Update on Windows Persistent Memory Support

Neal Christiansen
Microsoft
Agenda

- What is Persistent Memory (PM)
- Review: Existing Windows PM Support
- What’s New
  - New PM APIs
  - Large Page Support
  - Hyper-V Support
  - NVML on Windows
  - Improved Driver Model
  - Uncorrectable Error Handling
- Industry Standards Support
Reminder: 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)
  - Non-Volatile Memory (NVM)
  - Persistent Memory (PM)  Industry converging on this term
Agenda

- What is Persistent Memory (PM)
- Review: Existing Windows PM Support
- What’s New
  - New PM APIs
  - Large Page Support
  - Hyper-V Support
  - NVML on Windows
  - Improved Driver Model
  - Uncorrectable Error Handling
- Industry Standards Support
Support for JEDEC-defined NVDIMM-N devices

- Released:
  - August, 2016: Windows 10 Anniversary Update
  - September, 2016: Windows Server 2016
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
Memory Mapped IO on DAX Volumes

- Memory mapped sections map directly to PM hardware
  - No change to existing memory mapping APIs
- An application has direct access to persistent memory
  - Maximizes performance
Cached IO on DAX Volumes

- One Copy IO via the Cache manager
  - Creates a cache map that maps directly to persistent memory
  - Copies directly between user’s buffer and persistent memory
Non-cached IO on DAX Volumes

- Is converted to cached IO by the file system
  - Cache manager copies directly between user’s buffer and persistent memory
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 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 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
  - Functionality not currently supported on DAX volumes:
    - Sparse Files
    - Defragging files that are memory mapped
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 on DAX Volumes

- To minimize compatibility issues:
  - No existing filters will receive notification when a DAX volume is mounted
  - At filter registration time filters may indicate via a new registration flag that they understand DAX volume semantics
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
  - 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 and volume filter drivers
Sector Atomicity

- BTT – Block Translation Table
  - Algorithm created by Intel and standardized in UEFI 2.7
  - Provides 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
    - Has performance impact
  - 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
Agenda

- What is Persistent Memory (PM)
- Review: Existing Windows PM Support
- What’s New
  - New PM APIs
  - Large Page Support
  - Hyper-V Support
  - NVML on Windows
  - Improved Driver Model
  - Uncorrectable Error Handling
- Industry Standards Support
New Flush APIs for DAX mapped regions

- Available from both user and kernel modes
  - Usermode APIs do not transition to kernel mode
- Performs necessary work to optimally flush PM contents from CPU caches
  - Optimized for given hardware architecture
- MSDN documentation available at:
Rtl APIs for Flushing DAX mappings

- RtlGetNonVolatileToken
- RtlFreeNonVolatileToken
- RtlFlushNonVolatileMemory
- RtlDrainNonVolatileFlush
- RtlFlushNonVolatileMemoryRanges
- RtlWriteNonVolatileMemory
RtlGetNonVolatileToken

NTSTATUS
RtlGetNonVolatileToken (  
VOID *NvBuffer,  
SIZE_T Size,  
VOID **NvToken);

- Stores properties about the given DAX region in the returned NvToken object  
- If given region is not DAX mapped, usermode callers will fail with STATUS_INVALID_PARAMETER
RtlFreeNonVolatileToken

NTSTATUS
RtlFreeNonVolatileToken (VOID *NvToken);

- Frees token returned by RtlGetNonVolatileToken
RtlFlushNonVolatileMemory

NTSTATUS
RtlFlushNonVolatileMemory (VOID *NvToken,
VOID *NvBuffer,
SIZE_T Size,
ULONGLONG Flags);

- Optimally flushes the given DAX region
- Flag `FLUSH_NV_MEMORY_IN_FLAG_NO_DRAIN` tells the routine to not wait for the flush to drain (via the SFENCE instruction)
  - Allows for efficient flushing of multiple regions
RtlDrainNonVolatileFlush

NTSTATUS
RtlDrainNonVolatileFlush (VOID *NvToken);

- Waits for previous flush operations to complete (via the SFENCE instruction)
- Supports efficient flushing of multiple regions
RtlFlushNonVolatileMemoryRanges

```c
NTSTATUS
RtlFlushNonVolatileMemoryRanges ( 
    VOID *NvToken,
    NV_MEMORY_RANGE *NvRanges,
    SIZE_T TotalRanges,
    ULONG Flags);
```

typedef struct _NV_MEMORY_RANGE {
    VOID *BaseAddress;
    SIZE_T Length;
} NV_MEMORY_RANGE;

- Only one drain operation is issued across all ranges
- Flag `FLUSH_NV_MEMORY_IN_FLAG_NO_DRAIN` tells the routine to not wait for the flushes to drain (via the SFENCE instruction)
RtlWriteNonVolatileMemory

NTSTATUS
RtlWriteNonVolatileMemory (  
    VOID *NvToken,
    VOID UNALIGNED *NvDestination,
    VOID UNALIGNED *Source,
    SIZE_T Size,
    ULONG Flags);

- Functionally equivalent to memcpy()
- Does **not** flush the copied memory
NUMA information FSCTL

FSCTL_QUERY_VOLUME_NUMA_INFO

typedef struct _FSCTL_QUERY_VOLUME_NUMA_INFO_OUTPUT {
    ULONG NumaNode;
} FSCTL_QUERY_VOLUME_NUMA_INFO_OUTPUT;

Windows requires a PM disk to reside on a single NUMA node
Returns the NUMA node the given DAX volume resides on
Agenda

- What is Persistent Memory (PM)
- Review: Existing Windows PM Support
- What’s New
  - New PM APIs
  - Large Page Support
  - Hyper-V Support
  - NVML on Windows
  - Improved Driver Model
  - Uncorrectable Error Handling
- Industry Standards Support
What are Large Pages

- Modern CPUs manage memory using 4K pages
- An application's memory usage is managed via page tables controlled by the operating system's 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.
Windows Large Page Support

- DAX partitions are now aligned to 2mb boundaries
- NTFS generically supports cluster sizes up to 2mb (in powers of 2)
  - Former limit was 64K
- A file memory mapped on a DAX volume with a 2mb cluster size is guaranteed to be mapped by the memory manager using Large Pages
- Huge page support will be available in the future
  - Huge pages are 1gb in size
Agenda

- What is Persistent Memory (PM)
- Review: Existing Windows PM Support
- What’s New
  - New PM APIs
  - Large Page Support
  - Hyper-V Support
  - NVML on Windows
  - Improved Driver Model
  - Uncorrectable Error Handling
- Industry Standards Support
Hyper-V PM Support

- Windows & Linux guests in generation 2 VMs see virtual PMEM (vPMEM) devices
- New VHD file type: .VHDPMEM
  - Can convert between VHDX and VHDPMEM formats (using `convert-vhd` PowerShell cmdlet)
  - Admin decide at create time if the VHDPMEM file has a BTT or not
  - VHDPMEM files can be mounted as a SCSI device on the host for read/write access
- Each vPMEM device is backed by one .VHDPMEM file
Hyper-V PM Support

- DAX and BTT programming models, including Win32 APIs and NVML, supported from guest
- Uses large pages automatically, when available
- No support for VM meta-operations including:
  - Live Migration
  - Checkpoints
  - Backup
  - Save/Restore
Agenda

- What is Persistent Memory (PM)
- Review: Existing Windows PM Support
- What's New
  - New PM APIs
  - Large Page Support
  - Hyper-V Support
  - NVML on Windows
  - Improved Driver Model
  - Uncorrectable Error Handling
- Industry Standards Support
NVML on Windows

- NVML is an open source library originally implemented by Intel
  - Available for Windows and Linux via GitHub
  - [https://github.com/pmem/nvml/](https://github.com/pmem/nvml/)
- Microsoft is working with Intel, HPE and HP Labs on the Windows port
  - It is feature complete
NVML on Windows

- 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
  - Makes its own atomicity guarantees
  - Works in both PM and non-PM hardware environments

- Use Case
  - Alternative to handling flushing or creating PM-aware data structures yourself
  - Simpler application development
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

http://pmem.io/nvml
Agenda

- What is Persistent Memory (PM)
- Review: Existing Windows PM Support
- What’s New
  - New PM APIs
  - Large Page Support
  - Hyper-V Support
  - NVML on Windows
  - Improved Driver Model
  - Uncorrectable Error Handling
- Industry Standards Support
Windows Server 2016 architecture

User mode

Kernel mode

I/O (block and DAX)

Management status (operational state)

BIOS → ACPI.sys → scmbus.sys

Does I/O directly to the NVDIMM-N

scmdisk0101.sys

Scmdisk0101.sys – NVDIMM-N specific driver
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 per physical NVDIMM on the system
    - On a system with two 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
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 Driver Architecture (post-Window Server 2016)

User mode

I/O (block and DAX)

Management status of the logical disk

Kernel mode

Common PM disk driver

Management status of the physical NVDIMM

Type specific NVDIMM drivers

Does I/O directly to the NVDIMM

pmem.sys

nvdimmn.sys

nvdimmn.sys

BIOS

ACPI.sys

scmbus.sys

2017 Storage Developer Conference. © Microsoft Corporation. All Rights Reserved.
Agenda

- What is Persistent Memory (PM)
- Review: Existing Windows PM Support
- What’s New
  - New PM APIs
  - Large Page Support
  - Hyper-V Support
  - NVML on Windows
  - Improved Driver Model
- Uncorrectable Error Handling
- Industry Standards 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 unmapped:
    - Fails future mapping requests
  - If mapped:
    - Asks the memory manager to unmap the given page
    - Unmapping by memory manager is best effort
Agenda

- What is Persistent Memory (PM)
- Review: Existing Windows PM Support
- What’s New
  - New PM APIs
  - Large Page Support
  - Hyper-V Support
  - NVML on Windows
  - Improved Driver Model
  - Uncorrectable Error Handling
- Industry Standards Support
Industry Standards Support

- JEDEC JESD 245, 245A: Byte Addressable Energy Backed Interface
  - Defines the host to device interface and features supported for a NVDIMM-N
- UEFI 2.5 – 2.7
  - Label format
  - Block Translation Table (BTT): sector atomicity support
- ACPI 6.0 – 6.2
  - NVDIMM Firmware Interface Table (NFIT)
  - NVM Root and NVDIMM objects in ACPI namespace
  - Address Range Scrub (ARS)
    - Uncorrectable memory error handling
  - Notification mechanism for NVDIMM health events and runtime detected uncorrectable memory error
Upcoming Windows Release Information

- New support to be Released:
  - Oct 17, 2017: Windows 10 “Fall Creators Update”
  - Windows Server Fall Update
Questions