

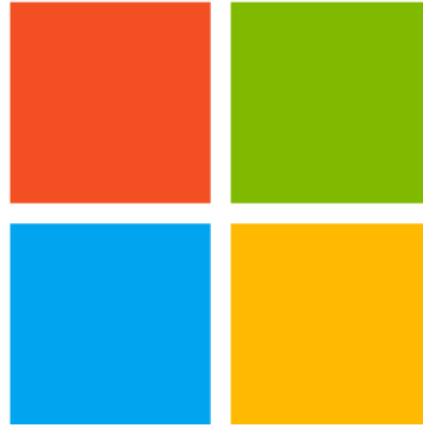


JANUARY 24, 2019 | SANTA CLARA, CA

## PM Support in Linux and Windows

Dr. Stephen Bates, CTO, Eideticom

Neal Christiansen, Principal Development Lead, Microsoft



# Windows Support for Persistent Memory

# Availability of Windows PM Support

## ➤ Client Workstation:

- ◆ August, 2016: Windows 10 Anniversary Update
- ◆ April, 2017: Windows 10 Creators Update
- ◆ October, 2017: Windows 10 Fall Creators Update
- ◆ April, 2018: Windows 10 April 2018 Update
- ◆ October, 2018: Windows 10 October 2018 Update

## ➤ Server:

- ◆ September, 2016: Windows Server 2016
- ◆ November, 2019: Windows Server 2019

## ➤ Supported Hardware:

- ◆ JEDEC NVDIMM-N
- ◆ HPE Scalable Persistent Memory

- Supported since Windows Server 2016
- Advantages:
  - ◆ Improved IO performance by eliminating OS overhead
- Disadvantages:
  - ◆ Functionality loss due to elimination of OS hook points, examples:
    - › Software Encryption
    - › Software Compression
- Only NTFS supports DAX

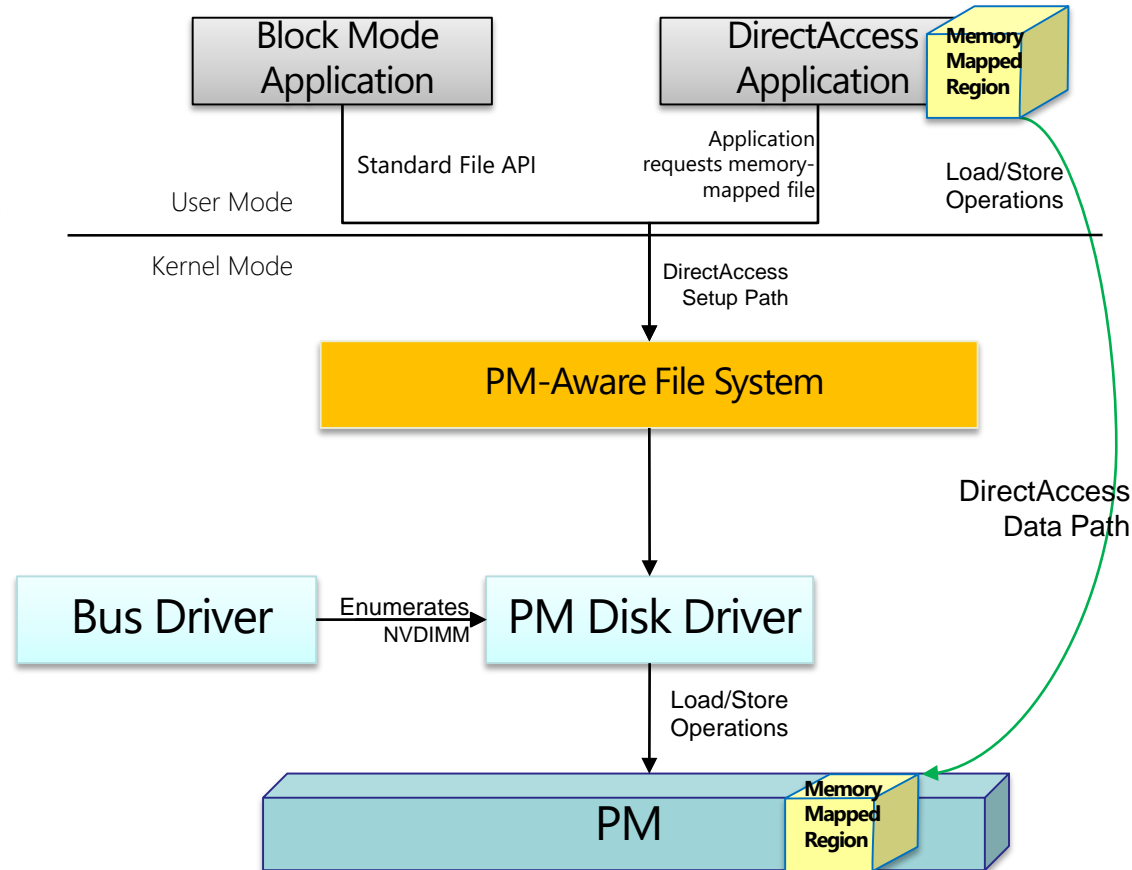
# How DAX Works

## ➤ The Idea

- App has direct access to PM via existing memory-mapping semantics
- Updates directly modify PM
  - ◆ Storage Stack not involved

## ➤ Characteristics

- True device performance (no software overhead)
- Byte-Addressable



## ➤ Non-Cached IO

- ◆ Is converted to cached IO by the file system

## ➤ Cached IO

- ◆ Cache Manager maps directly to persistent memory
- ◆ Copies directly between user's buffer and persistent memory

## ➤ Memory Mapped IO

- ◆ Memory mapped sections point directly to PM hardware
- ◆ Provides applications with zero-copy access to PM hardware

## Is fully backwards compatible

- Maintains existing storage semantics
  - ◆ Sector atomicity supported by the PM disk driver
- Fully compatible with existing applications and filter drivers
- Supported by all Windows file systems

- Available in Windows Server 2019
- Windows & Linux guests in generation 2 VMs see virtual PMEM (vPMEM) devices
  - ◆ Memory mapped files in the guest have direct access to PM hardware on the host
  - ◆ Full Win32 and PMDK support
- New VHD file type: **.VHDPMEM**



# What are Large and Huge Pages

- ▶ Modern CPUs manage memory using 4K pages
- ▶ An applications memory usage is managed via page tables controlled by the operating systems memory manager
- ▶ CPU's contain a mapping table cache called the TLB (translation lookaside buffer) that caches page table mappings
- ▶ For applications with a large memory footprint -- the CPU can spend a lot of time reading page table entries into the TLB
- ▶ A Large Page allows a contiguous 2mb region to be described with a single TLB entry
  - ◆ Applications typically see a significant performance improvement
- ▶ A Huge Page allows a contiguous 1gb region to be described with a single TLB entry

# Windows PM support for Large & Huge Pages

- Available in Windows Server 2019
- DAX partitions are aligned to 2mb boundaries
- NTFS now supports cluster sizes up to 2mb (in powers of 2)
  - ◆ In Server 2016 the limit was 64K
- A memory mapped file on a DAX volume with a 2mb cluster size is guaranteed to be mapped using at least Large Pages
- Large and Huge page alignment is supported on cluster sizes <2mb

# Windows PM Management Support

- Windows Server 2019 introduces Powershell support for managing physical and logical persistent memory devices
  - ◆ Ability to enumerate, create and delete logical persistent memory devices
  - ◆ Ability to enumerate and initialize physical persistent memory devices
  - ◆ Example cmdlets:
    - › Get-PmemDisk
    - › New-PmemDisk
    - › Remove-PmemDisk
    - › Get-PmemPhysicalDevice
    - › Initialize-PmemPhysicalDevice
    - › Get-PmemUnusedRegion

# Windows PMDK Support

- Available since Windows Server 2016
- Defines a set of application API's for efficient use of PM hardware
  - ◆ Abstracts out OS specific dependencies
  - ◆ Underlying implementation uses memory mapped files
  - ◆ Makes its own atomicity guarantees
  - ◆ Works in both PM and non-PM environments
  - ◆ Simpler application development model
- Open source library available for Windows and Linux via GitHub
  - ◆ <https://github.com/pmem/pmdk/>

# Examples uses of PM on Windows

## ▶ **SQL Server 2016**

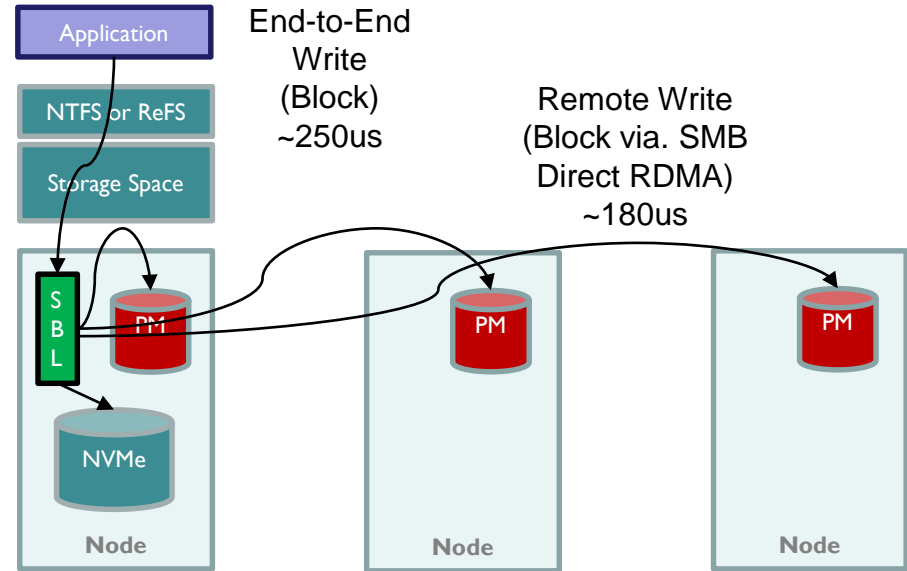
- ◆ **Tail-of-the-Log:** Transactions committed to DAX volume
- ◆ 2X performance gain by:
  - › **Halving** the transaction latency
  - › De-serializing the SQL logging threads

# Write Caching for HCI with Storage Spaces Direct & Windows Server 2019

- Ultra-low-latency write caching (SBL) for all-NVMe deployments
- Industry record performance demonstrated at Ignite '18

Benchmark	Performance
4K 100% Rand Read	13.8 Million IOPs
4K 90/10% Rand R/W	9.45 million IOPs
2MB Sequential Read	549 GB/s Throughput

12-nodes with Triple-Mirrored Scoped Spaces with ReFS  
12 x Intel® S2600WFT, **384 GiB** memory, **2 x 28-core** “CascadeLake”, **1.5 TB** Intel® Optane™ DC persistent memory as cache, **32 TB** NVMe (4 x 8TB Intel® DC P4510) as capacity, **2 x Mellanox ConnectX-4 25 Gbps**

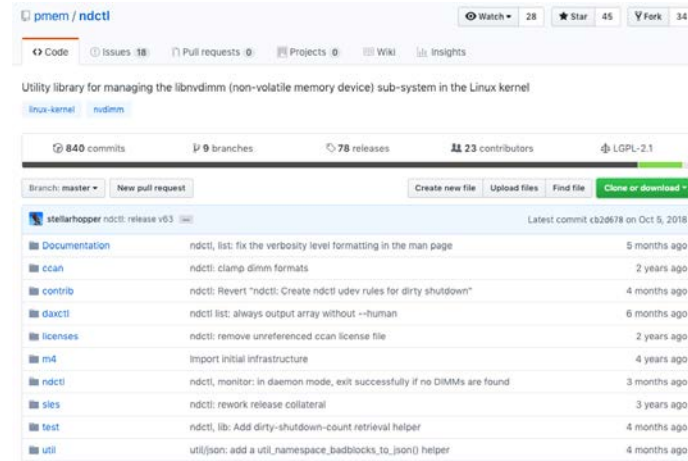




## Linux Support for Persistent Memory

# Managing PM in Linux: ndctl

- How do you determine if you have PM in your system?
- How do you manage the PM in your system?
- How do you determine the health of PM in your system?
- Ties into physical layer specifications like NFIT and HMAT [1].



pmem / ndctl

Utility library for managing the libnvdimm (non-volatile memory device) sub-system in the Linux kernel

linux-kernel | nvdimm

840 commits | 9 branches | 78 releases | 23 contributors | LGPL-2.1

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

stellarhopper ndctl: release v63 | Latest commit c820678 on Oct 5, 2018

File	Commit	Time
Documentation	ndctl, list: fix the verbosity level formatting in the man page	5 months ago
ccan	ndctl: clamp dimm formats	2 years ago
contrib	ndctl: Revert "ndctl: Create ndctl udev rules for dirty shutdown"	4 months ago
dxctl	ndctl list: always output array without --human	6 months ago
licenses	ndctl: remove unreferenced ccan license file	2 years ago
m4	Import initial infrastructure	4 years ago
ndctl	ndctl, monitor: in daemon mode, exit successfully if no DIMMs are found	3 months ago
sls	ndctl: rework release collateral	3 years ago
test	ndctl, lib: Add dirty-shutdown-count retrieval helper	4 months ago
util	util(json): add a util_namespace_badblocks_to_json() helper	4 months ago

```
generic@pmsummit1:~$ ndctl list
{
  "dev": "namespace0.0",
  "mode": "raw",
  "size": 16646144,
  "sector_size": 512,
  "blockdev": "pmem0",
  "numa_node": 0
}
generic@pmsummit1:~$
```

[1] [http://www.uefi.org/sites/default/files/resources/ACPI\\_6\\_2.pdf](http://www.uefi.org/sites/default/files/resources/ACPI_6_2.pdf)



# Using PM in Linux: A block device

- PM can be consumed by Linux applications using a block interface.
- Useful when a file-system is not required.
- Not super-efficient as block device changes on sector, not byte granularity.
- All your favourite block-device tools can be used.

```
~ — batesste@tyrone: ~/vpns — ssh vm
[generic@pmsummit-1:~$ sudo fdisk -l /dev/pmem0
Disk /dev/pmem0: 15.9 MiB, 16646144 bytes, 32512 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
[generic@pmsummit-1:~$
[generic@pmsummit-1:~$ lsblk
NAME MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda   252:0    0   12G  0 disk
`-vda1 252:1    0   12G  0 part /
pmem0 259:0    0 15.9M  0 disk
[generic@pmsummit-1:~$
generic@pmsummit-1:~$ █
```

[1] [http://www.uefi.org/sites/default/files/resources/ACPI\\_6\\_2.pdf](http://www.uefi.org/sites/default/files/resources/ACPI_6_2.pdf)

# Using PM in Linux: A filesystem

- Like any standard block device we can put a filesystem over it.
- In this case we can put ANY filesystem we like on top.
- But if we use a DAX aware filesystem we get added benefits (PM optimizations).
- Now we can have files and directories and all that good stuff!
- In Linux both EXT4 and XFS have DAX support.

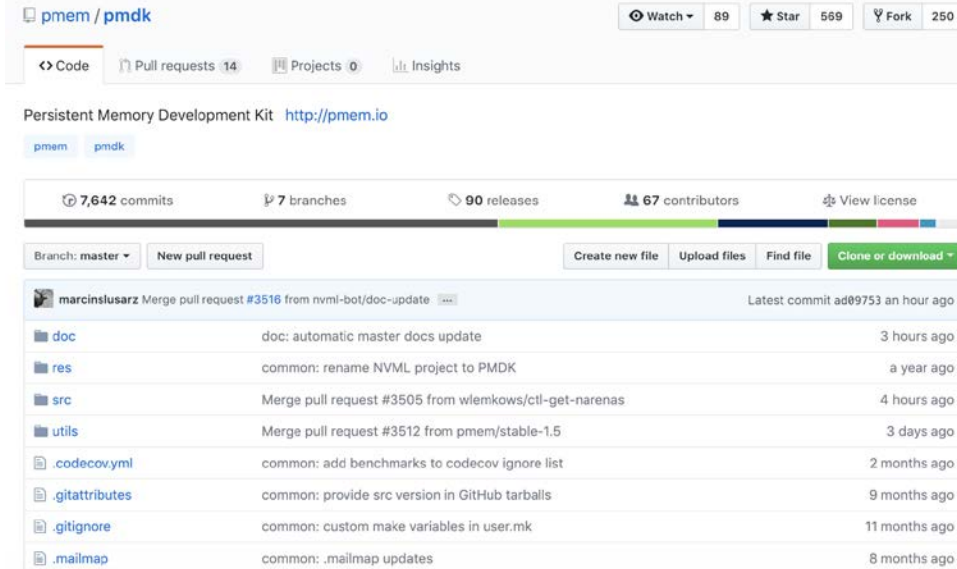
```

[guest@pmsummit-1:~]$ mkfs.xfs -f /dev/pmem0
mkfs.xfs: cannot open /dev/pmem0: Permission denied
[guest@pmsummit-1:~]$ sudo mkfs.xfs -f /dev/pmem0
[sudo] password for guest:
meta-data=/dev/pmem0          isize=512    agcount=4, agsize=56320 blks
                           =                sectsz=4096  attr=2, projid32bit=1
                           =                crc=1      finobt=1, sparse=0, rmapbt=0, reflink=0
data                =                bsize=4096  blocks=225280, imaxpct=25
                           =                sunit=0    swidth=0 blks
naming              =version 2        bsize=4096  ascii-ci=0 ftype=1
log                 =internal log     bsize=4096  blocks=1605, version=2
                           =                sectsz=4096 sunit=1 blks, lazy-count=1
realtime            =none             extsz=4096  blocks=0, rtextents=0
[guest@pmsummit-1:~]$ sudo mount -o dax /dev/pmem0 /mnt/
[guest@pmsummit-1:~]$ dmesg | tail
[ 16.212472] pmem0: detected capacity change from 0 to 130023424
[ 184.699585] pmem0: detected capacity change from 0 to 134217728
[ 203.539578] pmem0: detected capacity change from 0 to 939524096
[ 210.083474] pmem0: detected capacity change from 0 to 922746880
[ 232.851656] random: crng init done
[ 232.851694] random: 7 urandom warning(s) missed due to ratelimiting
[10604.527337] SGI XFS with ACLs, security attributes, realtime, no debug enabled
[10604.588508] XFS (pmem0): DAX enabled. Warning: EXPERIMENTAL, use at your own risk
[10604.588563] XFS (pmem0): Mounting V5 Filesystem
[10604.627537] XFS (pmem0): Ending clean mount
[guest@pmsummit-1:~]$ █

```

# Using PM in Linux: mmap()

- Now we have files that live in a PM-aware filesystem on a PM-aware block device on physical PM.
- mmap() has been around for a while ;-). Maps the file into the Virtual Address space of the running process.
- Now the world becomes our oyster (see PMDK for more!)



pmem / pmdk

Watch 89 Star 569 Fork 250

Code Pull requests 14 Projects 0 Insights

Persistent Memory Development Kit <http://pmem.io>

pmem pmdk

7,642 commits 7 branches 90 releases 67 contributors View license

Branch: master New pull request Create new file Upload files Find file Clone or download

Commit	Message	Time
marcinslusarz Merge pull request #3516 from nvml-bot/doc-update		Latest commit ad89753 an hour ago
doc	doc: automatic master docs update	3 hours ago
res	common: rename NVML project to PMDK	a year ago
src	Merge pull request #3505 from wlemkows/ctl-get-narens	4 hours ago
utils	Merge pull request #3512 from pmem/stable-1.5	3 days ago
.codecov.yml	common: add benchmarks to codecov ignore list	2 months ago
.gitattributes	common: provide src version in GitHub tarballs	9 months ago
.gitignore	common: custom make variables in user.mk	11 months ago
.mailmap	common: .mailmap updates	8 months ago

git clone <https://github.com/pmem/pmdk.git>

# Thank You