SNIA PERSISTENT MEMORY PM SUMMIT

JANUARY 18, 2017 | SAN JOSE, CA

Persistent Memory in Windows

Tom Talpey, Microsoft

Agenda

- Windows Persistent Memory support
- Windows PMEM storage access details
- SQL Server 2016 results

Windows PMEM Support

- Persistent Memory is supported in Windows 10 and Windows Server 2016
  - PM support is foundational in Windows and is SKU-independent
- Support for JEDEC-defined NVDIMM-N devices available from
  - Windows Server 2016
  - Windows 10 (Anniversary Update – Fall 2016)

Windows PMEM Access Methods

- Direct Access (DAX) Filesystem
  - Mapped files with load/store/flush paradigm
  - Cached and noncached with read/write paradigm
- Block-mode ("persistent ramdisk")
  - Raw disk paradigm
- Application interfaces
  - Mapped and traditional file
  - NVM Programming Library
  - "PMEM-aware" open coded

Windows Goals for Persistent Memory

- Support zero-copy access to persistent memory
- Most existing user-mode applications will run without modification
- Provide an option to support 100% backward compatibility
  - Does introduce new types of failure modes
- Provide sector granular failure modes for application compatibility

PM Storage Drivers

- New driver model optimized for PM hardware
  - SCM Bus Driver
    - Enumerates the physical and logical PM devices on the system
    - Not part of the IO Path
  - SCM Disk Drivers
    - Driver for logical PM devices
    - Storage abstraction layer to rest of the OS
    - Hardware-specific
    - Windows uses a native 4K sector size
- Introduces new interfaces
  - Expose byte addressable storage functionality
  - Supports management of PM hardware

Introducing a New Class of Volume

- Direct Access Storage (DAX) Volume
  - Memory mapped files will provide applications with direct access to PM
    - Maximizes performance
  - DAX mode is chosen at volume format time
    - Why: compatibility issues with various components, examples:
      - File system filters
      - Bitlocker (volume level software encryption)
      - Volsnap (volume snapshot provider)
  - Some existing functionality is lost
  - DAX Volumes are currently supported by NTFS
    - Part of Windows 10 Anniversary Update / Server 2016 releases

Memory Mapped IO in DAX mode

- On DAX formatted volumes memory mapped sections map directly to PM hardware
  - No change to existing memory mapping APIs
- When an application creates a memory mapped section:
  - The memory manager (MM) asks the file system if the section should be created in DAX mode (Direct Access Storage)
  - The file system returns YES when:
    - The volume resides on PM hardware
    - The volume has been formatted for DAX mode

Memory Mapped IO in DAX mode

- This is true zero-copy access to storage
  - An application has direct access to persistent memory
- Important → No paging reads or paging writes will be generated

Cached IO in DAX mode

- The cache manager creates a cache map that maps directly to PM hardware
- The cache manager copies directly between user's buffer and persistent memory
  - Cached IO has one-copy access to persistent storage
- Cached IO is coherent with memory mapped IO
- As in memory mapped IO, no paging reads or paging writes are generated
  - No Cache Manager Lazy Writer thread

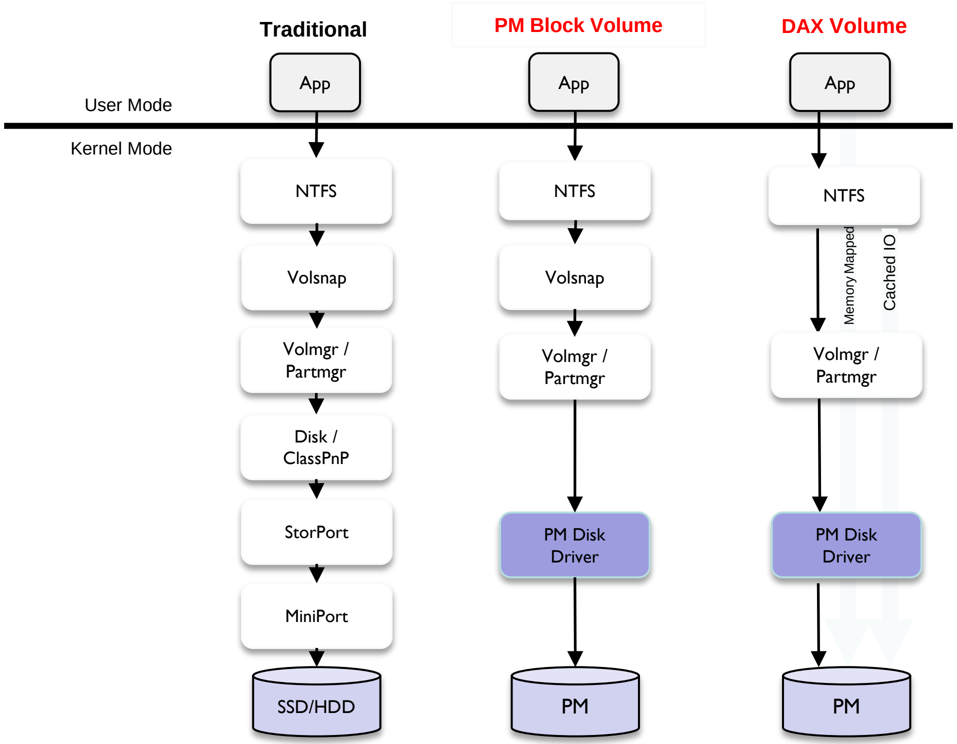
Non-cached IO in DAX Mode

- Is simply converted to cached IO by the file system
  - Cache manager copies directly between user's buffer and persistent memory
- Is coherent with cached and memory mapped IO

Backward App Compatibility on PM Hardware

- **Block Mode Volumes**
  - Maintains existing storage semantics
    - All IO operations traverse the storage stack to the PM disk driver
    - Sector atomicity guaranteed by the PM disk driver
    - Has shortened path length through the storage stack to reduce latency
      - No storport or miniport drivers
      - No SCSI translations
  - Fully compatible with existing applications
  - Supported by all Windows file systems
  - Works with existing file system filters
  - Block mode vs. DAX mode is chosen at format time

IO Stack Comparisons



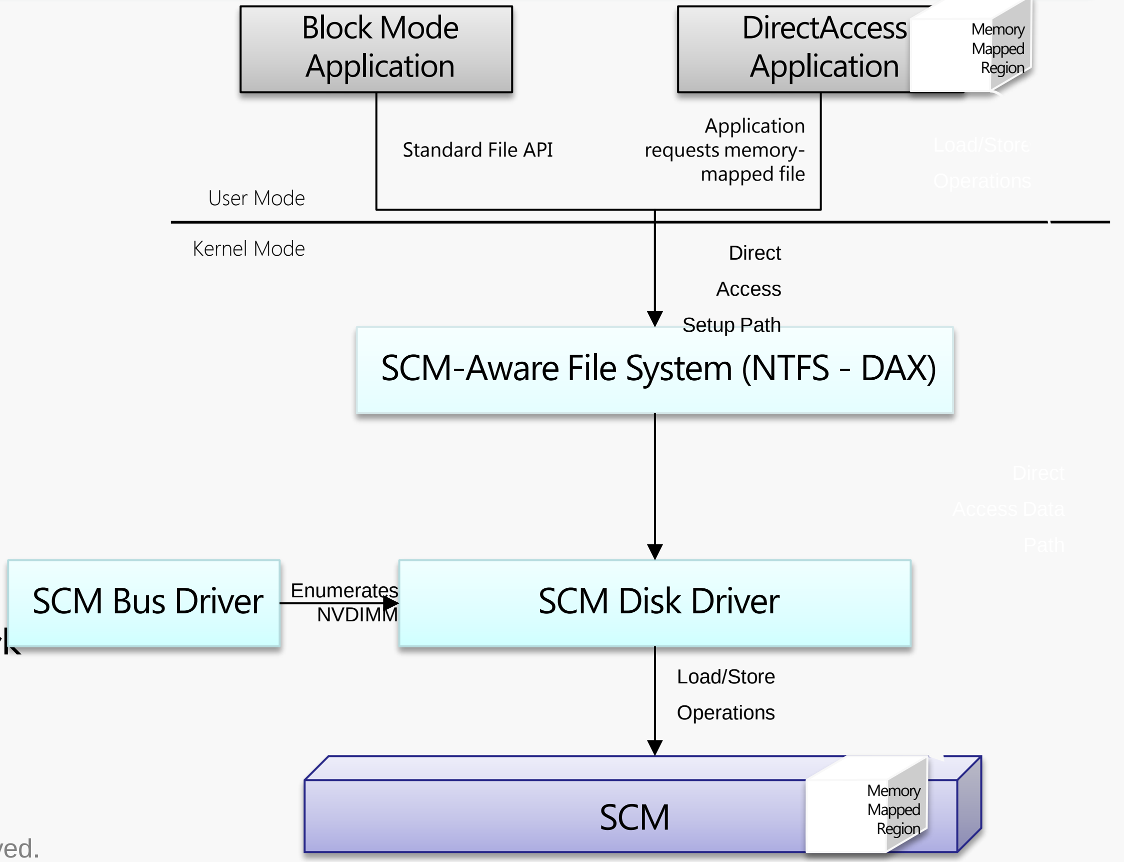
Direct Access Architecture

Overview

- App has direct access to SCM via existing memory-mapping semantics
- Updates directly modify SCM, Storage Stack not involved
- DAX volumes identified through new flag

Characteristics

- True device performance (no software overhead)
- Byte-Addressable
- Filter Drivers relying on I/O may not work or attach – no I/O, new volume flag
- AV Filters can still operate (Windows Defender already updated)



Performance Comparison

4K random writes

1 Thread, single core

	IOPS	Avg Latency (ns)	MB / Sec
NVMe SSD	14,553	66,632	56.85
Block Mode NVDIMM	148,567	6,418	580.34
DAX Mode NVDIMM	1,112,007	828	4,343.78

Using DAX in Windows

DAX Volume Creation

→ `Format n: /dax /q`

→ `Format-Volume –DriveLetter n –IsDAX $true`

DAX Volume Identification

Is it a DAX volume?

→ call `GetVolumeInformation(“C:\”, ...)`

→ check `lpFileSystemFlags` for `FILE_DAX_VOLUME (0x20000000)`

Is the file on a DAX volume?

→ call `GetVolumeInformationByHandleW(hFile, ...)`

→ check `lpFileSystemFlags` for `FILE_DAX_VOLUME (0x20000000)`

Using DAX in Windows

Memory Mapping

```
1. HANDLE hMapping = CreateFileMapping(hFile, NULL, PAGE_READWRITE, 0, 0, NULL);
2. LPVOID baseAddress = MapViewOfFile(hMapping, FILE_MAP_WRITE, 0, 0, size);
3. memcpy(baseAddress + writeOffset, dataBuffer, ioSize);
4. FlushViewOfFile(baseAddress, 0);
```

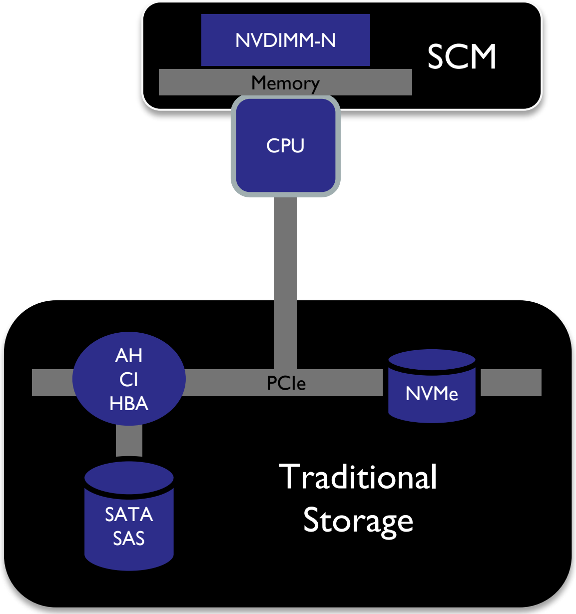
OR … use non-temporal instructions for NVDIMM-N devices for better performance

```
1. HANDLE hMapping = CreateFileMapping(hFile, NULL, PAGE_READWRITE, 0, 0, NULL);
2. LPVOID baseAddress = MapViewOfFile(hMapping, FILE_MAP_WRITE, 0, 0, size);
3. RtlCopyMemoryNonTemporal(baseAddress + writeOffset, dataBuffer, ioSize );
```

Future … use open source NVM Programming Library

Fast Transactions in SQL Server

- **Problem**
  - DB Transactions gated by log write speed
  - The faster the log, the more DB updates possible
- **Opportunity**
  - Accelerate Log Commits
  - Accelerate DB
  - Provide better customer SLAs
- **Approach**
  - Log on SCM – Persistent Medium on Memory Bus
  - NVDIMM-N supported in WS 2016
  - NVDIMM-N based on DDR4 RAM + Flash for backup
  - Exposes Block Interface (like a Disk) or Direct Access Interface (Load/Store)
  - Direct Access (DAX): Enlightened apps can directly access their data on the SCM device via Load/Store instructions



Faster Transaction Processing with PMEM

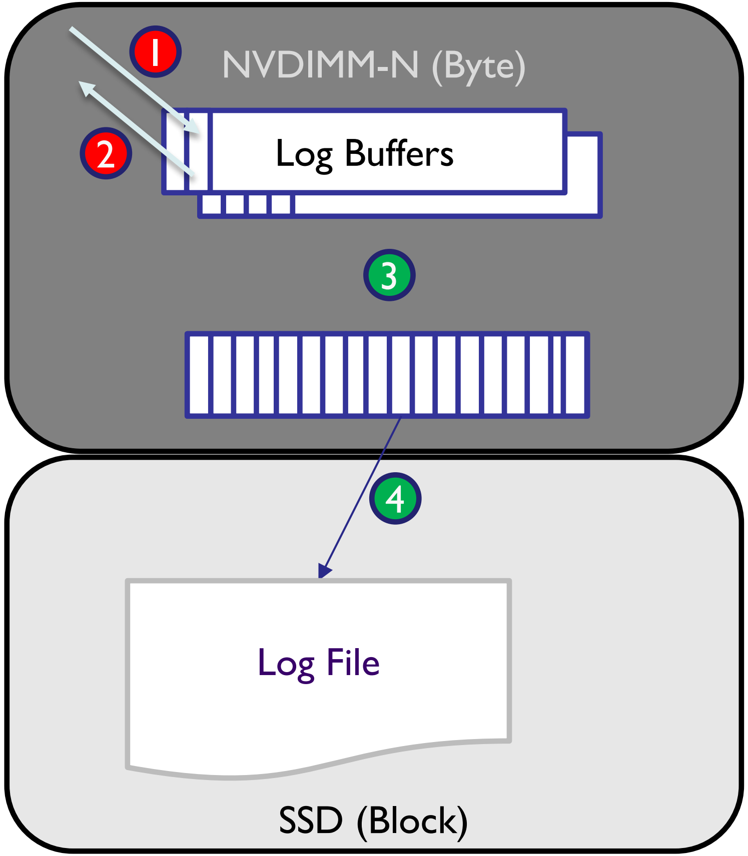
In the Past:

- Copy log records into buffer, building up block
- Close log block once commit arrives
- Schedule I/O to persist block on SSD
- Complete transaction when I/O completes

With "Tail of Log":

1. Copy log records into buffer, building up block **persistently in PMEM**
2. Complete transaction when commit arrives
3. Close log block when full
4. Schedule I/O to re-persist full block to SSD

Red indicates the critical path for a transaction



Accelerating SQL 16 with PMEM

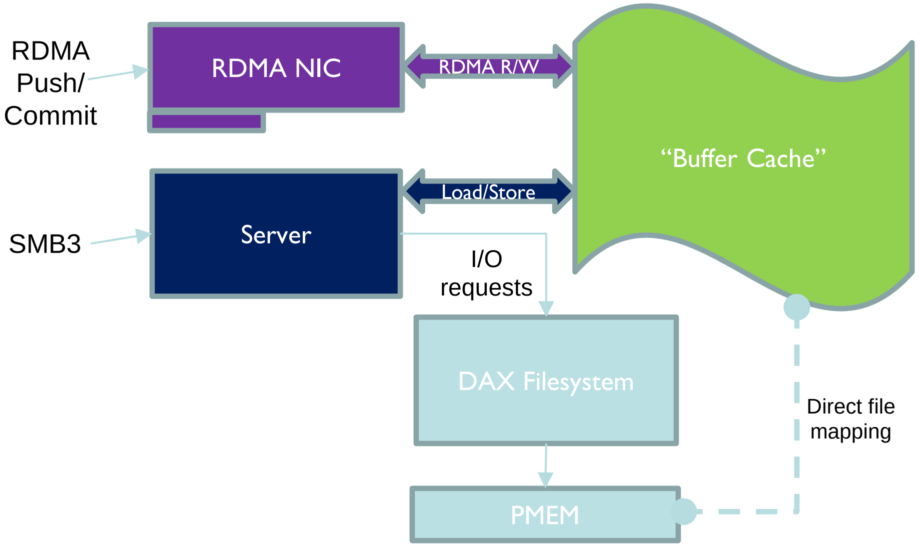
- SQL 16 can use a byte-addressable log (commit @ DRAM speed)
- Enabled through DAX volumes on NVDIMM-N in WS 2016
- Accelerate In-Memory DB updates by up to 2x

Configuration	HK on NVMe (block)	HK on NVDIMM-N (DAX)
Row Updates / Second	63,246	124,917
Avg. Time / Txn (ms)	0.379	0.192

Configuration: Row Size: 32B, Table Size: 5GB, Threads:24, Batch Size: 1

Going Remote – SMB3

- SMB3 RDMA and "Push Mode" discussed at previous SNIA Storage Developers Conference
- Enables **zero-copy** remote read/write
  - Ultra-low latency and overhead
- When RDMA Commit extensions become available, this can become reality



More detail

- Windows Persistent Memory @ IDF 2016
  - http://myeventagenda.com/sessions/0B9F4191-1C29-408A-8B61-65D7520025A8/14/5#sessionID=1446
- SQL Server 2016 (also with demos)
  - https://myignite.microsoft.com/videos/2767
  - https://channel9.msdn.com/Shows/Data-Exposed/SQL-Server-2016-and-Windows-Server-2016-SCM--FAST
  - https://blogs.msdn.microsoft.com/sqlserverstorageengine/2016/12/02/transaction-commit-latency-acceleration-using-storage-class-memory-in-windows-server-2016sql-server-2016-sp1/
- SMB3 @ SDC2016
  - http://www.snia.org/sites/default/files/SDC/2016/presentations/persistent_memory/Tom_Talpey_Low_Latency_Remote_Storage_A_Full-stack_View.pdf

Questions?

- Thank you!