Hewlett Packard
Enterprise

Deploying Persistent Memory on HPE ProLiant servers running Windows Server 2012 R2

Technical white paper

Contents

Abstract

Hewlett-Packard Enterprise (HPE) implements Non-Volatile DIMM (NVDIMM) technology in the HPE 8GB NVDIMM. This new DIMM includes 8 gigabytes of DRAM backed by an equal amount of NAND Flash and delivers the performance of memory with the persistence of storage. HPE ProLiant DL360 and DL380 Gen9 servers shipped with Intel® E5-2600v4 processors are equipped to use the HPE 8GB NVDIMM when running a supported operating system, such as Windows Server 2012 R2 as discussed in this paper.

Introduction

We have introduced HPE Persistent Memory using NVDIMM technology, a medium that provides the extreme performance of DRAM components and the persistence of NAND flash-based components for storage. Close (memory bus) association with the system's processor allows HPE NVDIMMs to function as extreme-performance workload accelerators in the top tier of a storage hierarchy.

Note
JEDEC defines NVDIMM-N technology as energy-backed devices. Microsoft classifies such technology as "storage-class memory."

HPE has developed Windows Server 2012 R2 drivers to allow HPE NVDIMM deployment. This capability allows:

- Most existing user-mode applications to be run without modification
- Sector granular failure modes to preserve application compatibility

Persistent Memory offers a more streamlined operation through a simplified I/O software stack. Figure 1 compares the HPE NVM stack with that of traditional storage volumes. As indicated, standard existing apps can benefit with HPE NVDIMMs while high-performance apps will take full advantage of the streamlined stack.

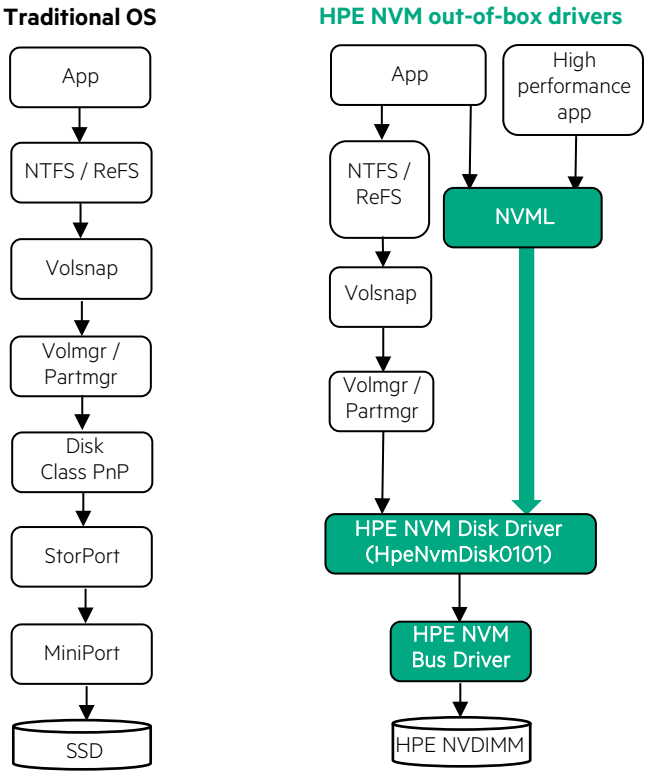


Figure 1. Comparison of traditional OS to HPE NVM I/O stack.

The initial Persistent Memory enablement offered by HPE Windows Server 2012 R2 out-of-box drivers allows HPE NVDIMMs (working with NTFS and ReFS only) to be implemented as follows:

- As a block storage disk interface that delivers very low latency and high throughput for applications compared to SAS or NVMe hard disk drives (HDDs) and solid state drives (SSDs). You can access HPE NVDIMM as a persistent storage disk by formatting with the disk management tool and then accessing with file explorer.
- As a byte-addressable memory interface (i.e., "Memory Mapped Interface"), allowing applications to directly access physical memory locations on the HPE NVDIMM device, enabled on a per-volume basis or interleave access for multiple NVDIMMs. NVM library (NVML) provides API access to custom applications. Note that there is no paging reads or writes with memory mapping.

Hardware/firmware requirements

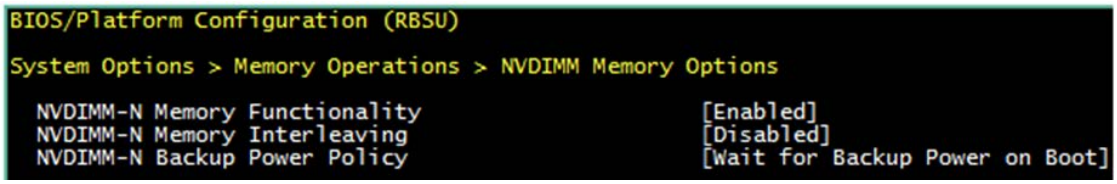
HPE NVDIMMs can be deployed on an HPE ProLiant DL380 Gen9 and HPE ProLiant DL360 Gen9 Servers configured with the following components:

- Factory-installed Intel Broadwell (Xeon® E5-2600 v4) processor(s)
- HPE Smart Storage Battery
- Minimum of one registered DIMM (RDIMM) per processor (LRDIMMs not compatible with NVDIMMs)
- Latest system firmware and HPE iLO 4 firmware versions.

Hardware/firmware configuration

To configure a platform for HPE NVDIMM operation, proceed as follows:

1. Power down the system to be configured. Install HPE NVDIMM(s) as described in the "Installation" section of the *HPE NVDIMM User Guide for HPE ProLiant Gen9 Servers* p/n 860023-001.
2. Boot the system and update system firmware and HPE iLO firmware with Intelligent Provisioning (embedded into ProLiant Gen9 servers and accessed during POST) and Service Pack for ProLiant (SPP).
3. Reboot system and press F9 at the POST display to go to RBSU (via System Utilities).
4. In RBSU, enable NVDIMM (as shown below). Note that previously used NVDIMMs may need to be "sanitized" (i.e., have their NAND memory flushed of any data) before being enabled in the system. UEFI System Utilities includes a NVDIMM sanitizing function. NVDIMM interleaving can also be enabled in RBSU to increase the amount of persistent memory storage space when using multiple NVDIMMs per channel.



At this point, hardware and firmware configuration is complete and the operating system is ready for NVDIMM enablement.

Installing NVDIMM drivers

To enable storage capability on HPE NVDIMM devices on a Windows Server 2012 R2 environment the following HPE NVM drivers need to be installed:

- HPE NVM Bus Driver – responsible for the enumeration of physical and logical SCM devices on the host. This driver is not part of the data path.
- HPE NVM Disk Driver – implements the storage abstraction layer, and handles all management and data path to the logical storage backed by the NVDIMM-N devices. It allows both traditional block storage access (currently using a 512B sector size) and new byte-addressable storage access.

Existing applications like SQL server can re-direct their transaction logs to the NVDIMM drive. This requires no application change and still offers high performance. Performance will also improve if a custom application can be changed to use NVML API calls to directly access NVDIMM with the NVDIMM byte-addressable driver.

To install the drivers:

1. Download each component to the unit.
2. Run the component as administer.
3. When prompted, select **Install**

Windows Server 2012 R2 HPE NVDIMM enablement

Figure 2 shows how Windows Device Manager presents NVDIMMs as NVM Byte-Addressable Memory Disk Devices on a system using the out-of-box driver from HPE. Two NVDIMMs are indicated in the Device Manger window.

Note
This initial NVDIMM enablement offered by the Windows Server 2012 R2 out-of-box driver allows individual NVDIMM devices and the Host hardware interleaving of NVDIMM devices. Windows Server components like Storage Spaces can be used on NVDIMM devices. NVDIMM devices can be grouped together and create larger simple or mirrored NVDIMM volumes

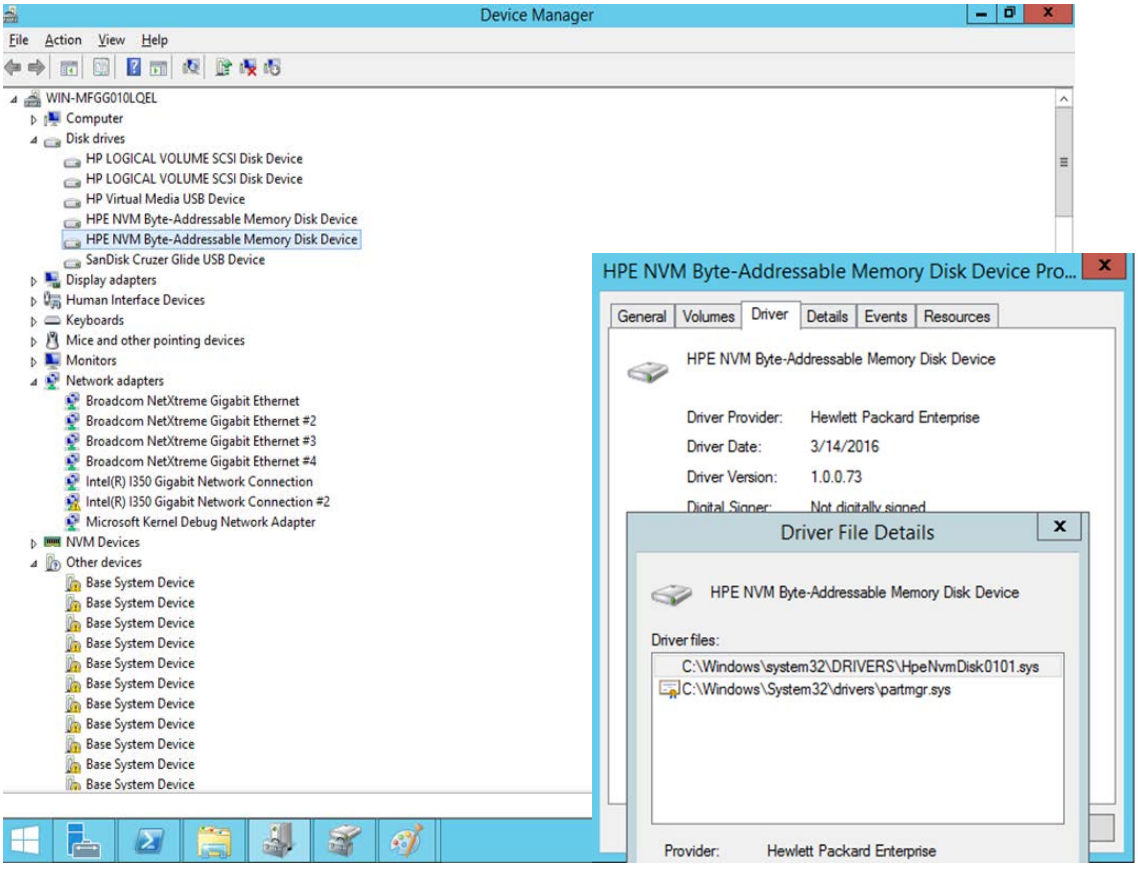


Figure 2. Windows Device Manager screens showing system configuration of two HPE NVDIMMs and device driver details.

A storage volume and then a Windows File System can be created on NVDIMM devices. With Windows Server 2012 R2 and HPE out of box drivers, NTFS and ReFS Windows File Systems support NVDIMM devices. The File System detects at mount time if a given volume resides on an NVDIMM. HPE out of box drivers support byte addressable mode.

Figure 3 shows the Windows Disk Management screen indicating the two NVDIMMs configured as Disk 2 and Disk 3 volumes.

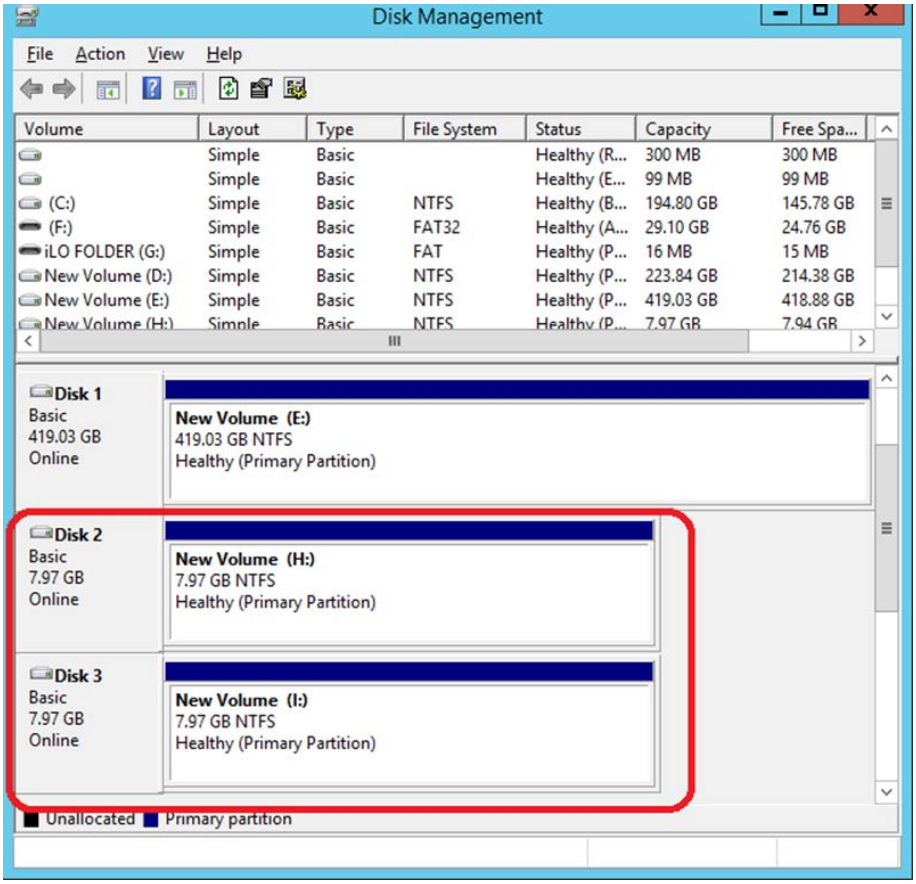


Figure 3. Windows Disk Management screen

HPE NVDIMM-N use scenarios with Windows Server 2012 R2

HPE NVDIMM offers various uses for applications in the Windows Server 2012 R2 environment:

- SQL server can store transaction logs in the NVDIMM devices
- Exchange server can store logs in the NVDIMM device
- NVML can be used for fast storage access

NVML support for HPE NVDIMM-N with Windows Server 2012 R2

HPE offers a set of NVML APIs in the initial NVDIMM-N enablement release for Windows Server 2012 R2 based on early Windows enablement code from http://pmem.io/nvml. Table 1 lists key NVML APIs that have been tested on systems with HPE NVDIMM-Ns and Windows Server 2012 R2 out-of-box driver.

Table 1. Key NVML APIs Tested with Windows Server 2012 R2 and HPE NVDIMMs.

LIBRARY NAME	API NAME	DESCRIPTION
Libpmem	int pmem_is_pmem(void *_addr_, size_t _len_);	Returns true if the entire range [_addr_, _addr+len_) consists of persistent memory. Returns false otherwise.
	void pmem_persist(void *addr, size_t len);	Forces any changes in the range [_addr_, _addr+len_) to be stored durably in persistent memory.
	void *pmem_map(int _fd_);	Maps an entire file for read/write access.
	int pmem_unmap(void *_addr_, size_t _len_);	Deletes all the mappings for the specified address range, and causes further references to addresses within the range to generate invalid memory references.
	void *pmem_memmove_persist(void *_pmemdest_, const void *_src_, size_t _len_);	Provides the same memory copying as **memmove**, and ensure that the result has been flushed to persistence before returning.
	void *pmem_memcpy_persist(void *_pmemdest_, const void *_src_, size_t _len_);	Provides the same memory copying as **memcpy**, and ensure that the result has been flushed to persistence before returning.
	void *pmem_memset_persist(void *_pmemdest_, int _c_, size_t _len_);	Provides the same memory copying as **memset**, and ensure that the result has been flushed to persistence before returning.
libpmemblk	PMEMblkpool *pmemblk_create(const char *_path_, size_t _bsize_, size_t poolsize, mode_t mode);	Creates a block memory pool with the given total _poolsize_ divided up into as many elements of size _bsize_ as will fit in the pool.
	PMEMblkpool *pmemblk_open(const char *_path_, size_t _bsize_);	Opens an existing block memory pool, returning a memory pool handle used with other libpmemblk APIs.
	void pmemblk_close(PMEMblkpool *_pbp_);	Closes the memory pool indicated by _pbp_ and deletes the memory pool handle.
	int pmemblk_read(PMEMblkpool *_pbp_, void *_buf_, off_t _blockno_);	Reads a block from memory pool _pbp_, block number _blockno_, into the buffer _buf_.
	int pmemblk_write(PMEMblkpool *_pbp_, const void *_buf_, off_t _blockno_);	Writes a block from _buf_ to block number _blockno_ in the memory pool _pbp_.
Libpmemobj	PMEMobjpool *pmemobj_create(const char *_path_, const char *_layout_, size_t poolsize, mode_t mode);	Creates a transactional object store with the given total _poolsize_. _path_ and specifies the name of the memory pool file to be created.
	PMEMobjpool *pmemobj_open(const char *_path_, const char *_layout_);	Opens an existing object store memory pool and returns a memory pool handle that can be used with other libpmemobj APIs.
	void pmemobj_close(PMEMobjpool *_pop_);	Closes the memory pool indicated by _pop_ and deletes the memory pool handle.
	void pmemobj_persist(PMEMobjpool *_pop_, void *_addr_, size_t _len_);	Forces any changes in the range [_addr_, _addr+len_) to be stored durably in persistent memory.
	void *pmemobj_memcpy_persist(PMEMobjpool *_pop_, void *_dest_, const void *_src_, size_t _len_);	Provides the same memory copying as **memcpy**, and ensures the result has been flushed to persistence before returning.
	void *pmemobj_memset_persist(PMEMobjpool *_pop_, void *_dest_, int _c_, size_t _len_);	Provides the same memory copying as **memset**, and ensures the result has been flushed to persistence before returning

Refer to http://pmem.io/nvml/ for more details on the APIs above, and for information on other NVML APIs that may be supported in a subsequent release of HPE NVDIMM-Ns and Windows Server 2012 R2 out-of-box drivers.

Storage Spaces

Microsoft has improved Storage Spaces in the Windows 2012 R2 version by adding the ability to use SSDs for automated tiering and/or write-back caching. This provides a highly resilient, scalable, and cost-effective storage solution for business-critical deployment of Storage Space (virtual or physical) and introduces new features and enhancements surrounding scalability and performance of storage virtualization platforms. Storage Spaces can automatically tier data across HDDs and SSDs in JBOD enclosures, which are actually virtual hard drives placed on the storage pools and presented to the operating system as volumes just like a hard drive. Storage space can be enabled in the Windows GUI or in the UEFI Power Shell.

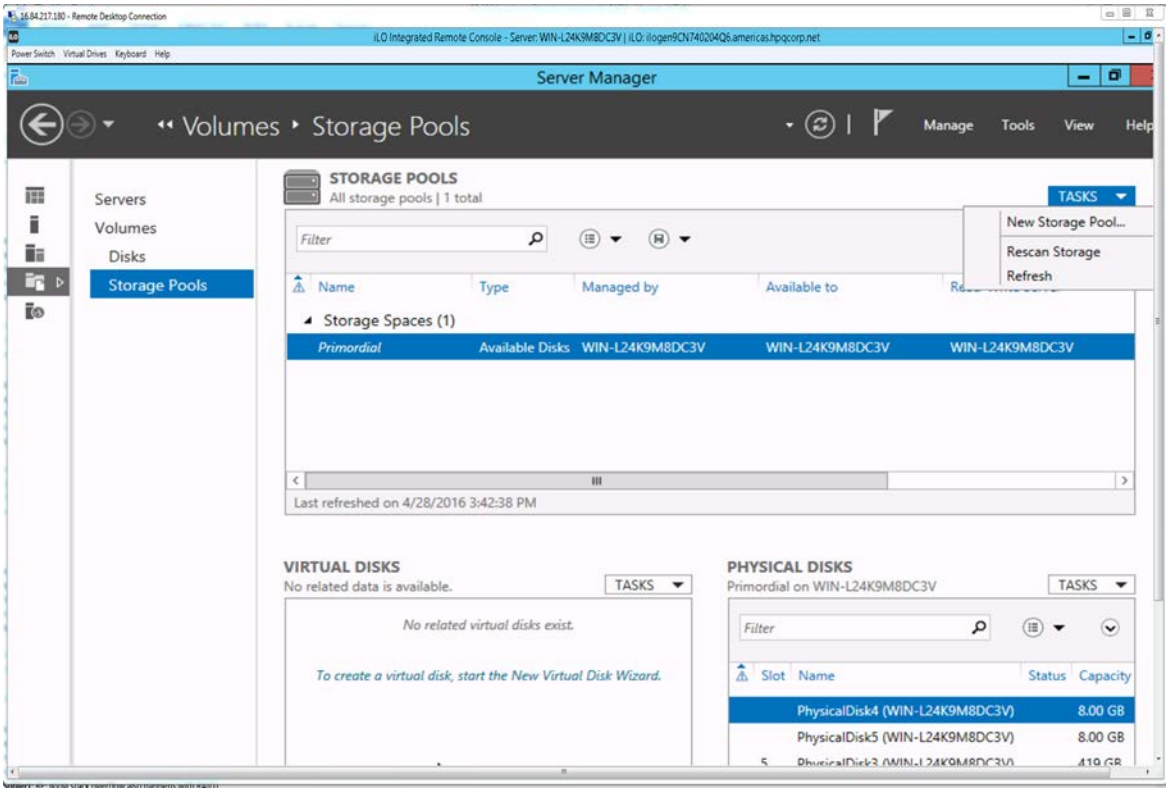
Using the Windows GUI to Enable Storage Spaces

The Windows GUI can be used to create storage and mirror pools.

Storage Pool

To create a storage pool, proceed as follows::

1. In Windows, go to Server Manager and select → File and Storage Service → Servers → Storage Pool →
2. In the Tasks dropdown menu for Storage Pools select New Storage Pool… (as shown below).

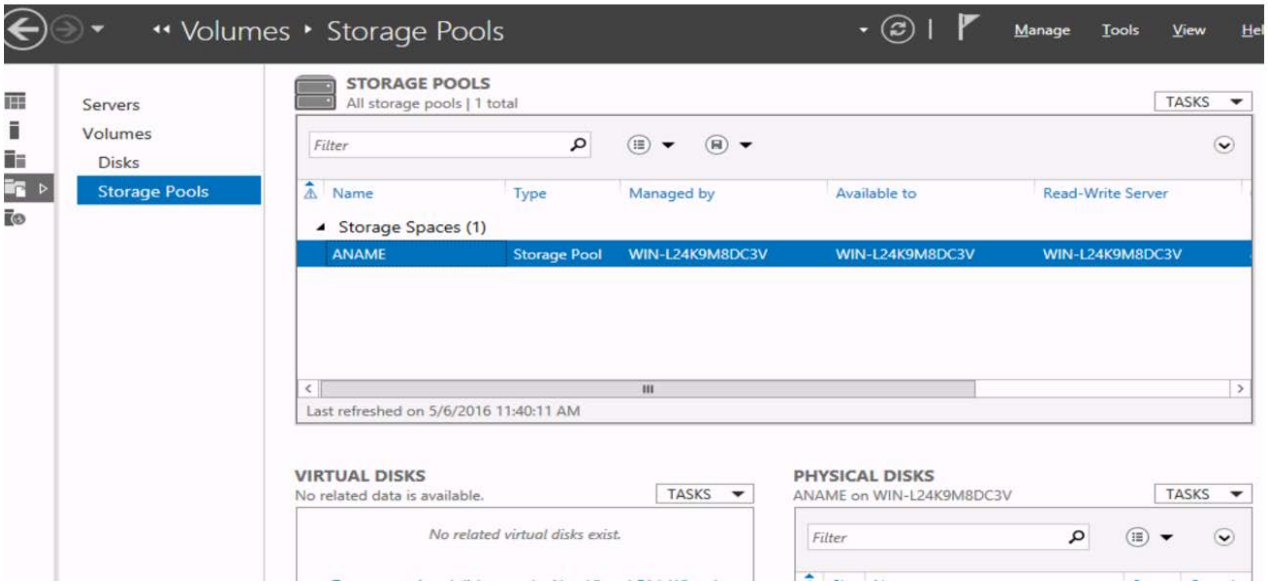


3. Select Before you begin then click Next.
4. Enter a name (example: ANAME) then click Next.
5. Click on Select all drives then click Next.
6. Select Create.

Mirror Pool

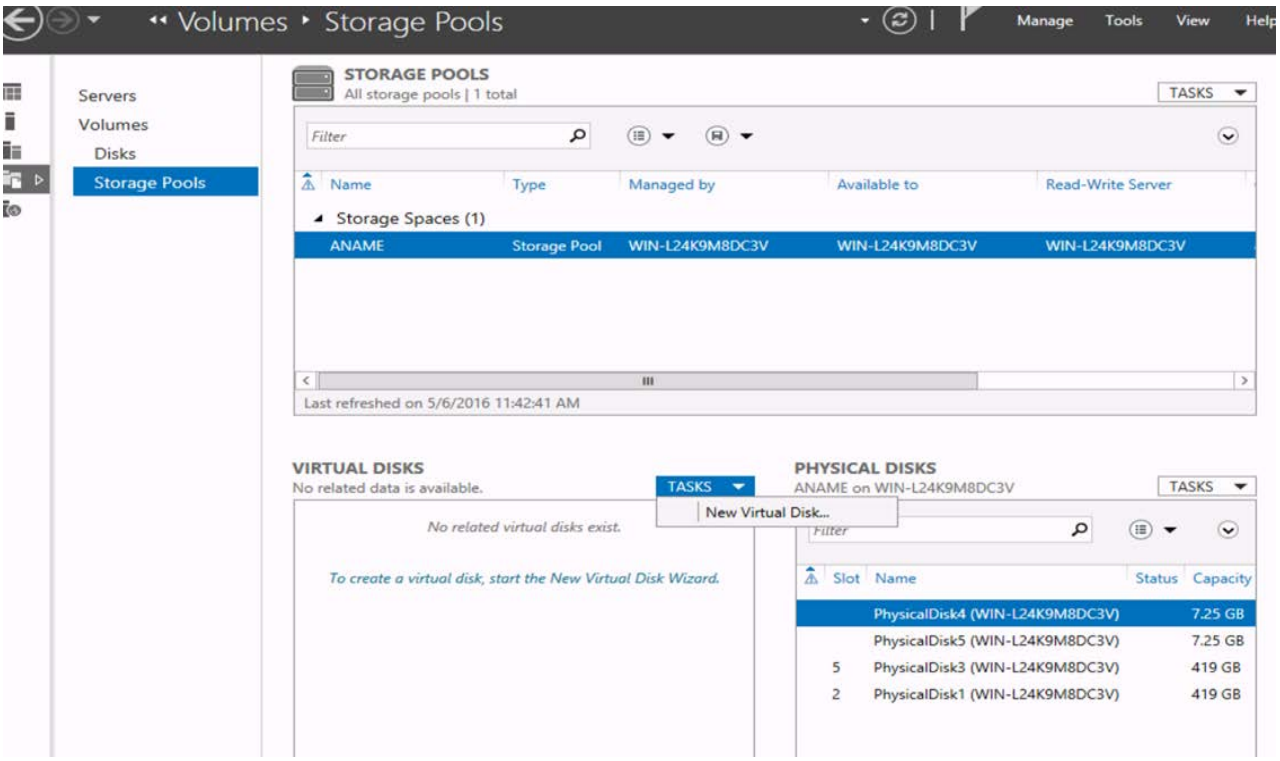
A mirror pool is then created with the following procedure::

1. In Windows Server Manager (same GUI as before), navigate to the Storage Pools screen. The screen should display the ANAME Storage Space created in the previous procedure.



.

2. In the Virtual Disks area, click **Tasks** and from the drop down menu select **New Virtual Disk…** as shown below.



3. Select **Before U Begin** then click **Next**.
4. The ANAME pool should be displayed. Click **Next**.

5. Enter a name (example: BNAME) and select the check box for "Create storage tiers on this virtual disk." Then click **Next**.
6. Select **Mirror** then click **Next**.
7. Select **Fixed** then click **Next**.
8. Select **Maximum size for Tier** and click **Next**. Click **Create**.

Using PowerShell script to enable Storage Spaces

Scripts can be created and executed in PowerShell to automatically perform tasks otherwise requiring a user to perform in the Server Manager GUI.

Creating pools

The following PowerShell script prepares disks and creates storage pools:

```
# To allow unsigned PS script execution, run:
#    cmd.exe /c powershell.exe set-executionpolicy -ExecutionPolicy Bypass -Force

# Clean up virtual disks and storage pools
Get-VirtualDisk | Remove-VirtualDisk -EA SilentlyContinue -Confirm:$False
Get-StoragePool -IsPrimordial $False -EA SilentlyContinue | Remove-StoragePool -Confirm:$False

# Clean up all disks except boot|system disks
$Disks = Get-Disk |? {($_.IsBoot -eq $False) -and ($_.IsSystem -eq $False) -and ($_.BusType -
notmatch "USB") -and ($_.Model -notmatch "Virtual Disk") -and ($_.NumberOfPartitions -ne 0)}
$Disks | Set-Disk -IsReadOnly $False
$Disks | Set-Disk -IsOffline $False
$Disks | Clear-Disk -RemoveData -Confirm:$False

# Create a storage pool with all available physical disks
$AllPd = Get-PhysicalDisk |? {($_.CanPool -eq $True) -and ($_.Model -notmatch "Virtual Disk")}
$NvmPd = $AllPd |? {($_.UniqueId -match "HPENVM*") -or ($_.BusType -match "SCM")}
$HddPd = $AllPd |? {($_.UniqueId -notmatch "HPENVM*") -and ($_.BusType -notmatch "SCM")}
$Pool = New-StoragePool -StorageSubSystemFriendlyName *Storage* -FriendlyName ALLPD-POOL -
PhysicalDisks $AllPd

# Configure NVM disks for Journal, and other disks as HDDs
$NvmPd | Set-PhysicalDisk -Usage Journal
$HddPd | Set-PhysicalDisk -MediaType HDD

# Create a virtual disk with Mirror resiliency, using up all NVM and HDD disks
$VDisk = $Pool | New-VirtualDisk -FriendlyName ALLPD-VDISK -ResiliencySettingName Mirror -
UseMaximumSize -ProvisioningType Fixed
# Create and format partition and volume on the disk
$Disk = $VDisk | Get-Disk
$Disk | Initialize-Disk -PartitionStyle GPT
$Part = $Disk | New-Partition -UseMaximumSize
$Part | Set-Partition -IsOffline $False
$Vol  = $Part | Format-Volume -FileSystem NTFS -NewFileSystemLabel ALLPD-DISK -Force -
Confirm:$False
$Part | Add-PartitionAccessPath –AssignDriveLetter
```

After the above script has run, the Windows Server Manager should display as shown in Figure 4:

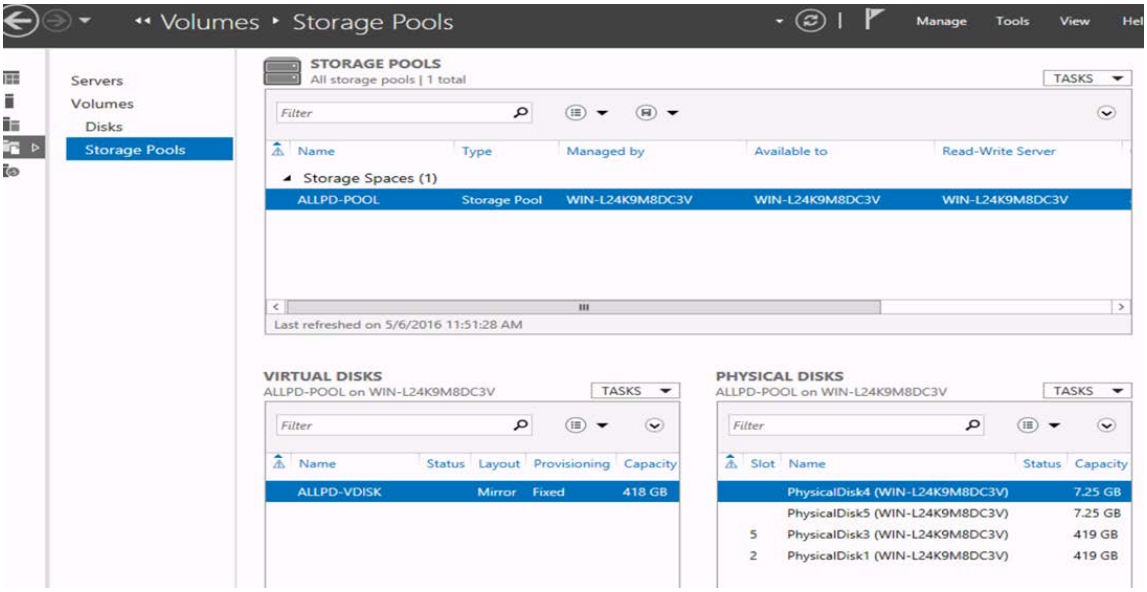


Figure 4. Windows Server Manager screen showing results of PowerShell script.

Removing pools

The following PowerShell script removes storage pools from the configuration.

```
# To allow unsigned PS script execution, run:
#    cmd.exe /c powershell.exe set-executionpolicy -ExecutionPolicy Bypass -Force

# Clean up virtual disks and storage pools
Get-VirtualDisk | Remove-VirtualDisk -EA SilentlyContinue -Confirm:$False
Get-StoragePool -IsPrimordial $False -EA SilentlyContinue | Remove-StoragePool -Confirm:$False

# Clean up all disks except boot|system disks
$Disks = Get-Disk |? {($_.IsBoot -eq $False) -and ($_.IsSystem -eq $False) -and ($_.BusType -
notmatch "USB") -and ($_.Model -notmatch "Virtual Disk") -and ($_.NumberOfPartitions -ne 0)}
$Disks | Set-Disk -IsReadOnly $False
$Disks | Set-Disk -IsOffline $False
$Disks | Clear-Disk -RemoveData -Confirm:$False
```

Performance results with Microsoft® DiskSpd tool

Microsoft DiskSpd is a tool that can simulate many types of workloads in various configurations, including Storage Spaces. Tables 2 through 4 show the results of running DiskSpd on the following system configuration:

System:	HPE ProLiant DL360 G9
Memory:	Two DIMMs and four NVDIMMs
Drive Storage:	Two HDDs, two SSDs
Storage Space Configuration:	Two NVDIMMs with mirroring

Table 2. 4KiB random reads (diskspd.exe -d10 -w0 -r -b4K -t8 -o<qcount> -l #<disk>)

STORAGE TYPE	BYTES	IOS	MB/S	IO/S	LATENCY(US)
1xHDD (qcount=4)	15015936	3666	1	366	87000
1xSSD (qcount=4)	5016391680	1224705	477	122301	261
1xNVM (qcount=1)	109399236608	26708798	10433	2670955	3
2xNVM Mirrored Pool (qcount=1)	60152684544	14685714	5736	1468623	5

Table 3. 4KiB random writes (diskspd.exe -d10 -w100 -r -b4K -t8 -o<qcount> -l #<disk>)

STORAGE TYPE	BYTES	IOS	MB/S	IO/S	LATENCY(US)
1xHDD (qcount=4)	12472320	3045	1	304	105400
1xSSD (qcount=4)	3503214592	855277	333	85430	374
1xNVM (qcount=1)	104729706496	25568776	9987	2556886	3
2xNVM Mirrored Pool (qcount=1)	39056904192	9535377	3724	953571	8

Table 4. 64KiB random reads (diskspd.exe -d10 -w0 -r -b64K -t8 -o<qcount> -l #<disk>)

STORAGE TYPE	BYTES	IOS	MB/S	IO/S	LATENCY(US)
1xHDD (qcount=4)	210108416	3206	20	320	99600
1xSSD (qcount=4)	9778823168	149213	932	14914	2144
1xNVM (qcount=1)	131579576320	2007745	12548	200780	39
2xNVM Mirrored Pool (qcount=1)	236971229184	3615894	22598	361582	22

Best practices: recommended action if one NVDIMM goes offline during mirroring

NVDIMMs are storage devices and support mirroring. As with any device, errors or failures are possible. Should a mirrored NVDIMM go offline due to an error or failure, we recommend taking the following action:

Scenario: Two NVDIMMs (example: disk 4 and disk 5) have been mirrored using the Windows diskmgmt utility. Files have been copied to the mirror drive (example: d: folder).

Error condition: one NVDIMM (example: disk 5) has gone offline due to an error condition ("//ERROR CONDITION" indicated by diskmgmt).

1. Back up files from mirror drive (d folder) to another drive.
2. Open disk part. (i.e., run **diskpart** from the DOS command prompt window)
3. In diskpart, clean the online drive: **run** > select **disk 4** and **clean**
4. In diskmgmt, bring disk 5 online (enters foreign drive). If the NVDIMM does not go online, replace the NVDIMM.
5. Import foreign drive.
6. Repeat steps 2 and 3 to clean drive 5.
7. Re-attempt to mirror drive 4 and drive 5.

Note
If one disk is corrupt during software mirroring, both disks need to be re-set.

Upgrading from Windows 2012 R2 to Windows Server 2016 Technical Preview 5

Microsoft has developed Windows Server 2016 Technical Preview 5 (TP5) with native support of NVDIMMs. To upgrade from Windows Server 2012 R2 to Windows Server 2016 TP5 perform the following:

1. Backup the data on the NVDIMMs
2. Break the interleave set in BIOS/Sanitize
3. Perform a clean installation of Windows Server 2016 TP5.
4. If you need more capacity, then you can create a simple storage spaces solution in block mode or DACs.

Resources, contacts, or additional links

HPE Persistent Memory
hpe.com/servers/persistentmemory

HPE Persistent Memory Whiteboard video
youtube.com/watch?v=BKA_SOPqHfg

HPE Persistent Memory DLON 2015
youtube.com/watch?v=vMrzXOBSeqA

HPE Servers Technical White Papers
hpe.com/docs/servertechnology

Sign up for updates

4AA6-4681ENW, August 2016

Hewlett Packard Enterprise