Windows Persistent Memory Support

Neal Christiansen
Microsoft
Principal Development Lead

What is “Persistent Memory”?

- Non-volatile storage with RAM-like performance
  - Low latency/high bandwidth.
- Resides on the memory bus
- Terms used to describe the hardware:
  - Storage Class Memory (SCM)
  - Byte Addressable Storage (BAS)
  - Non-Volatile Memory (NVM)
  - Persistent Memory (**PM**) ⟸ Industry converging on this term

File Systems and Persistent Memory

- **PM is a disruptive technology**

- Customers want the fastest performance
  - System software is in the way!
- Customers want application compatibility
- Conflicting goals

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

Windows PM Support

- PM support is foundational and Windows SKU independent
- Support for JEDEC-defined NVDIMM-N devices available in Windows 10 Anniversary Update and Windows Server 2016
  - Available for preview in Windows 10 Insider Builds and Windows Server 2016 TP5

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 both:
    - The volume resides on PM hardware
    - The volume has been formatted for DAX mode

Memory Mapped IO in DAX mode

- When a view to a memory mapped section is created on a DAX mode volume:
  - MM asks the file system for the physical memory ranges for a given range of the file
  - The file system translates the range into one or more volume relative extents (sector offset and length)
  - The file system then asks the PM disk driver to translate these extents into physical memory ranges
  - MM then updates its paging tables for the section to map directly to the persistent storage

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 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

File System Metadata in DAX Mode

- NTFS file system metadata will not use DAX mode sections
  - Meaning paging reads/writes will be generated for all file system metadata operations
  - Needed to maintain existing ordered write guarantees for write-ahead logging
- One or more metadata files may use DAX mode in the future

Impacts to File System Functionality in DAX Mode

- Direct access to persistent memory by applications eliminates the traditional hook points that file systems use to implement various features
- File system functionality that can not be supported on DAX enabled volumes:
  - No NTFS software encryption support (EFS)
  - No NTFS software compression support
  - No NTFS TxF support (transactional NTFS)
  - No NTFS USN range tracking of memory mapped files
  - No NTFS resident file support

Impacts to File System Functionality in DAX Mode

- File system no longer knows when a writeable memory mapped section is modified:
  - The following file system features are now updated at the time a writeable mapped section is created:
    - File's modification and access times
    - Marking the file as modified in the USN (change) Journal
    - Signaling directory change notification
- Functionality on DAX volumes not supported in the upcoming release:
  - Sparse Files
  - Defragging files

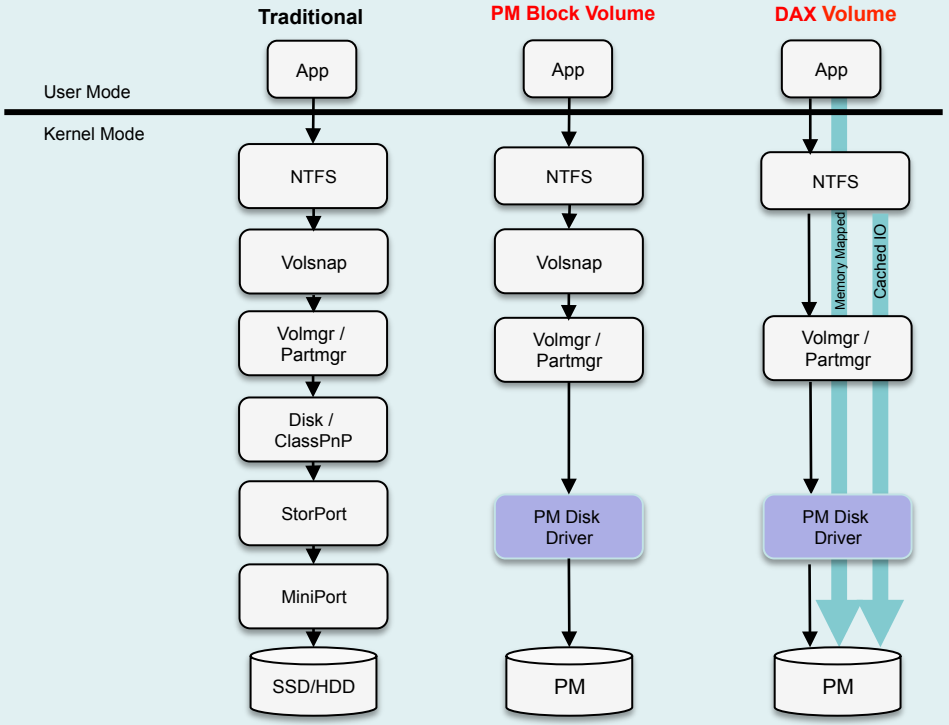
New Volume Device Class

- New byte addressable partition type
  - Set at format time
- Why: Improves compatibility and performance by removing non-DAX aware/capable volume drivers:
  - VOLSNAP – no support for volume snapshots
  - BITLOCKER – no support for software encryption
  - 3rd Party volume stack filters

Backward 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



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-N	148,567	6,418	580.34
DAX Mode NVDIMM-N	1,112,007	828	4,343.78

Accelerating SQL 16 with PM

	Row Updates / Second	Avg. Time / Txn (ms)
NVMe SSD	63,246	0.379
Dax Mode NVDIMM-N	124,917	0.192

What is a File System Filter

- A driver that layers above the file system
- Augments file system functionality
  - May interact with all operations as they come into and out of the file system
- Example classes of filters:
  - Anti-virus
  - Replication
  - Hierarchical Storage Management (HSM)
  - Security Enhancer
  - Encryption
  - Compression
  - Quota
  - Activity monitor

File System Filters in DAX Mode

To minimize compatibility issues:

- No existing filters will receive notification when a DAX volume is mounted
- At filter registration time filters will indicate via a new registration flag that they understand DAX mode semantics

Compatibility Issues with Filters in DAX Mode

Reason: **No paging IO**

- Data transformation filters (ex: encryption and compression)
  - No opportunity for these filters to transform data for memory mapped files
- Replication filters
  - Difficult to detect when a file has changed
  - Difficult to efficiently track what ranges of a file have been modified
- Anti-virus filters
  - Minimally impacted because scanning is performed at file open and close time
  - Need to update how to detect changed files
- Quota filters
  - Minimally impacted as file size changes behave as always

Sector Atomicity

- BTT – Block Translation Table
  - Algorithm created by Intel
  - Provides efficient sector level atomicity of writes
    - Eliminates sub-sector torn writes
    - On power loss either see contents of old sector or new sector
    - Provides compatibility for existing applications that have built-in assumptions around storage failure patterns
    - Minimal performance impact
  - Implemented by remapping the physical PM address of a given LBA (volume relative logical block address)
    - Not compatible with DAX mode memory mapped views since physical PM addresses are given to the memory manager

BTT Usage

- Uses small portion of PM space for mapping tables and control structures
  - BTT structures are not visible outside the PM driver
- File system controls if a given write should use BTT or not
  - New per-IO flag indicates if a given LBA may be remapped or not
    - If NOT SET the given LBA may be remapped (use BTT)
    - If SET the given LBA must not be remapped (do not use BTT)

BTT Usage

- NTFS Block mode volumes always indicate that an LBA may be remapped (use BTT)
- NTFS DAX mode volumes:
  - All metadata writes are remappable (use BTT)
    - NTFS metadata can not detect sub-sector torn writes
  - All other writes are not remappable (do not use BTT)

Application use of PM

- Intel NVML Library
  - Open source library implemented by Intel
    - Available for Linux via GitHub
    - https://github.com/pmem/nvml/
  - Defines a set of application API's for efficient use of PM hardware
    - Abstracts out OS specific dependencies
    - Underlying implementation uses memory mapped files
      - All access via API calls
    - Has its own per-file BTT implementation for atomicity guarantees
    - Works in both PM and non-PM hardware environments
  - Microsoft is working with Intel, HPE and HP Labs on a Windows port
    - Most functionality is up and running
    - We welcome anyone else that would like to contribute

Overview of NVML Libraries

- libpmemobj – transactional object store
- libpmemblk – provides arrays of atomically updated fixed size blocks
- libpmemlog – atomic append to log
- libpmem – low level support for rest of libraries
- libvmem – a volatile memory pool from a DAX mapped file

- http://pmem.io/nvml

Call to Action

- PM is an exciting new technology
- PM is a disruptive technology
- Performance tradeoffs
  - Significant storage performance improvement without application modification
  - Even better performance improvements possible with application modification
- **Windows supports PM today**
  - What are you doing to be ready?
  - Engage with your preferred OEM about their PM platform support