067df0fff] ACPI data
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x0000000067df1000-0x000000006d893fff] ACPI NVS
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x000000006d894000-0x000000006f367fff] reserved
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x000000006f368000-0x000000006f7fffff] usable
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x000000006f800000-0x000000008fffffff] reserved
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x00000000fed20000-0x00000000fed44fff] reserved
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x00000000ff000000-0x00000000ffffffff] reserved
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x0000000100000000-0x000000306fffffff] usable
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x0000000100000000-0x00000000306fffffff] reserved
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x0000000307000000-0x000000ab2fffffff] persistent (type 7)
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x000000ab3000000-0x000000d32fffffff] usable
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x000000d33000000-0x000001c8affffffff] persistent (type 7)
Dec 11 10:56:47 rx24-2 kernel: BIOS-e820: [mem 0x000001c8b0000000-0x000001c8c7ffffff] reserved
Configuration – memmap= ... (2)

Dec 11 10:56:47 rx24-2 kernel: NX (Execute Disable) protection: active
Dec 11 10:56:47 rx24-2 kernel: user-defined physical RAM map:
Dec 11 10:56:47 rx24-2 kernel: user: [mem 0x0000000000000000-0x000000000009bff] usable
Dec 11 10:56:47 rx24-2 kernel: user: [mem 0x000000000009c00-0x00000000000ffff] reserved
Dec 11 10:56:47 rx24-2 kernel: user: [mem 0x000000000100000-0x00000000052bff] usable
Dec 11 10:56:47 rx24-2 kernel: user: [mem 0x00000000052c000-0x0000000007bff] reserved
Dec 11 10:56:47 rx24-2 kernel: user: [mem 0x0000000007bff-0x0000000007dffff] ACPI data
Dec 11 10:56:47 rx24-2 kernel: user: [mem 0x0000000007df1000-0x000000000d4ff] ACPI NVS
Dec 11 10:56:47 rx24-2 kernel: user: [mem 0x000000000d4f4000-0x000000000e000000] reserved
Dec 11 10:56:47 rx24-2 kernel: user: [mem 0x000000000e000000-0x0000000000100000] persistent (type 12)
Dec 11 10:56:47 rx24-2 kernel: user: [mem 0x000000000fd000000-0x000000000fe7ffff] reserved
Dec 11 10:56:47 rx24-2 kernel: user: [mem 0x000000000fed20000-0x000000000fed44ff] reserved
Dec 11 10:56:47 rx24-2 kernel: user: [mem 0x000000000fed44ff-0x000000000fffffff] reserved
Dec 11 10:56:47 rx24-2 kernel: user: [mem 0x0000000100000000-0x0000000d7fffffff] persistent (type 12)
Configuration – File Systems

```
[root@rx24-1 ~]# ll /dev/pmem*
brw-rw---- 1 root disk 259, 5 Mar 21 09:35 /dev/pmem0
brw-rw---- 1 root disk 259, 4 Mar 20 09:52 /dev/pmem1
brw-rw---- 1 root disk 259, 7 Mar 21 09:37 /dev/pmem2
[root@rx24-1 ~]# mkfs -t ext4 /dev/pmem0
[root@rx24-1 ~]# mkfs -t ext4 /dev/pmem1
[root@rx24-1 ~]# mkfs -t ext4 /dev/pmem2
[root@rx24-1 ~]# mount -o dax /dev/pmem0 /mnt/pmem0
[root@rx24-1 ~]# mount -o dax /dev/pmem1 /mnt/pmem1
[root@rx24-1 ~]# mount -o dax /dev/pmem2 /mnt/pmem2
[root@rx24-1 ~]# mount | grep pm
/dev/pmem0 on /mnt/pmem0 type ext4 (rw,relatime,dax)
/dev/pmem1 on /mnt/pmem1 type ext4 (rw,relatime,dax)
/dev/pmem2 on /mnt/pmem2 type ext4 (rw,relatime,dax)
[root@rx24-1 ~]# df -k | egrep 'pm|File'
Filesystem  1K-blocks  Used  Available  Use% Mounted on
/dev/pmem0   452995648  73756  429841240   1% /mnt/pmem0
/dev/pmem1   254978620 36955404  205001316  16% /mnt/pmem1
/dev/pmem2   51343840  53272  48652744   1% /mnt/pmem2
```
- Technology
- Setup and Configuration
- Software Programming Stack
Intel® DIMMs: Broad Ecosystem Enabling

Open Interfaces and Tools

- SNIA: Standard NVM Programming Model
- 10th Anniversary: Standard NVDIMM Platform Interface ACPI 6.0
- www.PMEM.io: Open Source NVM Enabling (NVML)

Operating Environments

- Microsoft
- Linux® Kernel 4.4†
- fedora
- redhat
- SUSE
- vmware
- Future vSphere® Release†

Public Statements of Support for Products or Programming Model

- cloudera
- Microsoft
- ORACLE®
- redhat
- SAP
- vmware

† Planned. Subject to change
Open NVM Programming Model

50+ Member Companies

SNIA Technical Working Group
Initially defined 4 programming modes required by developers

Spec 1.0 developed, approved by SNIA voting members and published

Interfaces for PM-aware file system accessing kernel PM support
Interfaces for application accessing a PM-aware file system
Kernel support for block NVM extensions
Interfaces for legacy applications to access block NVM extensions

NVM-Libraries & Drivers
NVMe Block Interface
DCPMM as Storage SW Architecture

- Management
  - Management UI
  - Management Library

- Storage over App Direct
  - Application
  - Standard Raw Device Access
  - Standard File API
  - File System
  - OS Native NVDIMM Driver

- App Direct
  - Application
  - Standard File API
  - Load/Store
  - User Space
  - Kernel Space
  - Persistent Memory Aware File System
  - MMU Mappings

- I/O with OS Buffer cache
- mov with CPU Lx cache

- Memory Mode
  - Persistent Region
    - (Used in App Direct mode and Storage Over App Direct)
  - Storage Over AD
  - DCPMM Storage Mode

- Apache Pass
- DCPMM specific OS components
The Data Path

Core
L1
L1
L2
L2
Core
L1
L1
L2
L2
Core
L1
L1
L2
L2
Core
L1
L1
L2
L2
Memory Controller
NV-DIMM / PMEM
Memory Controller
NV-DIMM / PMEM
MOV
The Data Path

MOV

Core
L1
L1
L2

Core
L1
L1
L2

Core
L1
L1
L2

Core
L1
L1
L2

Memory Controller
NV-DIMM / PMEM

Memory Controller
NV-DIMM / PMEM

CLFLUSH
CLFLUSHOPTH
CLWB

ADR = Flush the WPQ automatically on power-fail or shutdown
Caskade Lake CPU - lscpu

Model name: Intel(R) Xeon(R) Platinum 8270 CPU @ 2.70GHz
CPU(s): 104
Thread(s) per core: 2
Core(s) per socket: 26
Socket(s): 2
CPU MHz: 1000.061
CPU max MHz: 4000.0000
CPU min MHz: 1000.0000
BogoMIPS: 5400.00
L1d cache: 32K
L1i cache: 32K
L2 cache: 1024K
L3 cache: 36608K
NUMA node0 CPU(s): 0-3,7-9,13-15,20-22,52-59,61-65,67-72-74
.Flags: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology
nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pccid dca sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb cat_l3 cdp_l3 invpcid_single intel_pipn ssbd mba ibrs ibpb stibp tpr_shadow vmvx flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm mpx rdt_a avx512f avx512dq rdseed adx smap clflushopt clwb intel_pt avx512cd avx512bw avx512vl xsaveopt xsavec xsetbv1 xsaves cqmm_l1c cqmm_occupp_l1c cqmm_mgmt_total cqmm_mgmt_local dtherm ida arat pln pts hwp hwp_act_window hwp_epp hwp_pkg_req pku ospke avx512_vnni flush_l1d arch_capabilities
Linux Kernel Layers for PMEM
Linux 4.2 + subsystems added support of NVDIMMs. Mostly stable from 4.4

NVDIMM modules presented as device links: /dev/pmem0, /dev/pmem1

QEMU support (experimental)

XFS-DAX and EXT4-DAX available

File system extensions to bypass the page cache and block layer to memory map persistent memory, from a PMEM block device, directly into a process address space.

nd_btt.ko  Block Translation Table: Persistent memory is byte addressable. Existing software may have an expectation that the power-fail-atomicity of writes is at least one sector, 512 bytes. The BTT is an indirection table with atomic update semantics to front a PMEM/BLK block device driver and present arbitrary atomic sector sizes.

nd_pmem.ko  A system-physical-address range where writes are persistent. A block device composed of PMEM is capable of DAX. A PMEM address range may span an interleave of several DIMMs.

nd_blk.ko  A set of one or more programmable memory mapped apertures provided by a DIMM to access its media. This indirection precludes the performance benefit of interleaving, but enables DIMM-bounded failure modes.
NVM Library: pmem.io
64-bit Linux* Initially

What does this mean?

NVM Libraries (optional)

- Open Source
  - [https://pmem.io/pmdk](https://pmem.io/pmdk)
  - libpmem
  - libpmemobj
  - libpmemblk
  - libpmemlog
  - libvmem
  - libvmmmalloc

Intel DIMM

Kernel Space

pmem-Aware File System

User Space

Application

Standard File API

Load/Store
Programming example with libpmem

```c
int main(int argc, char *argv[]) {
  /* create a pmem file and memory map it */
  if (((pmemaddr = pmem_map_file("/mnt/pmem0/my-file", 4096, PMEM_FILE_CREATE, 0666, &mapped_len, &is_pmem)) == NULL) {
    perror("pmem_map_file");
    exit(1);
  }

  /* store a string to the persistent memory */
  strcpy(pmemaddr, "hello, persistent memory");

  /* flush above strcpy to persistence */
  if (is_pmem)
    pmem_persist(pmemaddr, mapped_len);
  else
    pmem_msync(pmemaddr, mapped_len);

  /* Delete the mappings. The region is also automatically unmapped when the process is terminated. */
  pmem_unmap(pmemaddr, mapped_len);
}
```
Original libart art_insert routine

```c
void*
art_insert(art_tree *t, const unsigned char *key, int key_len, void *value)
{
    int old_val = 0;
    void *old = recursive_insert(t->root, &t->root, key, key_len, value, 0, &old_val);
    if (!old_val) t->size++;
    return old;
}
```
libart art_insert routine ... ported to libpmemobj

```
TOID(var_string)
art_insert(PMEMobjpool *pop, const unsigned char *key, int key_len, void *value, int val_len)
{
    int old_val = 0;
    TOID(var_string) old;
    TOID(struct art_tree_root) root;

    TX_BEGIN(pop) {
        root = POBJ_ROOT(pop, struct art_tree_root);
        TX_ADD(root);

        old = recursive_insert(pop, D_RO(root)->root, &(D_RW(root)->root), (const unsigned char *)key, key_len, value, val_len, 0, &old_val);
        if (!old_val) D_RW(root)->size++;
    } TX_ONABORT {
        abort();
    } TX_END

    return old;
}
```
- Technology
- Setup and Configuration
- Software Programming Stack
- Measurements
addr-test
addr-test – 1x DCPMM vs. DRAM

**Measurement on pre-production HW**
pmdk pmembench – 1x DCPMM vs. DRAM

Measurement on pre-production HW
Test system PRIMERGY

- DRAM 32 GB
- DCPMM 256 GB
- empty

namespace

pmm0

pmem1

pmem2

rx24-1 RX2540-M4
CPU Xeon 82xx, 24C, 2.6GHz
BIOS R1.3.1 for D3384-B1x
mlc – idle_latency

**Measurement on pre-production HW**

```
mlc -c30 --idle_latency -J/mnt/pmem[012] -p36,6
mlc –c2 --idle_latency -J/mnt/pmem[012] -p36,6
```
mlc – loaded_latency 100% read - bandwidth

Measurement on pre-production HW
mlc – loaded_latency 100% random write

Measurement on pre-production HW

mlc --loaded_latency -d0 -g<perthreadfile> <perthreadfile>: 1-[2/4/.../40] W6 rand pmem /mnt/pmem[0/1/2]
## Improvements of pmdk and AEP generations

- **Libart3**: adaptive radix tree as an example with libpmemobj
- **Single threaded application**
- **DIMM access on local / far socket**

<table>
<thead>
<tr>
<th></th>
<th>pmdk 1.2</th>
<th></th>
<th>pmdk 1.3</th>
<th></th>
<th>pmdk 1.4</th>
<th></th>
<th>pmdk 1.5</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>local</td>
<td>far</td>
<td>local</td>
<td>far</td>
<td>local</td>
<td>far</td>
<td>local</td>
<td>far</td>
</tr>
<tr>
<td><strong>simulated pmem</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>insert</td>
<td>15965</td>
<td>26448</td>
<td>10359</td>
<td>17415</td>
<td>9774</td>
<td>16318</td>
<td>9946</td>
<td>15255</td>
</tr>
<tr>
<td>re-insert</td>
<td>1046</td>
<td>1141</td>
<td>1097</td>
<td>1232</td>
<td>1101</td>
<td>1235</td>
<td>1096</td>
<td>1234</td>
</tr>
<tr>
<td>lookup</td>
<td>798/747</td>
<td>773/737</td>
<td>793/752</td>
<td>754/750</td>
<td>785/747</td>
<td>777/746</td>
<td>794/760</td>
<td>784/754</td>
</tr>
<tr>
<td><strong>AEP ES2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>insert</td>
<td>34273</td>
<td>41197</td>
<td>23356</td>
<td>28031</td>
<td>20888</td>
<td>26899</td>
<td>18482</td>
<td>27542</td>
</tr>
<tr>
<td>re-insert</td>
<td>1514</td>
<td>1671</td>
<td>1622</td>
<td>1771</td>
<td>1603</td>
<td>1776</td>
<td>1621</td>
<td>1802</td>
</tr>
<tr>
<td>lookup</td>
<td>811/848</td>
<td>807/853</td>
<td>831/878</td>
<td>823/891</td>
<td>828/887</td>
<td>819/884</td>
<td>829/871</td>
<td>785/912</td>
</tr>
<tr>
<td><strong>AEP QS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>insert</td>
<td>35269</td>
<td>40531</td>
<td>22881</td>
<td>27064</td>
<td>19760</td>
<td>25493</td>
<td>16739</td>
<td>25840</td>
</tr>
<tr>
<td>re-insert</td>
<td>1479</td>
<td>1651</td>
<td>1508</td>
<td>1686</td>
<td>1510</td>
<td>1692</td>
<td>1496</td>
<td>1699</td>
</tr>
<tr>
<td>lookup</td>
<td>810/831</td>
<td>841/859</td>
<td>810/879</td>
<td>866/904</td>
<td>831/885</td>
<td>822/897</td>
<td>812/852</td>
<td>832/895</td>
</tr>
</tbody>
</table>

*(in CPU cycles)*

Measurement on pre-production HW
- Technology
- Setup and Configuration
- Software Programming Stack
- Measurements
- **Direction, Vision, Outlook**
Use Case: High Availability, Replication

Usage: replicate data that is stored in local PM across a fabric and store it in remote PM

What it looks like

PMoF
The Holy Grail of PMoF

Loads and stores on a client CPU affect Persistent Memory across the fabric!

We are a loooong way from here!

The knights that say “c”!
Open issues and open questions

- DCPMMs support encryption, but today manual entry in BIOS at startup
  - Future (?) : Key-Mgmt, integrate DIMM unlock in Kernel boot

- With direct load/store all block based tools can no longer be used (Virus detection, delta backup based on modified blocks, ...)

- DCPMM re-configuration (extension, replacement)

- Hierarchical memory management ... does not exist today

- How can write (CPU store) errors be detected and mapped to the related process without I/O logic?
  - In Linux EDAC (Error Detection And Correction) would map a MCE to a DIMM

- How to handle malware in persistent memory?

- All pmdk libraries today are heavily overweight and hardly show the advantage of store/load to advantage

- Security, performance and manageability of PMoF
DCPMM can be shipped in April 2019 for all Cascade Lake systems
- attractive price compared to DRAM
- ~ 4x higher capacity as DRAM
- ~ 10x higher latency as DRAM
- Accessible as DRAM: load(address), store(address)

Available on all Fujitsu x86 Server PRIMERGY and PRIMEQUEST with SLES12-SP4 and SLES15-SP1

Version 1 of a new Technology, Speed and capacity will improve

Usability
- All mandatory interfaces (setup, monitor, error detection, programming model, C-Libraries, encryption, PMoF) are available today
Summary (2)

- Many open issues and question especially in the area of error detection / recovery, setup, configuration, security, manageability
- Inefficient and heavily overweighed C-Libraries
- How to replace the behaviour and tools that today depend on the block device ?
- Security, performance and manageability of PMoF
- Hierarchical memory management (near, far, remote, persistent, volatile, shared … memory)
shaping tomorrow with you