Accelerating SQL Server with Persistent Memory

Lee Prewitt

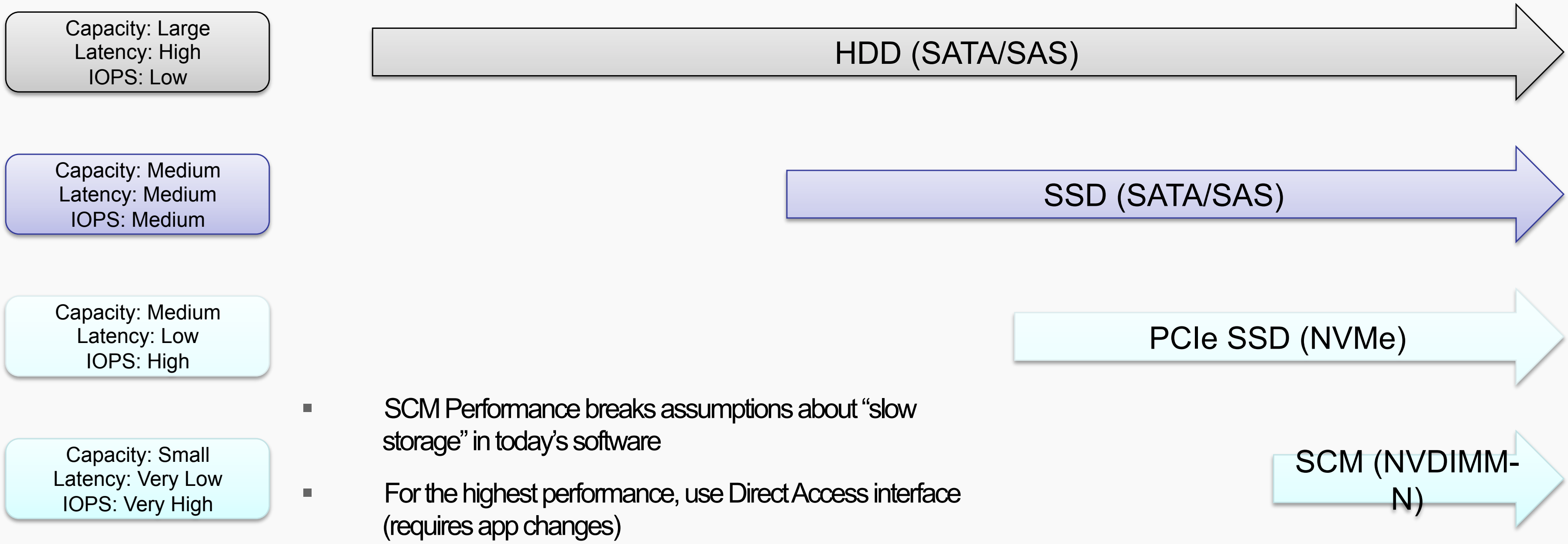
Principal Program Manager

Microsoft

Flash Memory Summit

Technology Evolution

- Storage technology has made significant strides (capacity, latency, IOPS).



Capacity: Large
Latency: High
IOPS: Low

HDD (SATA/SAS)

Capacity: Medium
Latency: Medium
IOPS: Medium

SSD (SATA/SAS)

Capacity: Medium
Latency: Low
IOPS: High

PCIe SSD (NVMe)

Capacity: Small
Latency: Very Low
IOPS: Very High

SCM (NVDIMM-N)

- SCM Performance breaks assumptions about “slow storage” in today’s software
- For the highest performance, use Direct Access interface (requires app changes)
- For early adoption, utilize it via the block interface (requires no app changes)

File Systems and Storage Class Memory

SCM is a disruptive technology

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

Windows Goals for Storage Class 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
- Make available sector granular failure modes for application compatibility

Flash Memory Summit

Introducing a New Class of Volume

- Direct Access (DAX) Storage Volume
  - Memory mapped files provide applications with direct access to byte-addressable SCM
    - Maximizes performance
  - DAX mode is chosen at volume format time
    - Why: compatibility issues with various components, examples:
      - File system filters
      - BitLocker
      - VolSnap
  - Some existing functionality is lost
  - DAX Volumes are supported by NTFS

SCM Storage Drivers

- New type of volume requires a new driver model
  - SCM Bus Driver
    - Enumerates the physical and logical SCM devices on the system
    - Not part of the IO Path
  - SCM Disk Drivers
    - Driver for logical SCM devices
    - Storage abstraction layer to rest of the OS
    - Hardware-specific
      - Supports both in-box or vendor-specific drivers
    - Windows will use a native 4K sector size
- Introduces new interfaces
  - Expose byte addressable storage functionality
  - Supports management of SCM hardware

Flash Memory Summit

Memory Mapped IO in DAX mode

- On DAX formatted volumes memory mapped sections map directly to SCM 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
  - The file system returns YES when:
    - The volume resides on SCM hardware
    - The volume has been formatted for byte addressable mode

Memory Mapped IO in DAX mode

- When a section is created in DAX mode
  - 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 storage stack 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

- When cached IO is requested on a DAX enabled volume the cache manager creates a cache map that maps directly to SCM hardware
- Cache manager copies directly between the 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

- Sends IO operations down the storage stack to the SCM storage driver
  - Maintains existing failure semantics for application compatibility
  - Is coherent with cached and memory mapped IO

File System Metadata in DAX Mode

- 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

- The file system no longer knows when writeable memory mapped sections are 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 Journal (change journal)
    - Signaling directory change notification

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 encryption support (EFS)
  - No NTFS compression support
  - No NTFS TxF support
  - No NTFS USN range tracking of memory mapped files
  - No NTFS resident file support

Flash Memory Summit

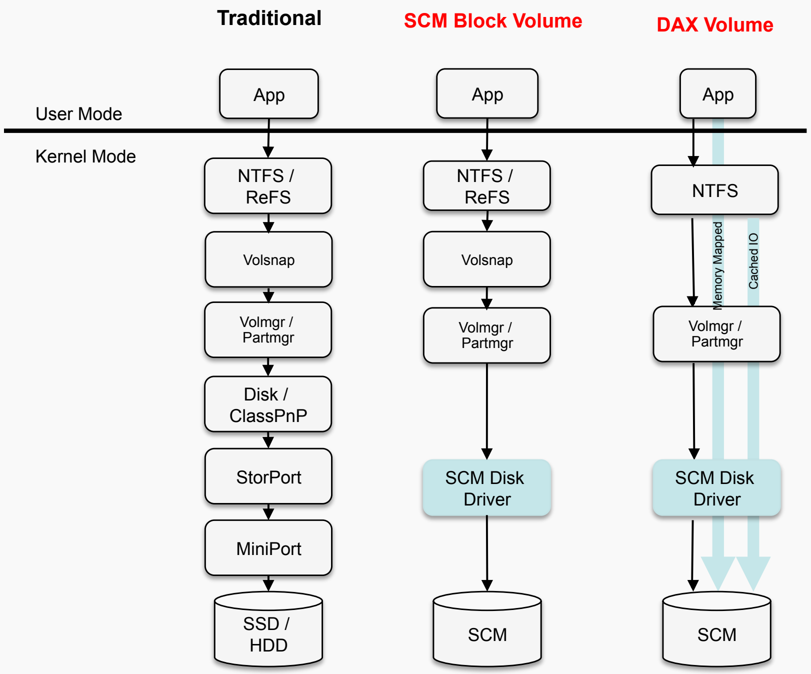
Backward Compatibility with SCM Hardware

- Block Mode Volumes
  - Maintains existing storage semantics
    - All IO operations traverse the storage stack to the SCM driver
    - Has a shortened path length through the storage stack
      - No storport or miniport drivers (too much latency)
      - No SCSI translations
  - Fully compatible with existing applications
  - Supported by all Windows file systems
  - Works with existing file system and storage filters
  - Block mode vs. DAX mode is chosen at format time

New Volume Device Class (ScmVolume)

- New byte addressable partition type
  - Set at format time
- Why: Prevents non-DAX aware components from attaching to this new volume class
  - VOLSNAP – no support for volume snapshots
  - BITLOCKER – no support for software encryption
  - 3rd Party volume stack filters
  - Improves performance by removing non-DAX aware drivers

IO Stack Comparisons



Flash Memory Summit

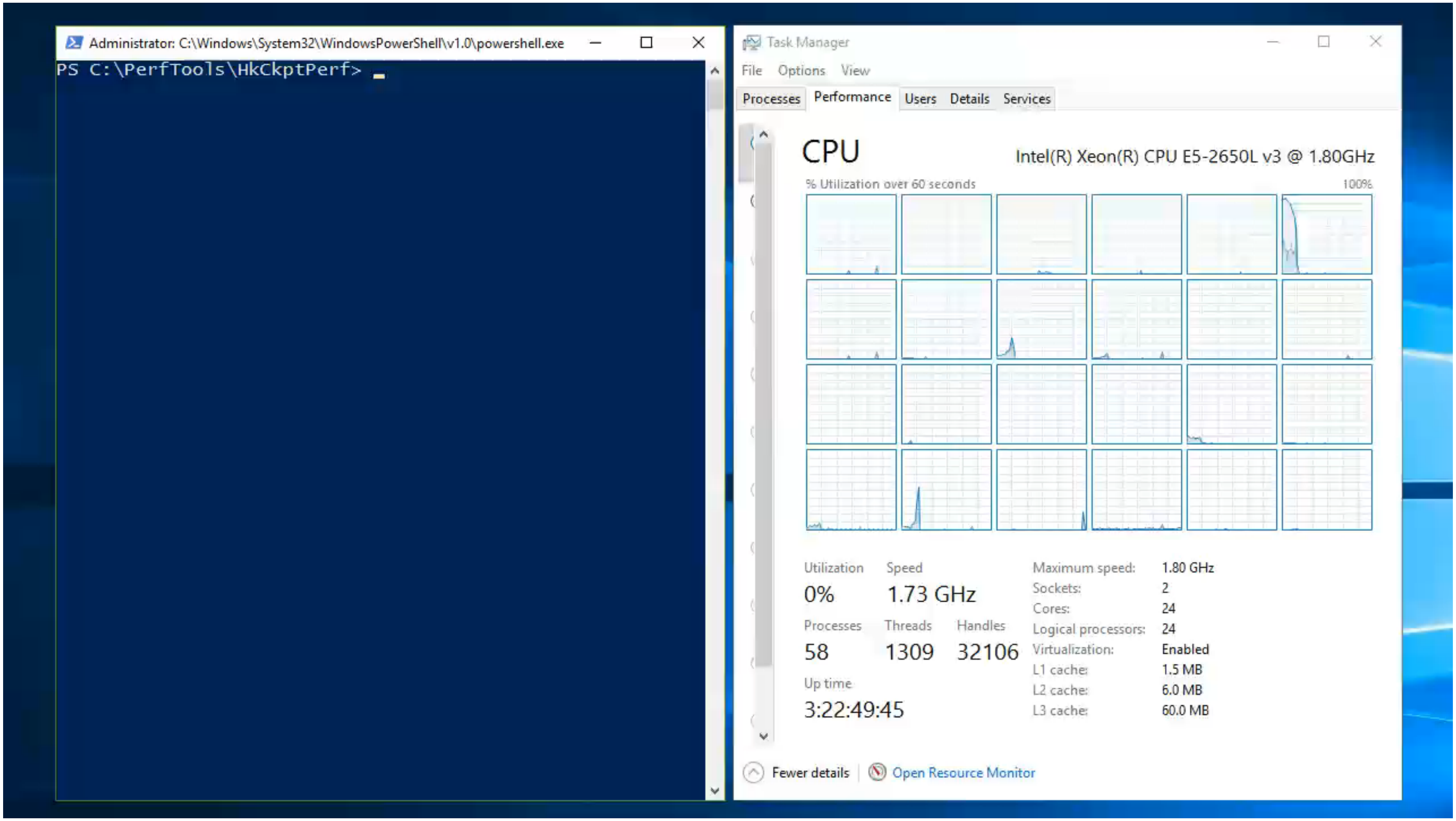
More Transactions

- How do I increase my transaction throughput?
- How do I reduce my transaction latency?

- Options:

More CPU!	2S system → 4S system
More Memory!	128GB → 256GB → 1TB+
Faster Storage!	HDD → SATA SSD → NVMe SSD

  - SQL Server 2016 “Tail Of Log” Preview on Persistent Memory



```
PS C:\PerfTools\HkCkptPerf> _
```

CPU Intel(R) Xeon(R) CPU E5-2650L v3 @ 1.80GHz

Utilization 0%

Speed 1.73 GHz

Processes 58

Threads 1309

Handles 32106

Up time 3:22:49:45

Maximum speed: 1.80 GHz
Sockets: 2
Cores: 24
Logical processors: 24
Virtualization: Enabled
L1 cache: 1.5 MB
L2 cache: 6.0 MB
L3 cache: 60.0 MB

Faster Transaction Processing via Persistent Memory Use

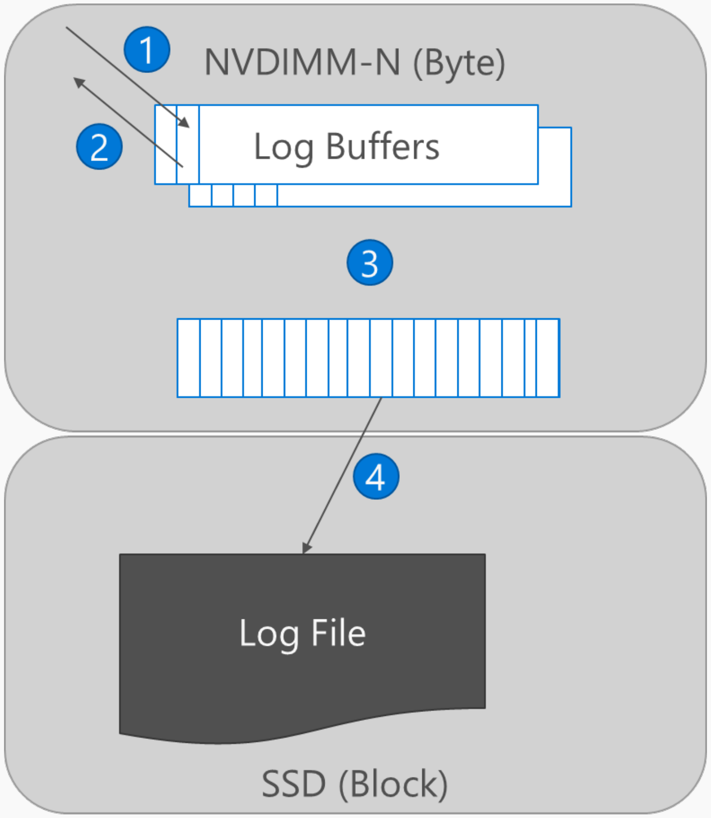
In the Past:

- 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 ToL Preview:

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

Blue indicates the critical path for a transaction



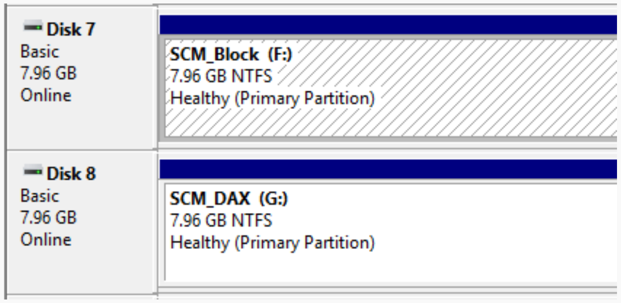
Native NVDIMM-N Support in Server 2016

- Windows exposes a latency-optimized "disk" device (block interface)
- DirectAccess (DAX): enlightened apps (SQL 2016) can directly access their data on the Persistent Memory (PM) device via Load/ Store instructions
- Use of DAX on NVDIMM-N provides DRAM-like performance

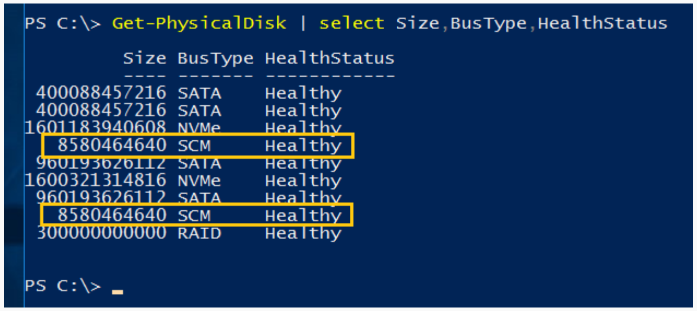
4K Random Write	Thread Count	IOPS	Latency (us)
NVDIMM-N (Block)	1	187,302	5.01
NVDIMM-N (DAX)	1	1,667,688	0.52

Data gathered on pre-release hardware and software, final results may differ.

Workload: 4KB random writes against a 1GB file on NTFS, MSFT-internal testing tool



Disk 7 Basic 7.96 GB Online	SCM_Block (F:) 7.96 GB NTFS Healthy (Primary Partition)
Disk 8 Basic 7.96 GB Online	SCM_DAX (G:) 7.96 GB NTFS Healthy (Primary Partition)



```
PS C:\> Get-PhysicalDisk | select Size,BusType,HealthStatus

         Size BusType HealthStatus
         ---- ------- ------------
 400088457216 SATA    Healthy
 400088457216 SATA    Healthy
1601183940608 NVMe    Healthy
   8580464640 SCM     Healthy
 960193626112 SATA    Healthy
1600321314816 NVMe    Healthy
 960193626112 SATA    Healthy
   8580464640 SCM     Healthy
 300000000000 RAID    Healthy


PS C:\> _
```