SNIA PERSISTENT MEMORY PM SUMMIT

JANUARY 24, 2018 | SAN JOSE, CA

Windows Persistent Memory Support

Neal Christiansen
Microsoft

Agenda

- Review: Existing Windows PM Support
- What’s New
  - New PM APIs
  - Large & Huge Page Support
  - Dax aware Write-ahead LOG
  - Improved Driver Model
  - Uncorrectable Error Handling
  - Hyper-V & NVML Support

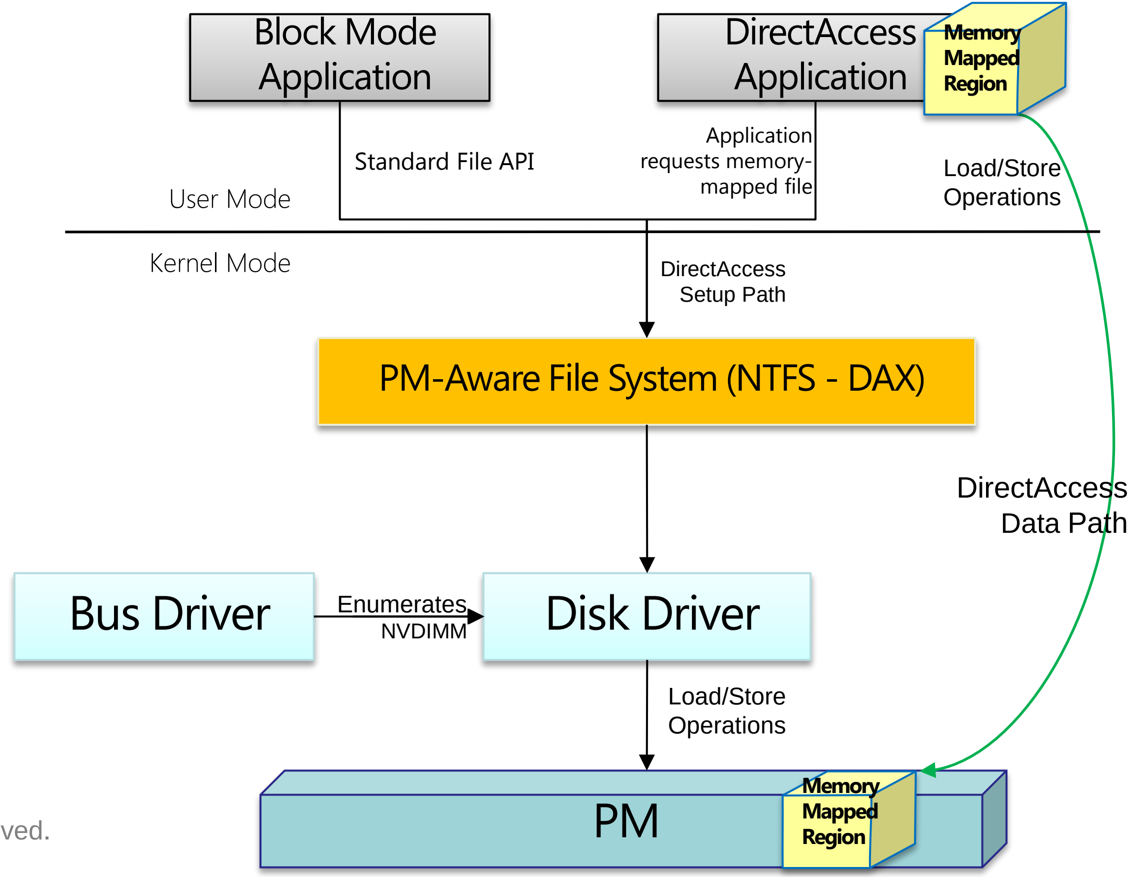
How DAX Works

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



Windows DAX Volume Support

- DAX mode is chosen at volume format time
  - Why: compatibility issues with existing components, examples:
    - File system filters
    - Bitlocker (volume level software encryption)
    - Volsnap (volume snapshot provider)
  - Some existing functionality is lost
  - DAX Volumes are only supported by the NTFS file system

IO on DAX Volumes

- Memory Mapped IO
  - Memory mapped sections map directly to PM hardware
- Cached IO
  - Cache Manager maps directly to persistent memory
  - Copies directly between user’s buffer and persistent memory
- Non-Cached IO
  - Is converted to cached IO by the file system

Impacts to File System Functionality on DAX Volumes

- Direct access to PM by applications eliminates the traditional hook points that file systems use to implement various features
- File System functionality that is not available on DAX enabled volumes in direct access mode:
  - No NTFS software encryption support (EFS)
  - No NTFS software compression support
  - No NTFS TxF support (Transactional NTFS)
  - No NTFS USN (change journal) range tracking of memory mapped files
  - No NTFS resident file support

Impacts to File System Functionality on DAX Volumes

- 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

Block Mode Volumes on PM Hardware

Is backwards compatible

- Maintains existing storage semantics
  - All IO operations traverse the storage stack to the PM disk driver
  - Sector atomicity guaranteed by the PM disk driver
    - Uses BTT – Block Translation Table
  - Has shortened path length through the storage stack to reduce latency
- Fully compatible with existing applications
- Supported by all Windows file systems
- Works with existing file system filter and volume filter drivers

Agenda

- Review: Existing Windows PM Support
- What's New
  - New PM APIs
  - Large & Huge Page Support
  - Dax aware Write-ahead LOG
  - Improved Driver Model
  - Uncorrectable Error Handling
  - Hyper-V & NVML Support

New Flush APIs for DAX Mapped Regions

- Available from both user and kernel modes
  - User-mode versions do not transition to kernel mode to flush
- Performs necessary work to optimally flush PM contents from CPU caches
  - Optimized for given hardware architecture and implementation
- MSDN documentation available at:
  - https://msdn.microsoft.com/en-us/library/windows/hardware/ff553354(v=vs.85).aspx

Rtl APIs for Flushing DAX mappings

- RtlGetNonVolatileToken
  - Token stores properties about the given DAX region
  - User-mode version performs a single OS call
- RtlFreeNonVolatileToken
- RtlFlushNonVolatileMemory
- RtlDrainNonVolatileFlush
  - Allows parallel flushing
- RtlFlushNonVolatileMemoryRanges
- RtlWriteNonVolatileMemory

DAX Specific File System Operations

- NUMA support
  - Windows requires a PM disk to reside on a single NUMA node
  - FSCTL_QUERY_VOLUME_NUMA_INFO
    - Returns the NUMA node the given DAX volume resides on
- Bad block detection
  - FSCTL_QUERY_BAD_RANGES
    - Returns those regions of a file that have bad PM blocks

Agenda

- Review: Existing Windows PM Support
- What’s New
  - New PM APIs
  - Large & Huge Page Support
  - Dax aware Write-ahead LOG
  - Improved Driver Model
  - Uncorrectable Error Handling
  - Hyper-V & NVML Support

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 Large Page Support

- DAX partitions are now aligned to 2mb boundaries
- NTFS natively 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
  - A 2mb cluster size is recommended for optimal Large and Huge page support
  - Large and Huge page alignment is supported on cluster sizes <2mb

Windows Huge Page Support

- Huge page alignment has to be requested
  - FSCTL_SET_DAX_ALLOC_ALIGNMENT_HINT
    - Allows an application to specify alignment requirements for a file
    - Can specify a primary and fallback alignment
      - ex: Prefer Huge pages but Large pages are OK
    - Can specify if the alignment requirements are mandatory or not
    - Can specify a file offset where the alignment requirements begin
      - Ex: VHDX file alignment requirements start after the VHDX header
    - Alignment request will be honored regardless of Cluster Size
- Can I see how my file is aligned?
  - FSCTL_QUERY_FILE_REGIONS
    - Allows an application to query the alignment state of a file
    - fsutil dax queryfilealignment <filename> [options]

Agenda

- Review: Existing Windows PM Support
- What's New
  - New PM APIs
  - Large & Huge Page Support
  - Dax aware Write-ahead LOG
  - Improved Driver Model
  - Uncorrectable Error Handling
  - Hyper-V & NVML Support

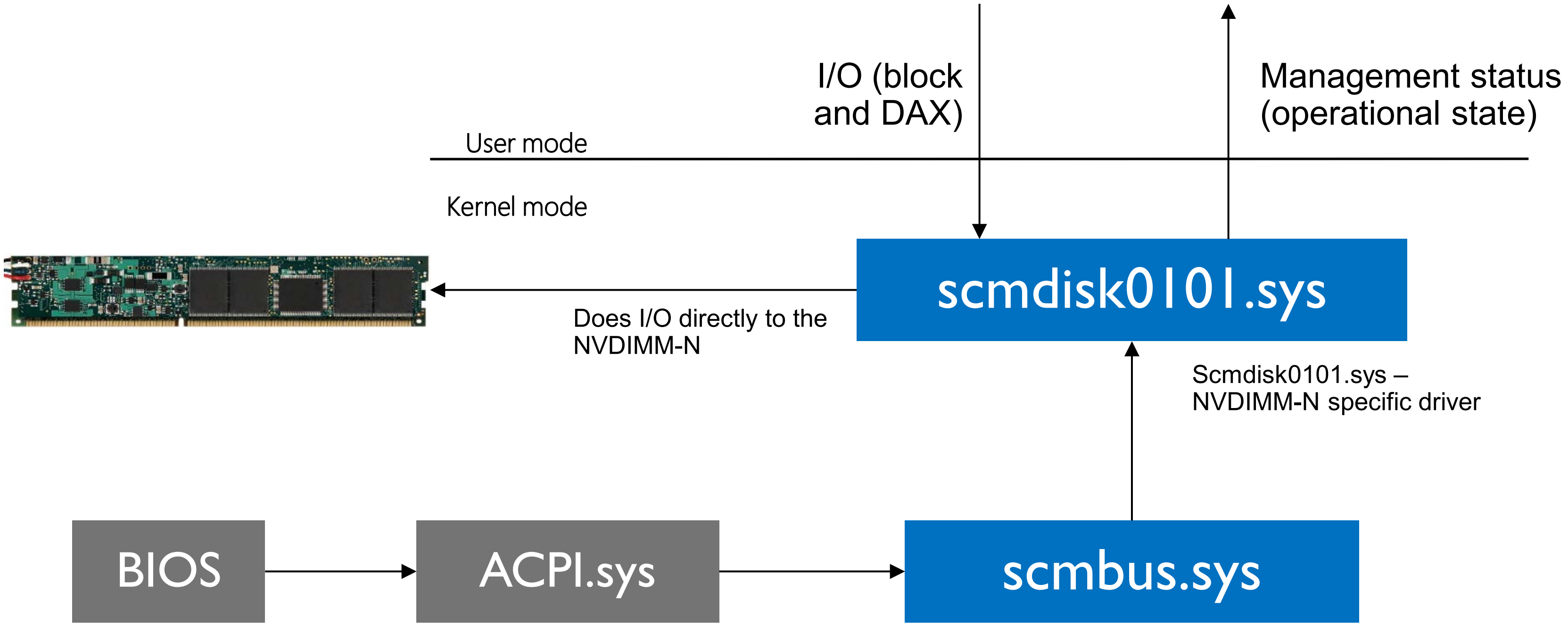
Dax Aware Write-ahead LOG

- NTFS is a journaled file system with a Write-ahead LOG for resiliency
- In Server 2016 all LOG writes were sent down the storage stack
  - WriteThrough operations on a DAX volume did not perform as desired
    - WriteThrough is where all data and metadata is durably committed before the operation returns
- The NTFS Write-ahead LOG is now memory mapped directly to persistent memory
  - LOG updates are now immediately durable
  - WriteThrough performance improvements

Agenda

- Review: Existing Windows PM Support
- What's New
  - New PM APIs
  - Large & Huge Page Support
  - Dax aware Write-ahead LOG
  - Improved Driver Model
  - Uncorrectable Error Handling
  - Hyper-V & NVML Support

Windows Server 2016 Driver Architecture



I/O (block and DAX)

Management status (operational state)

User mode

Kernel mode

scmdisk0101.sys

Does I/O directly to the NVDIMM-N

Scmdisk0101.sys – NVDIMM-N specific driver

BIOS

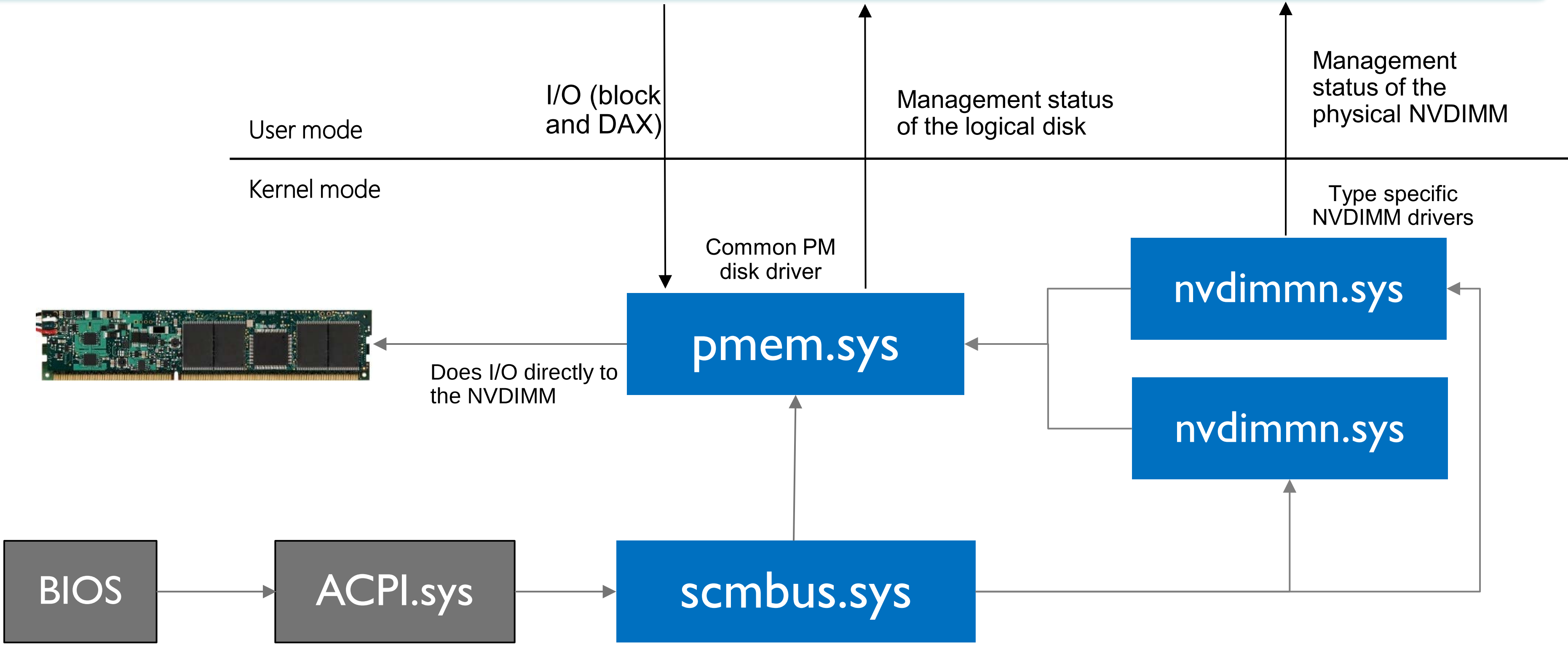
ACPI.sys

scmbus.sys

New Architecture

- For NVDIMM-N, ScmDisk0101.sys is replaced by two drivers
  - **Pmem.sys**: controls a byte-addressable interleave set and is responsible for all I/O, BTT etc.
  - **NvdimmN.sys**: controls a physical NVDIMM-N and is responsible for monitoring its health
    - There is one physical NVDIMM PDO (physical device object) per physical NVDIMM on the system
    - On a system with interleaved NVDIMM-Ns, there will be one pmem.sys PDO and two nvdimmn.sys PDOs
- Physical NVDIMMs are a new device stack, with a new management experience
  - New IOCTL interface

Driver Architecture for Server vNext


User mode
I/O (block and DAX)
Management status of the logical disk
Management status of the physical NVDIMM
Kernel mode
Type specific NVDIMM drivers
Common PM disk driver
pmem.sys
nvdimmn.sys
nvdimmn.sys
Does I/O directly to the NVDIMM
BIOS
ACPI.sys
scmbus.sys

Benefits of the New Architecture

- Easy to support new NVDIMM types
  - Only have to write a physical NVDIMM driver; pmem.sys doesn’t change
- Clear separation of responsibilities
  - Pmem.sys manages the logical disk functionalities
  - Physical NVDIMM drivers manage physical devices

New PM Management Support

- Powershell support for managing physical and logical persistent memory devices
  - Ability to enumerate, create and delete logical persistent memory devices (i.e. namespaces) on persistent memory devices
  - Ability to enumerate physical persistent memory devices on system
  - Example powershell cmdlets (these names are subject to change):
    - Get-PmemDisk
    - New-PmemDisk
    - Remove-PmemDisk
    - Get-PmemPhysicalDevice
    - Initialize-PmemPhysicalDevice
    - Get-PmemUnusedRegion

Agenda

- Review: Existing Windows PM Support
- What's New
  - New PM APIs
  - Large & Huge Page Support
  - Dax aware Write-ahead LOG
  - Improved Driver Model
  - Uncorrectable Error Handling
  - Hyper-V & NVML Support

Uncorrectable Error Handling

- Server 2016 was limited to boot-time detection only
- Runtime detection now supported
- For detected bad pages the PM disk driver:
  - Fails Block IOs
  - If not memory mapped:
    - Fails future mapping requests
  - If memory mapped:
    - Asks the memory manager to unmap the given page
    - Unmapping by memory manager is best effort

Agenda

- Review: Existing Windows PM Support
- What’s New
  - New PM APIs
  - Large & Huge Page Support
  - Dax aware Write-ahead LOG
  - Improved Driver Model
  - Uncorrectable Error Handling
  - Hyper-V & NVML Support

Hyper-V and NVML Support

- For additional information on this topic please see Tom Talpey’s presentation at 12:10pm today

Availability of Persistent Memory Support

- Client:
  - August, 2016: Windows 10 Anniversary Update
  - April, 2017: Windows 10 Creators Update
  - October, 2017: Windows 10 Fall Creators Update
- Server:
  - September, 2016: Windows Server 2016

Questions