PAVILION

# Spectrum Scale
# Deployment Guide using Pavilion Hyperparallel Flash Array

Version 2.0

## Summary

The purpose of this document is to guide through the installation and configuration of Spectrum Scale using Pavilion Hyperparallel Flash Array and to describe the general set of configurations that have been validated. The following assumptions are made: The audience are familiar with IBM Spectrum Scale, because this paper is not intended to serve as a comprehensive IBM Spectrum Scale Guide.

✎ These are notes which add some more detail and color to the descriptions.

? A choice needs to be made which will depend on the end user requirements.

! An important piece of information which may impact full functionality.

# Pavilion Hyperparallel Flash Array (HFA)

## Key Recommendations

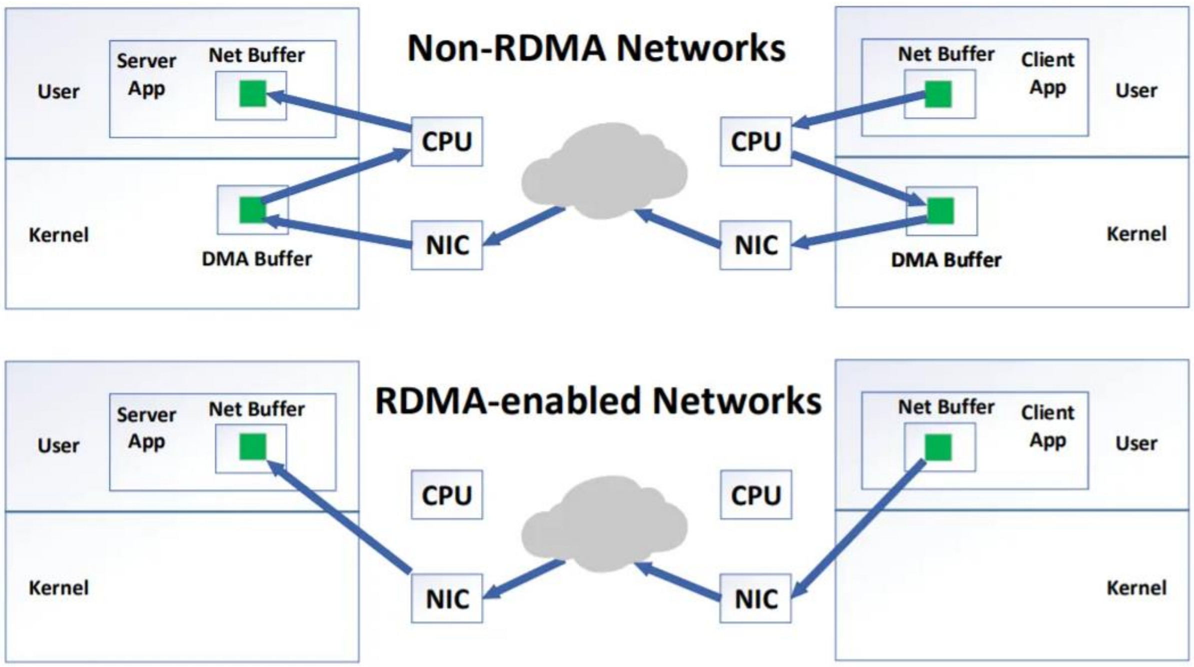
- Infiniband or RDMA capable Ethernet switch
- Ethernet
    - 100, 40, or 25 GbE Ethernet switch (100GbE for best performance), MTU 9216
    - Converged Ethernet NIC (Mellanox ConnectX-4 or newer) with MTU of 9000
- IB
    - EDR, FDR or FDR-10 IB switch (EDR for best performance)
    - Mellanox ConnectX-4 or newer
- RedHat or CentOS Linux 7.6 or newer
- Use Linux in-box driver or Mellanox OFED (3.4.2 or newer)
- Configure Pavilion NVMe-oF controllers to use NVMe-oF RoCE or IB, via GUI or CLI
- Create volumes and assign to controllers on the Pavilion HFA
- Install the “`nvme-cli`” package on clients and use it to attach volumes
- Use volumes on the client as if they were local NVMe devices
- Make the Pavilion volumes persistent across reboots with the PDS RPM
- Build HA volume connections with Linux multipath, Pavilion’s primary and standby controllers, and network infrastructure without any SPOFs

## NVMe and NVMe over Fabrics

Non-Volatile Memory Express (NVMe) is an advanced protocol used to access flash storage on PCI Express bus. “Non-Volatile” stands for persistent storage, while “Express” refers to the fact that the data travels over the PCI Express (PCIe) interface on the computer's motherboard. This gives the drive a direct connection with the CPU and memory and eliminates many latency-inducing layers of traditional storage stacks such as SAS or SATA controllers. All modern servers and operating systems support NVMe out-of-the-box, and for enterprise and cloud scale Solid State Drives (SSDs) it is the interface of choice.

Because NVMe removes the intermediate legacy storage controllers and storage stack from the OS, it can provide significantly reduced latency (on the order of tens of microseconds per 4K I/O on enterprise NVMe SSDs). Thanks to its use of the PCI Express interface, it is also able to support individual drive bandwidths of up to 4GB/s per individual NVMe SSD. NVMe over Fabrics (NVMeOF) is an industry standard extension of the NVMe protocol which allows remote NVMe storage to be attached to servers. This is similar to how legacy Fibre Channel and iSCSI protocols enabled servers to utilize disks in SAN arrays for data storage, but with massively higher bandwidth and an order of magnitude lower latency. Modern Linux operating systems support it out-of-the-box today, with work ongoing for Microsoft Windows and VMWare cloud operating systems.

The trick that enables NVMe-oF, NVMe remote storage, to provide as good or better performance than local direct-attached (DAS) NVMe storage is something called Remote DMA (RDMA). DMA is used in NVMe to allow the SSD to directly load or store data to server memory, without CPU intervention (just like 0-copy accelerated TCP on high-performance NICs). RDMA allows for this same kind of direct memory access, but from outside of the server itself. An RDMA enabled storage array like the Pavilion HFA can handle I/O requests without the server's CPU needing to copy any data whatsoever.



RDMA uses the same link layer but is a separate protocol with its own requirements separate from the more common IP with UDP or TCP on top. It is supported on the two standard networking infrastructures deployed today: InfiniBand and Ethernet.

## Introduction to IBM Spectrum Scale

IBM Spectrum Scale is a clustered parallel filesystem, which is used extensively in the HPC (High Performance Compute) industry. It was also known as IBM General Parallel File System or GPFS before Ver 4.1. In IBM Spectrum Scale, data is divided into blocks and striped across multiple disks of storage. When an IO operation is performed, data is accessed in parallel which allows for faster read and write speeds. IBM Spectrum Scale lets Enterprises share the storage infrastructure, while it intelligently places the data in the optimal storage tier. In addition to providing filesystem storage capabilities, IBM Spectrum Scale provides various tools for management like high availability, replication, mirroring, policy-based automation, disaster recovery and cluster administration. IBM Spectrum Scale supports both Windows & Linux or heterogeneous mix of Linux and Windows clusters.

The Pavilion HFA enables a unique, high performance architecture for IBM Spectrum Scale using NVMe over Fabrics Protocol.  This architecture can reduce hardware requirements, decrease latency, increase bandwidth, and provide a better user experience for high performance storage. This document describes the preferred architecture, how to configure the Pavilion HFA, and Spectrum Scale clients, and finally how to tune Spectrum Scale for optimal performance

## Key Terminology in IBM Spectrum Scale

### Spectrum scale Cluster:

A cluster in IBM Spectrum Scale consists of array of nodes and network shared disks that work together and can be managed as a single system. IBM Spectrum Scale cluster can be configured to use server-based repository type, where the cluster is explicitly configured to have a primary and secondary server to keep cluster configuration files.

### Block & Block size

Most file systems are based on a block device, which is a level of abstraction for the hardware responsible for storing and retrieving specified blocks of data. A block is the largest unit for single I/O operation and space allocation in a IBM Spectrum Scale file system. Usually, block size is specified when a file system is created. Block sizes supported are in the ranges of 16 KB - 16 MB.

### Network Shared Disk (NSD)

IBM Spectrum Scale operates a protocol that implements block-level interface over the network, called a Network Shared Disk (NSD). In a scenario, where all nodes cannot be directly connected to the disks, an NSD with a primary and a secondary server must be defined. I/O is then performed using the network connection to get to the NSD server that performs the I/O on behalf of the NSD clients.

### Cluster Node

A cluster node is a server which has IBM Spectrum Scale software installed which has access to storage either directly or through NSD.

### Cluster manager

The cluster manager node manages the entire cluster and has the responsibility for correctness of operation across the nodes and the cluster as a whole. Some of the responsibilities include monitoring of disk leases, failure detection and recovery management of nodes within the cluster. The cluster manager is chosen through an election process among the set of quorum nodes designated for the cluster, but admin can also explicitly define the nodes that can become the cluster manager node.

## Pavilion HFA architecture for high-performance IBM Spectrum Scale clusters

Traditional high performance, scale-out GPFS installations are based on the NSD Server model where individual I/O servers handle the actual disk traffic to locally attached storage arrays.  This architecture is also similar to "GPFS Appliance" model where the NSD servers and disk trays are prepackaged by a vendor or VAR.

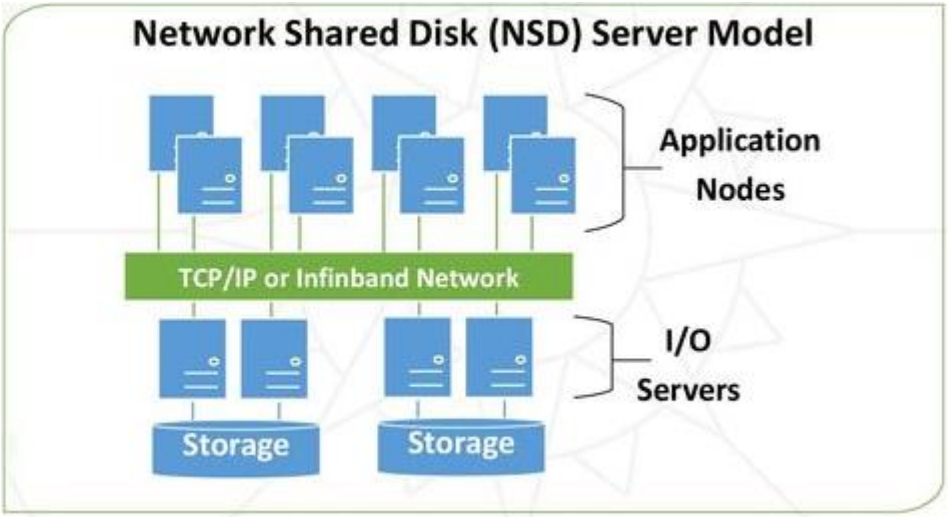

Network Shared Disk (NSD) Server Model
Application Nodes
TCP/IP or Infinband Network
I/O Servers
Storage
Storage

*Figure 1 -Traditional NSD Server Mode, © IBM*

This architecture provides scalability but has serious I/O bottlenecks when used with high performance flash like NVMe SSDs. NSD servers themselves, have limited network bandwidth, CPU horsepower and RAM, and can easily become chokepoints in a large cluster.

Using NVMe-oF protocol and high-performance network, the Pavilion HFA can eliminate these I/O bottlenecks by allowing remote connections to NVMe flash devices over standard RDMA fabrics such as Ethernet of InfiniBand.  Using RoCE or IB, all the application nodes in a cluster can directly connect to the shared, flash storage over the 100G/EDR network, eliminating the need for NSD servers.  NVMe-oF provides dedicated **NVMe queue pairs** to the application nodes, thereby reducing the latency and improving the server utilization.

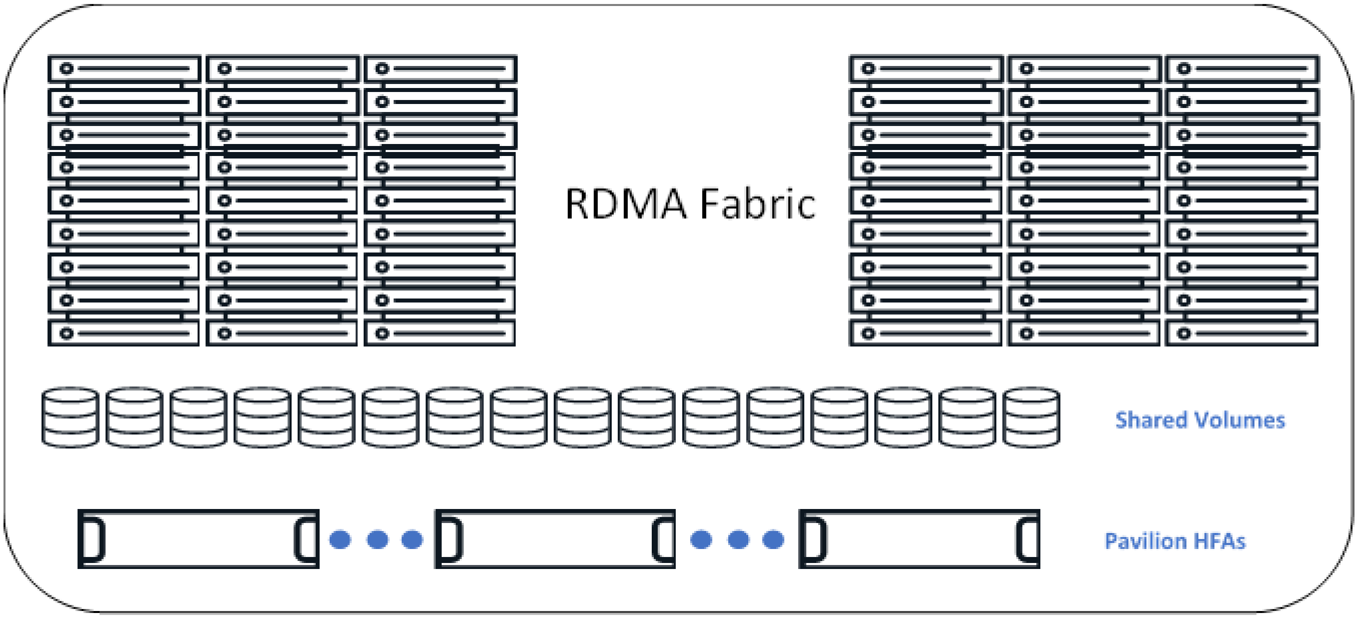


*Figure 2- GPFS with Pavilion HFA*

This architecture has several benefits compared to the older architecture.

- Reduced latency and increased storage bandwidth due to removal of NSD servers from IO path.
- Better balancing of cluster resources since application nodes perform their own I/O operations.
- Reduced cluster hardware due to the removal of NSD servers.
- Linearly scalable I/O performance (greater than 100M IOPS & 1TB/s throughput)
- Highly reliable storage using Pavilion HFA's multipath implementation.

This guide will describe how to create & deploy a cluster using this new architecture.

- Configuration of Pavilion HFA for NVMe-oF
- Host Configuration
- Cluster-wide High Availability, High Performance Architectures
- Deploying Spectrum Scale

# Configuring the Pavilion HFA for NVMe-oF

The Pavilion HFA can be configured with between 2 and 20 controllers. These controllers handle the connection between SSDs in the array (and all data services such as snapshots and clones) and the wider network and servers using that storage. Each one of these controllers can be configured to use a specific protocol, such as NFS, iSCSI, RoCE or IB. All exported volumes (LUNs) from any single controller will use the same protocol.

This section will describe how to configure the controller to use NVMe over RoCE/IB, a process which is often done only once per controller on installation. After configuring it to use the RoCE/IB protocol, standard volume creation and management can be performed without having to reset this setting. As such, these steps are often only run one time, on array installation.

The Pavilion HFA fully supports a CLI, a web-based GUI, and a RESTful interface for management. The following sections will focus on the CLI and GUI interfaces. Please see the separate RESTful interface manual for API-based configuration.

We will demonstrate both CLI and GUI interactions in the following sections. Only one method needs to be used, of course, to configure the volumes and array.  For one-off configurations, the GUI has better ease-of-use, but for larger deployments where automation is required, the CLI or RESTful interfaces are a better choice.

## Configuring a controller to use the RoCE/IB protocol

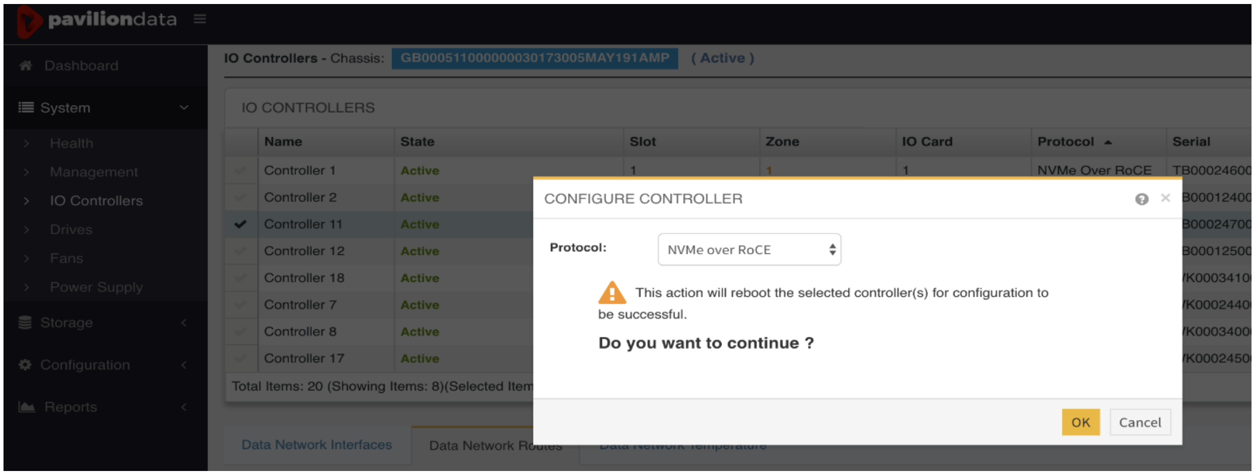
The following snippet shows the sequence of commands used in the CLI to enable the RoCE/IB protocol on a controller. Your controller numbering may be different, please refer to your purchased configuration. Note that if a controller is actively serving volumes, it cannot change its protocol. In that case, you will need to disconnect any volumes prior to configuring.

```
admin@GB0...AMP> switch config
Switched to config namespace

(config) configure controller id 11 protocol roce

OR

(config) configure controller id 11 protocol ib

  Creation of task was successful
  Monitor task : 2d04159f-a094-4e86-93de-cd935e240f93
  Command format : show task id [id]
```

The Pavilion HFA utilizes an asynchronous queue for configuration changes. When a long-running configuration change is requested, a background process ("task") is created to perform the request and control returned to the user immediately.

The "Creation of task was successful" messages indicate that the command is ongoing in the background.  See the CLI User's' Guide for more in-depth

In the GUI, simply log in to the array and click on the System->I/O Controllers side menu. Check the desired controller, and then use the "`Configure Protocol`" button and change the protocol to "`NVMe over RoCE`" or "`NVMe over IB`":



IO Controllers - Chassis: GB000511000000030173005MAY191AMP ( Active )

IO CONTROLLERS

| | Name | State | Slot | Zone | IO Card | Protocol | Serial |
|---|---|---|---|---|---|---|---|
| | Controller 1 | Active | 1 | 1 | 1 | NVMe Over RoCE | TB00024600 |
| | Controller 2 | Active | | | | | B00012400 |
| ✓ | Controller 11 | Active | | | | | B00024700 |
| | Controller 12 | Active | | | | | B00012500 |
| | Controller 18 | Active | | | | | /K00034100 |
| | Controller 7 | Active | | | | | /K00024400 |
| | Controller 8 | Active | | | | | /K00034000 |
| | Controller 17 | Active | | | | | /K00024500 |

Total Items: 20 (Showing Items: 8)(Selected Item

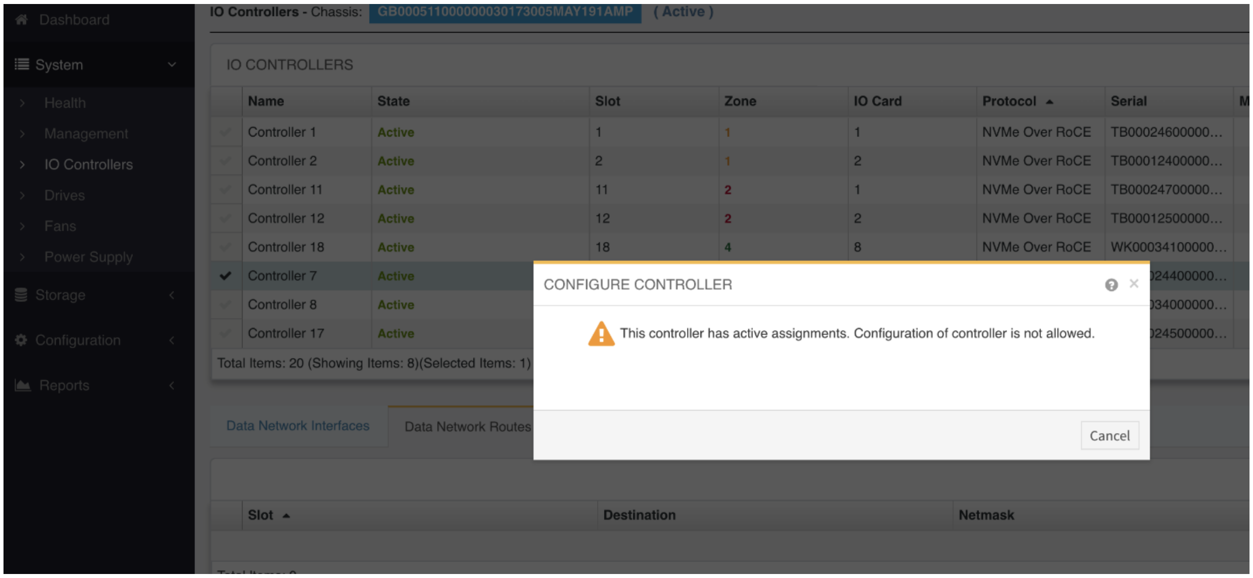
CONFIGURE CONTROLLER

Protocol: NVMe over RoCE

This action will reboot the selected controller(s) for configuration to be successful.

Do you want to continue ?

Press "`OK`" to configure protocol, which will internally reboot the controller. If there are volumes assigned to the controller, the following error message will be displayed. You will need to remove the volume assignments, in this case, to change the protocol.



IO Controllers - Chassis: GB000511000000030173005MAY191AMP ( Active )

IO CONTROLLERS

| | Name | State | Slot | Zone | IO Card | Protocol ▲ | Serial |
|---|---|---|---|---|---|---|---|
| ✓ | Controller 1 | Active | 1 | 1 | 1 | NVMe Over RoCE | TB00024600000… |
| ✓ | Controller 2 | Active | 2 | 1 | 2 | NVMe Over RoCE | TB00012400000… |
| ✓ | Controller 11 | Active | 11 | 2 | 1 | NVMe Over RoCE | TB00024700000… |
| ✓ | Controller 12 | Active | 12 | 2 | 2 | NVMe Over RoCE | TB00012500000… |
| ✓ | Controller 18 | Active | 18 | 4 | 8 | NVMe Over RoCE | WK00034100000… |
| ✓ | Controller 7 | Active | | | | | 24400000… |
| ✓ | Controller 8 | Active | | | | | 34000000… |
| ✓ | Controller 17 | Active | | | | | 24500000… |

Total Items: 20 (Showing Items: 8)(Selected Items: 1)

CONFIGURE CONTROLLER

⚠ This controller has active assignments. Configuration of controller is not allowed.

## Creating and Assigning Volumes on the Pavilion HFA

Once the one-time configuration of controllers is done, volumes may be created and then assigned to RoCE/IB interfaces. This process is similar to standard storage management processes, with several additions. Each volume to be carved from an available “media group” which is equivalent to a typical “drive shelf” or “LUN group.” Media groups are typically configured by the factory during the installation process, but if necessary, more info is available in the full User’s Guide. Controllers have multiple network interfaces whose configuration should also have been performed (typical IP and netmask) on system installation. Again, more detailed interface configuration is available in the User’s Guide.

Volume workflow is a simple two-step process which can be completed from the CLI, the GUI, or the RESTful interface:

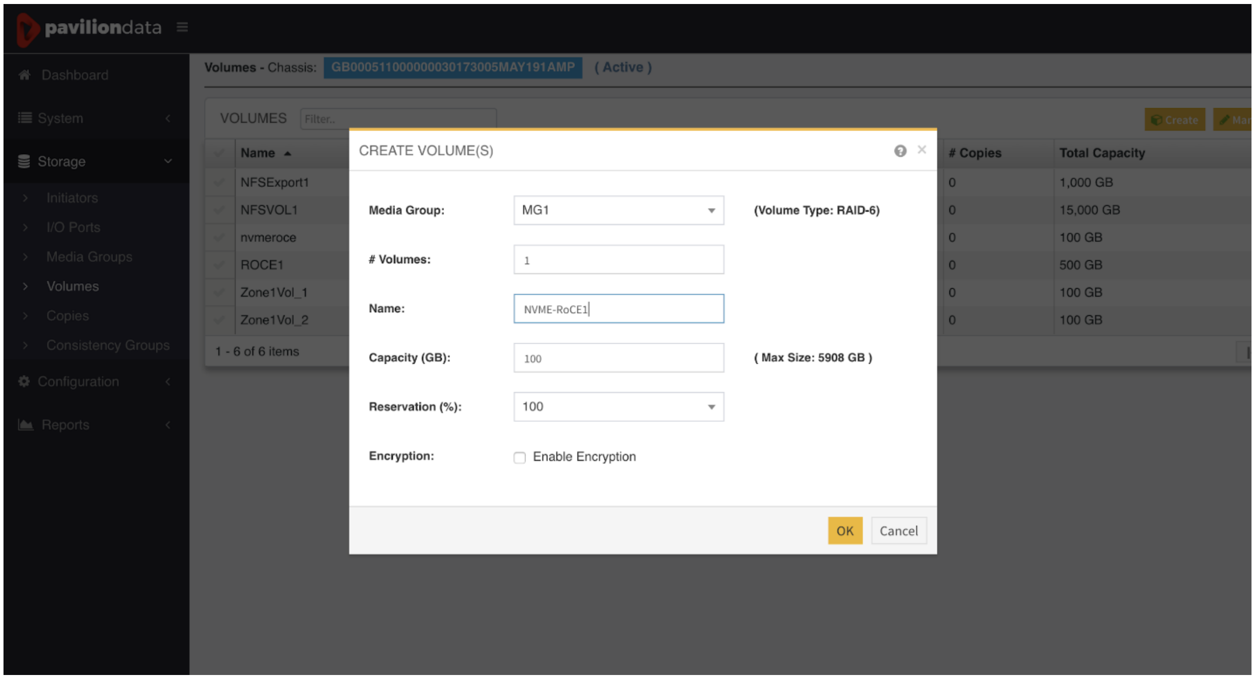
- Create volume of desired size, name, and options
- Assign newly created volume to a controller to enable access by clients

### Creating a volume

Create a volume using the CLI with a single command as shown below.

```
admin@GB0...AMP> switch config
Switched to config namespace

(config) create volume name <VOLNAME> size <SIZE> GB grpname <ZONE>

  Creation of task was successful
  Monitor task : ac63c2a2-a1ad-4930-aa18-510b5859bcf3
  Command format : show task id [id]
```

Using the GUI to create an array is also simple. Use the left-hand menu Storage->Volumes, then the “`Create`” and fill out the form presented.



Volumes - Chassis: GB00051100000030173005MAY191AMP ( Active )

| Name | # Copies | Total Capacity |
|---|---|---|
| NFSExport1 | 0 | 1,000 GB |
| NFSVOL1 | 0 | 15,000 GB |
| nvmeroce | 0 | 100 GB |
| ROCE1 | 0 | 500 GB |
| Zone1Vol_1 | 0 | 100 GB |
| Zone1Vol_2 | 0 | 100 GB |

CREATE VOLUME(S)

Media Group: MG1 (Volume Type: RAID-6)

# Volumes: 1

Name: NVME-RoCE1

Capacity (GB): 100 ( Max Size: 5908 GB )

Reservation (%): 100

Encryption: □ Enable Encryption

OK Cancel

## Assigning volume to a controller and RoCE interface

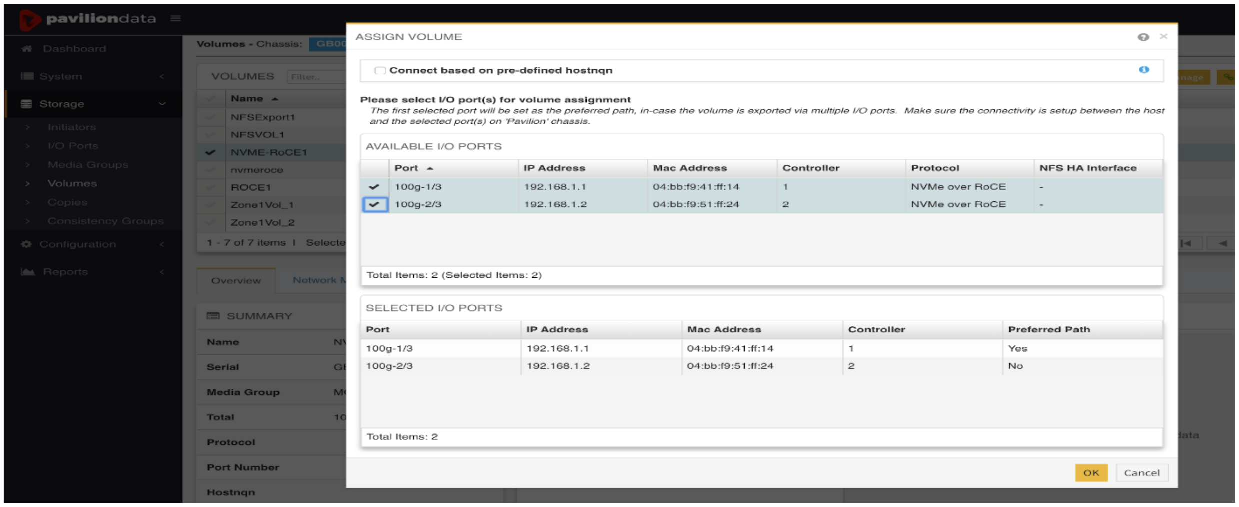
Once a volume is created (a process which may take several seconds while it is initialized), it is available for connecting to a specific controller and interface. Unless this step is performed, the volume will not be accessible to external clients.

To assign a volume via the CLI, the volume name (given previously) and the interface “`port-name`” (of the form `100g-<port>/<zone>` like “`100g-1/3`"). Make a note of the “`Device Serial`” as it will be used by the clients to connect to it later:

```
admin@GB0...AMP> switch config
Switched to config namespace

(config) assign volume name <VOLNAME> port-name <PORTNAME> preferred

  Creation of task was successful
  Monitor task : 7bc94647-118d-46fc-a3ed-467d21efcb64
  Command format : show task id [id]

Block Device: dms12
Device Serial: GB00051104bbf912
```

Make a note of the “Device Serial” as it will be required when connecting clients to this newly generated NVMe-over-RoCE volume. The “Device Name” shown is only for internal array use.

The same process can be done more simply using the GUI. Select the left-hand menu `Storage->Volumes`, then click on the configured volume and press the “`Assign`” button to bring up the assignment dialog. In this case, the “`Device Serial`” will be presented in the updated GUI list after connection:



ASSIGN VOLUME

□ Connect based on pre-defined hostnqn

**Please select I/O port(s) for volume assignment**

*The first selected port will be set as the preferred path, in-case the volume is exported via multiple I/O ports. Make sure the connectivity is setup between the host and the selected port(s) on 'Pavilion' chassis.*

AVAILABLE I/O PORTS

|  | Port | IP Address | Mac Address | Controller | Protocol | NFS HA Interface |
|---|---|---|---|---|---|---|
| ✓ | 100g-1/3 | 192.168.1.1 | 04:bb:f9:41:ff:14 | 1 | NVMe over RoCE | - |
| ✓ | 100g-2/3 | 192.168.1.2 | 04:bb:f9:51:ff:24 | 2 | NVMe over RoCE | - |

Total Items: 2 (Selected Items: 2)

SELECTED I/O PORTS

| Port | IP Address | Mac Address | Controller | Preferred Path |
|---|---|---|---|---|
| 100g-1/3 | 192.168.1.1 | 04:bb:f9:41:ff:14 | 1 | Yes |
| 100g-2/3 | 192.168.1.2 | 04:bb:f9:51:ff:24 | 2 | No |

Total Items: 2

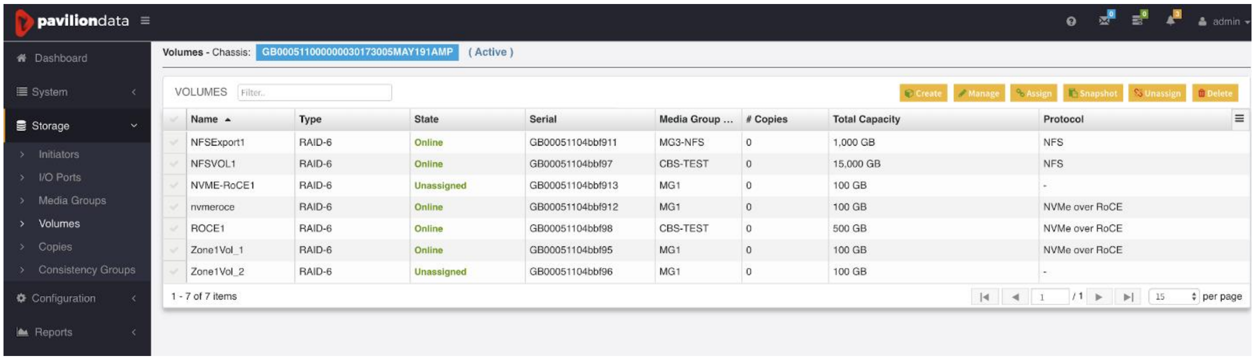
OK Cancel

## Verifying volume assignment

Once configured on the Pavilion HFA, volume status can be checked from both CLI and GUI. The CLI uses the following command (once in the “`show`” namespace):

```
admin@GB0...AMP> switch show
Switched to show namespace

(show) show volume-mapped-network name nvmeroce

Volume Mapped Network list
| Mac Address   | ... | IP Address    | Slot      | Controller | ...
+==============+=...=+==============+==========+============+=...
| 04:bb:f9:...  | ... | 192.168.1.1  | 100g-1/3 |          1 | ...
+--------------+-...-+--------------+----------+------------+-...
```

In the GUI, simply navigate using the left-hand menu to Storage->Volumes:



Volumes - Chassis: GB000511000000030173005MAY191AMP ( Active )

VOLUMES

| Name | Type | State | Serial | Media Group ... | # Copies | Total Capacity | Protocol |
|---|---|---|---|---|---|---|---|
| NFSExport1 | RAID-6 | Online | GB00051104bbf911 | MG3-NFS | 0 | 1,000 GB | NFS |
| NFSVOL1 | RAID-6 | Online | GB00051104bbf97 | CBS-TEST | 0 | 15,000 GB | NFS |
| NVME-RoCE1 | RAID-6 | Unassigned | GB00051104bbf913 | MG1 | 0 | 100 GB | - |
| nvmeroce | RAID-6 | Online | GB00051104bbf912 | MG1 | 0 | 100 GB | NVMe over RoCE |
| ROCE1 | RAID-6 | Online | GB00051104bbf98 | CBS-TEST | 0 | 500 GB | NVMe over RoCE |
| Zone1Vol_1 | RAID-6 | Online | GB00051104bbf95 | MG1 | 0 | 100 GB | NVMe over RoCE |
| Zone1Vol_2 | RAID-6 | Unassigned | GB00051104bbf96 | MG1 | 0 | 100 GB | - |

1 - 7 of 7 items

1 / 1 15 per page

# Host Configuration:

## Host OS Compatibility Matrix:

Below table displays the latest supported matrix for Linux and compatible OFED versions.

| OS Supported | OS Version | Kernel Version | Inbox Driver | OFED Version Driver (optional) |
|---|---|---|---|---|
| CentOS/RHEL | 8 | 4.18.0.32 | Yes | 4.7-1.0.0.1 |
| CentOS/RHEL | 7.6 | 3.10.0.957 | Yes | 4.5-1.0.1.0 |
| CentOS/RHEL | 7.5 | 3.10.0.862 | Yes | 4.4-1.0.0.0 |
| CentOS/RHEL | 7.4 | 3.10.0.693 | Yes | 4.1-1.0.2.0 |
| Ubuntu | 18.04 | 4.15.0-43-generic | Yes | 4.4-1.0.0.0 |
| Ubuntu | 17.04 | 4.15.0-43-generic | Yes | 4.0-2.0.2.0 |

Inbox RDMA: From Centos 7.4 onwards, the operating system supports Inbox-RDMA driver. There is a standard NVMe-oF driver in the specified compatible OFED versions, that can be used as well. Multipath support is provided using standard DM based multipath daemon, that works on top of both the drivers.

## Supported Client Network Adapters

| Vendor | Model Family | Transport Type | Transport Protocol | Part # Family |
|---|---|---|---|---|
| Mellanox | ConnectX 3 | IB, Ethernet | RDMA, TCP, NFS, iSCSI | MCX3* |
| Mellanox | ConnectX 4 | IB, Ethernet | RDMA, TCP, NFS, iSCSI | MCX4* |
| Mellanox | ConnectX 5 | IB, Ethernet | RDMA, TCP, NFS, iSCSI | MCX5* |
| Mellanox | ConnectX 6 | IB, Ethernet | RDMA, TCP, NFS, iSCSI | MCX6* |
| Chelsio | T6 | Ethernet | TCP, NFS, iSCSI | T62* |
| Intel | XXV710 | Ethernet | TCP, NFS, iSCSI | XXV710* |

| | | | | |
|---|---|---|---|---|
| Broadcom | N250G | Ethernet | TCP, NFS, iSCSI | BCM957* |
| Broadcom | P2100G/N2100G | Ethernet | TCP, NFS, iSCSI | BDM957* |
| Broadcom | P1100/N1100/M1100 | Ethernet | TCP, NFS, iSCSI | BDM957* |
| Broadcom | P150/N150/M150 | Ethernet | TCP, NFS, iSCSI | BDM957* |
| Broadcom | P425/N425 | Ethernet | TCP, NFS, iSCSI | BDM957* |

## RDMA Initiator Configuration (In-Box Drivers)

Volumes on the Pavilion HFA can be accessed by any RedHat or CentOS version 7.0 or later client. The user has the option of using in-box drivers (drivers that ship with the operating system) or using Mellanox's OFED (OpenFabrics Enterprise Distribution) drivers.

In-box drivers normally are installed automatically when the operating system is installed. However, there are often utilities and other additional files that a user may need to fully utilize the Pavilion HFA. This section will describe the process used to verify the driver configuration, ensure needed utilities are installed, and connect to volumes shared by the Pavilion HFA.

First, log in as root (or perform "sudo bash") to simplify verification. NVMe connection is a privileged process and normal users are not allowed to perform these kinds of operations (similar to the restrictions placed on standard storage management tools).

To check that the host has NVMe drivers installed by the operating system and to verify their versions, use the commands:

```
# modinfo nvme_rdma
filename:    /lib/modules/.../kernel/drivers/nvme/host/nvme-rdma.ko.xz
license:     GPL v2

# cat /sys/module/nvme_core/version
1.0
```

The tool "`nvme`" (the standard NVMe CLI) is used to connect to the Pavilion HFA over RoCE and attach and detach volumes to the client. Check that it is installed and that the version is 1.8 or later:

```
# nvme version
nvme version 1.8.1
```

If this command fails, then the host needs to install the package from the Yum repo as follows:

```
# nvme
-bash: nvme: command not found

# yum install -y nvme-cli
Installed size: 519 k
Downloading packages:
nvme-cli-1.8.1-3.el7.x86_64.rpm       | 282 kB  00:00:00
Running transaction
```

```
  Installing : nvme-cli-1.8.1-3.el7.x86_64   1/1
  Verifying  : nvme-cli-1.8.1-3.el7.x86_64   1/1

Installed:
  nvme-cli.x86_64 0:1.8.1-3.el7
```

Verify the ethernet cards (Mellanox ConnectX-4 or newer recommended) are recognized and operating with an MTU of 9000 (for best performance on 100G networks):

```
# lspci | grep Mellanox
02:00.0 Ethernet cntr: Mellanox Technologies MT27700 Family [ConnectX-4]
02:00.1 Ethernet cntr: Mellanox Technologies MT27700 Family [ConnectX-4]

# ethtool eth2 | grep -e detected -e Speed
      Speed: 100000Mb/s
      Link detected: yes

# ip a | grep eth2 | grep mtu
eth2: <BROADCAST,UP,LOWER_UP> mtu 9000 state UP group default qlen 1000
```

If the network is not configured with the proper MTU or other settings, use the standard operating system file `/etc/sysconfig/network-scripts/ifcfg-ethX` to change it. Adding "`MTU=9000`" at the end and restarting the network will ensure the higher MTU.

```
# vi /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth2
UUID=7caf0f5b-333e-4af5-8e91-a6a917998d7a
# Static IP Address #
BOOTPROTO=none
IPADDR=192.168.1.190
NETMASK=255.255.255.0
ONBOOT=yes
# Jumbo Frames
MTU=9000

# systemctl restart network.service
```

Ping the Pavilion controller data port IP address (the one that volumes were connected to in prior stages, not the management interface). Add "`-s 9000`" to verify jumbo frames are enabled.

```
# ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.081 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.032 ms

# ping 192.168.1.2 -s 8972 -M do
PING 192.168.1.2 (192.168.1.2) 8972(9000) bytes of data.
8980 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.091 ms
8980 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.073 ms
```

If the "ping -s 8972" command fails, but the plain "ping" command without the 8972 option succeeds, this is often the fault of a switch that's not configured to support MTU 9000 (jumbo frames). Please consult your switch’s operating guide to check the MTU on connected ports, and correct if necessary.

If the large pings fail, then a network switch port change is required. More details on configuring specific, tested switches are available in the “Networking Best Practices Guide.”

Once network connectivity is confirmed, the kernel should be queried to determine if the NVMe-oF modules are loaded. This is because even though the operating system installed the NVMEoF support files on initial install, the kernel will only load them if NVMe-oF connections are used.

```
# lsmod | grep nvme_rdma
#                  <---modules are not loaded
```

If the preceding command doesn’t list the “nvme_rdma” module, it will need to be loaded via the following commands to allow the NVMe CLI to make connections to volumes on the Pavilion HFA.

```
# modprobe -v nvme_rdma
Insmod /lib/modules/.../kernel/drivers/nvme/host/nvme-core.ko.xz
insmod /lib/modules/.../kernel/drivers/nvme/host/nvme-fabrics.ko.xz
insmod /lib/modules/.../kernel/drivers/infiniband/core/ib_core.ko.xz
insmod /lib/modules/.../kernel/drivers/infiniband/core/ib_cm.ko.xz
insmod /lib/modules/.../kernel/drivers/infiniband/core/iw_cm.ko.xz
insmod /lib/modules/.../kernel/drivers/infiniband/core/rdma_cm.ko.xz
insmod /lib/modules/.../kernel/drivers/nvme/host/nvme-rdma.ko.xz

# modprobe -v mlx4_ib
insmod /lib/modules/.../net/ethernet/mellanox/mlx4/mlx4_core.ko.xz
insmod /lib/modules/.../kernel/drivers/infiniband/hw/mlx4/mlx4_ib.ko.xz

# modprobe -v mlx5_ib
insmod /lib/modules/.../kernel/drivers/infiniband/hw/mlx5/mlx5_ib.ko.xz
```

After this check again if all modules are loaded correctly,

```
# lsmod | grep nvme
nvme_rdma      28408   0
rdma_cm        59673   1 nvme_rdma
ib_core       242235   6 rdma_cm,ib_cm,iw_cm,mlx4_ib,mlx5_ib,nvme_rdma
nvme_fabrics   19997   1 nvme_rdma
nvme_core      58852   2 nvme_fabrics,nvme_rdma
```

At this point, the host should be ready to query available volumes from the Pavilion HFA, and to connect to them and use the for local storage. Skip ahead to the “`Connecting Volumes to Clients`” section to continue attaching volumes to configured clients.

If Server is running with ConnectX-3, ConnectX-3 Pro, then mlx4_core, mlx4_en, mlx4_ib modules need to be loaded. If server is running with ConnectX-4, ConnectX-4 Pro then mlx5_core, mlx5_ib modules need to be loaded.

## RDMA Initiator Configuration (Mellanox OFED Drivers)

In the vast majority of deployments, the in-box drivers and utilities described previously will suffice for client configuration. However, in certain cases when users need additional functionality for other applications (say, for doing RDMA to a remote compute accelerator) the OFED driver stack may be used by administrators. This section will go over the installation of the OFED stack from Mellanox. Once the OFED drivers are installed, management for accessing Pavilion volumes is identical as the prior section.

There is no difference in performance expected between working inbox NVMe-over-RoCE drivers or the Mellanox OFED distributed drivers.

The OpenFabrics Alliance promotes remote direct memory access (RDMA) switched fabric technologies for server and storage connectivity. These high-speed data-transport technologies are used in high-performance computing facilities, in research and various industries.

Mellanox OFED (OpenFabrics Enterprise Distribution) Drivers, `MLNX_OFED`, is a Mellanox tested and packaged version of OFED and supports two interconnect types using the same RDMA (remote DMA) and kernel bypass APIs called OFED verbs – InfiniBand and Ethernet. Up to 100Gb/s InfiniBand and RoCE (based on RDMA over Converged Ethernet standard) over 10/25/40/50/100GbE are supported with OFED by Mellanox to enable OEMs and System Integrators to meet the needs of end users in the said markets.

The Mellanox OFED drivers can be downloaded from Mellanox website at https://www.mellanox.com/page/products_dyn?product_family=26. As of this writing version 4.6-1.0.1.1 is available from the direct link http://content.mellanox.com/ofed/MLNX_OFED-4.6-1.0.1.1/MLNX_OFED_LINUX-4.6-1.0.1.1-rhel7.6-x86_64.tgz.

Download the tarball using wget on the client system:

```
# wget \
http://content.mellanox.com/ofed/MLNX_OFED-4.6-1.0.1.1/MLNX_OFED_LINUX-
4.6-1.0.1.1-rhel7.6-x86_64.tgz

--2019-10-08 22:56:02--
Resolving content.mellanox.com (content.mellanox.com)... 107.178.241.102
```

```
Connecting                    to                    content.mellanox.com
(content.mellanox.com)|107.178.241.102|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 252193073 (241M) [application/x-tar]
Saving to: 'MLNX_OFED_LINUX-4.6-1.0.1.1-rhel7.6-x86_64.tgz'

100%[=======================>] 252,193,073 60.4MB/s   in 4.5s

2019-10-08 22:56:07 (54.0 MB/s) - 'MLNX_OFED_LINUX-4.6-1.0.1.1-rhel7.6-
x86_64.tgz' saved [252193073/252193073]
```

Unzip and untar the file, extract the contents into a directory and cd into the directory. Run the "`mlnxofedinstall`" script with the "`--with-nvmf`" option specified. NOTE: It is very important to have the "`--with-nvmef`" option on the installation or else the installed driver package will be unable to connect to Pavilion volumes over NVMe over RoCE. You may also be prompted to install additional CentOS support packages before installation can proceed. Do so and re-try in case of warning, as shown below:

```
# tar -xf MLNX_OFED_LINUX-3.4-2.0.0.0-rhel7.2-x86_64.tgz
# cd MLNX_OFED_LINUX-3.4-2.0.0.0-rhel7.2-x86_64/
# ./mlnxofedinstall --with-nvmf
Logs dir: /tmp/MLNX_OFED_LINUX.798.logs
General log file: /tmp/MLNX_OFED_LINUX.798.logs/general.log
Verifying KMP rpms compatibility with target kernel...
Error: One or more required packages for installing MLNX_OFED_LINUX are
missing.
Please install the missing packages using your Linux distribution Package
Management tool.
Run:
yum install gtk2 atk cairo tcl tk

# yum install -y gtk2 atk cairo tcl tk

# ./mlnxofedinstall --with-nvmf
Logs dir: /tmp/MLNX_OFED_LINUX.2515.logs
General log file: /tmp/MLNX_OFED_LINUX.2515.logs/general.log
Verifying KMP rpms compatibility with target kernel...
This program will install the MLNX_OFED_LINUX package on your machine.
Note that all other Mellanox, OEM, OFED, RDMA or Distribution IB packages
will be removed.
Those packages are removed due to conflicts with MLNX_OFED_LINUX, do not
reinstall them.

Do you want to continue?[y/N]:y
```

After the install, restart the "`openibd`" services to load the new driver and enable connections. Use the built-in "`ofed_info`" to verify the installation options afterwards:

```
To load the new driver, run:
/etc/init.d/openibd restart
Note: In order to load the new nvme-rdma and nvmet-rdma modules, the nvme
module must be reloaded.
```

```
# /etc/init.d/openibd restart
Unloading HCA driver:                                  [  OK  ]
Loading HCA driver and Access Layer:                   [  OK  ]

# ofed_info | more
MLNX_OFED_LINUX-4.6-1.0.1.1 (OFED-4.6-1.0.1):
```

At this point, the client setup process is identical to using in-box drivers. See the section "`Setting up the client for NVMe-oF`" above for detailed instructions.

## Connecting Volumes to Clients

Once a host has been configured with the proper drivers, either in-box or Mellanox OFED, then it may start connecting and using volumes shared by the Pavilion HFA. This section describes how to enumerate available volumes from a client, connect one or many of them, and use them as any other drive. It also describes the additional utility and configuration required to allow clients to reconnect to volumes automatically on reboot.

### Querying available volumes from the Pavilion HFA

The NVMe CLI can list the available volumes on any Pavilion HFA interface at this point, similar to the ISCSI "`iscsiadm -m discovery -t sendtargets`" command.

```
# nvme discover -t rdma -a <interface-ip>
Discovery Log Number of Records 1, Generation counter 1
=====Discovery Log Entry 0======
trtype:  rdma
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  20
trsvcid: 4420
subnqn:  GB00051104bbf912
traddr:  192.168.1.1
rdma_prtype: unrecognized
rdma_qptype: unrecognized
rdma_cms:    unrecognized
rdma_pkey: 0x0000
```

Only volumes assigned to the specific interface IP will be visible, of course. The user assigned names are, unfortunately, not available using this command, but the "`subnqn`" field maps directly to the "`Device Serial`" presented in the GUI and CLI sections on assigning volumes.

### Connecting a volume to a client

Attaching a volume to a client allows it to access the raw device in the same way as any local SSD storage. Use the "`nvme connect`" CLI command and the "`Device Serial`" or "`subnqn`" to select the specific device from the interface.

```
# nvme connect -t rdma -a 192.168.1.1 -n GB00051104bbf912

# nvme list
Node          SN           Model      Usage             ...
------------- ----...----- -------... ----------------- ...
/dev/nvme0n1  GB00...f912  PVL-MX0... 107.37GB/107.37GB ...
```

If controller is hosting multiple volumes and the host needs to connect to all volumes, "`nvme connect-all`" can be used as a quick way of connecting all volumes to the specific host. While this is not often seen in deployments, it can speed initial testing.

```
# nvme connect-all -t rdma -a 192.168.1.1

# nvme list
Node          SN           Model      Usage             ...
------------- ----...----- -------... ----------------- ...
/dev/nvme0n1  GB00...f912  PVL-MX0... 107.37GB/107.37GB ...
/dev/nvme1n1  GB00...f913  PVL-MX0... 107.37GB/107.37GB ...
/dev/nvme2n1  GB00...f914  PVL-MX0... 107.37GB/107.37GB ...
```

### Working with NVME-oF volumes on a client

After connecting the appropriate volumes to the client, you can use "`lsblk`" to show information about them:

```
# lsblk
NAME    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda       8:0    0 465.8G  0 disk
├─sda1    8:1    0   512M  0 part /boot
├─sda2    8:2    0  31.4G  0 part [SWAP]
└─sda3    8:3    0 433.8G  0 part /
nvme0n1 259:0    0   100G  0 disk
```

Note that the device nodes are connected to the host in the form of "`/dev/nvmeNNn1`", where NN increases from 0 to the number of devices less one. These device nodes are capable of being used exactly as if they were direct attached storage at this point.

You may create and mount any local filesystem on the attached `/dev/nvmeXXn1` device as if it were local. For the recommended XFS or EXT4 filesystems, the default options are sufficient for most users. If you have special needs of the filesystem (i.e. large number of inodes, no reserved space, or others) then add them to the "`mkfs`" command as usual. Note that the device node reports itself to Linux as an SSD, so the I/O scheduler will be "noop" as is best for performance automatically. The device node also has a 4K sector size, meaning that file systems should be configured with at least a 4K allocation unit (4K is the default).

```
# mkfs.ext4 /dev/nvme0n1
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
```

```
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
6553600 inodes, 26214400 blocks
1310720 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2174746624
800 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632,
      2654208, 4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

# mount /dev/nvme0n1 test -orelatime

# mount   # Verify mount succeeded
/dev/nvme0n1 on /root/test type ext4 (rw,relatime,data=ordered)
```

At this point the volume will not be persistent across reboots, nor will it be available in the case of a network link failure.  Please see the following sections for details on ensuring the NVMe-over-RoCE volumes are automatically connected and their file systems mounted on every reboot.

The NVMe-oF volume can be removed from the client only when all mounts and open references to it are closed, as is required for all normal devices. Unmount any filesystems and then use the “`nvme disconnect`” command to drop the connection to the Pavilion HFA.

```
# umount test

# nvme disconnect -n GB00051104bbf912
NQN:GB00051104bbf912 disconnected 1 controller(s)

# nvme list
#                <- empty list, no connections
```

## Making Pavilion volumes persistent across reboots

Once a client has been configured manually to mount a filesystem on a NVMe over RoCE volume served by the Pavilion HFA, it makes sense to make the NVMe connections persistent across reboots (so that services which automatically start can use the volume for data storage). Pavilion supplies a utility called the “`PDS NVMe-oF Service`” which allows for just such an automatic configuration. This section describes installing and configuring the utility.

To install the service, simply install the RPM supplied by Pavilion Data:

```
# rpm -i PDS_NVMEoF_SERVICE-1.6-1.el7.x86_64.rpm
Created symlink from /etc/systemd/system/multi-
user.target.wants/pds.service to /etc/systemd/system/pds.service.
Created symlink from /etc/systemd/system/local-
fs.target.wants/pds.service to /etc/systemd/system/pds.service.
no crontab for root
--------------------------------
Please update the pds.conf file in /etc/pds as per the chassis setup
configuration.
--------------------------------
run command 'systemctl start pds.service' to start the service after
editing pds.conf
--------------------------------
```

Once installed it will automatically be invoked when the server is restarted. However, it needs to be configured to properly identify the NVMe-oF devices to connect on startup. Do this by editing the file "/etc/pds/pds.conf" as shown below.

```
Version=1

# cntrlr_ip_addr_list contains controllers ip addresses and
# all the volumes under these controllers will be connected.
cntrlr_ip_addr_list=()

# nqns_list contains specific nqn's to be connected.
# All the nqn's would be checked against the controller's
# IP address in nqn_cntrlr_ip_list, and then connect.
nqns_list=()
nqns_cntrlr_ip_list=()

# Defaults for the NVMe connection, no need to change
num_io_queue=8
```

If the client needs to connect to all exposed volumes on a specific Pavilion interface (i.e. you've used "nvme  connect-all" to configure it above), simply add the interface IP or IPs (separated by a space) to the variable, "cntrlr_ip_addr_list.". For example:

```
cntrlr_ip_addr_list=(192.168.1.110)
```

In the more common case, where only a specific subset of volumes from a particular interface will be connected to the client on boot, simply use the "nqns_list" and to list the "Device Serials" or "subnqns" that should be connected (separated by spaces), and the "nqns_cntrlr_ip_list" to specify the IP addresses of interfaces serving them. This is an exhaustive search, and every nqn_cntrlr_ip will be queried to see if they can provide a connection to every "nqn_list' element. This allows for high availability as well as normal, single path connections to volumes. For example:

```
nqns_list=(GB0000001 GB0000002)
nqns_cntrlr_ip_list=(192.168.1.110 192.168.1.111)
```

Then make sure to add the appropriate mount options in /etc/fstab. Because device names may change between reboots, it is recommended to use the "LABEL=xxxx" option in any mount command lines, to ensure that the proper volume is mounted in the proper place, no matter the ordering of the device nodes provided by Linux.

```
      LABEL=mongodb1    /var/mongodb/data01     ext4   noatime 0         0
```

# Cluster-wide High Availability, High Performance Architectures

## High Availability Features

The Pavilion storage array is designed from the ground up with key high availability features to application uptime in cloud-scale environments. Specific features are listed below.

### No Single Point of Failure

Every component is at least dual redundant, including network ports, SSDs, internal PCIe fabric, I/O controllers, supervisor modules, power supplies and fans.

### Up to 20 Independent, Redundant I/O Controllers

All I/O controllers are active and serve I/O operations simultaneously, providing linear performance improvement as you add controllers. Each volume is available through multiple controllers, providing full availability even in the event of controller failure via multi-path I/O.

### Hot-Swappable Components

All components in the chassis are hot swappable for maximum serviceability, including SSDs, I/O line cards, supervisor modules, PCIe fabrics, fans and power supplies.

### Dual-Parity RAID with Hot Spare Support

By default, all user data volumes are provisioned from a drive group containing up to 18 NVMe SSDs in a RAID-6 (16+2) configuration. This ensures that up to two drives can fail without interrupting application access to data. The entire system contains up to 4 zones of media, each with its own independent RAID group. Hot spares are also supported (15+2+1), enabling the array to rebuild to full redundancy, allowing more time to schedule drive replacement without increased data risk.

### Redundant Supervisor Modules

All components are managed by redundant out-of-band management controllers, or supervisors. Management of the array is done independently of the I/O controllers and the data paths, providing greater flexibility and consistent performance even during maintenance operations.

### Redundant Internal PCIe Fabric

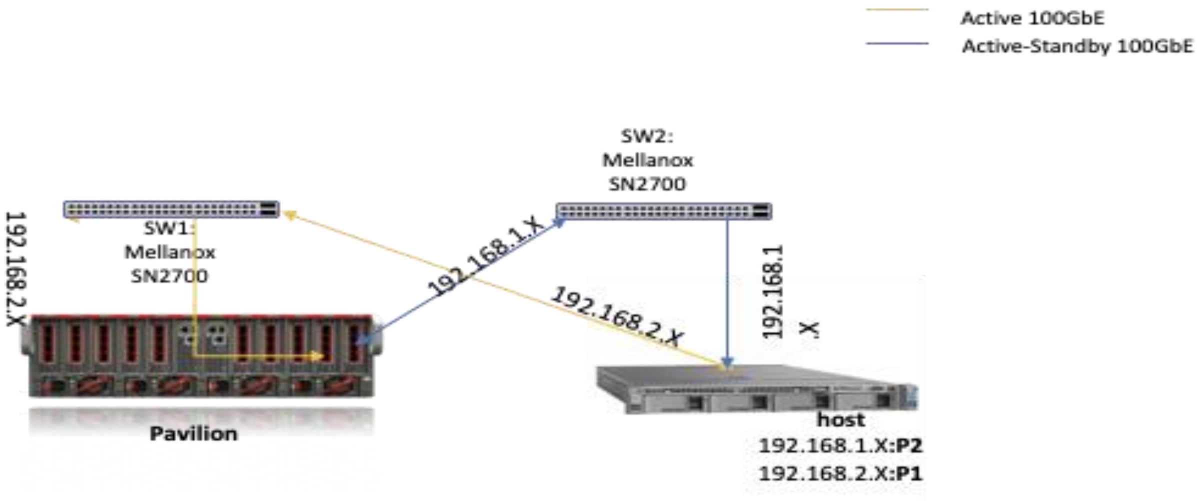
Pavilion's patented architecture employs a high-speed PCIe network to connect all of the internal components, including I/O Line Cards, the NVMe Drive Array, and the Supervisor Modules. This fabric is fully-redundant and is implemented on dual-redundant swappable PCIe switch cards contained in the chassis.

## Multipath Support

Many of the same concepts and techniques used to provide high availability storage connectivity for ISCSI and Fibre Channel configurations can also be used by NVMe over RoCE and the Pavilion HFA.

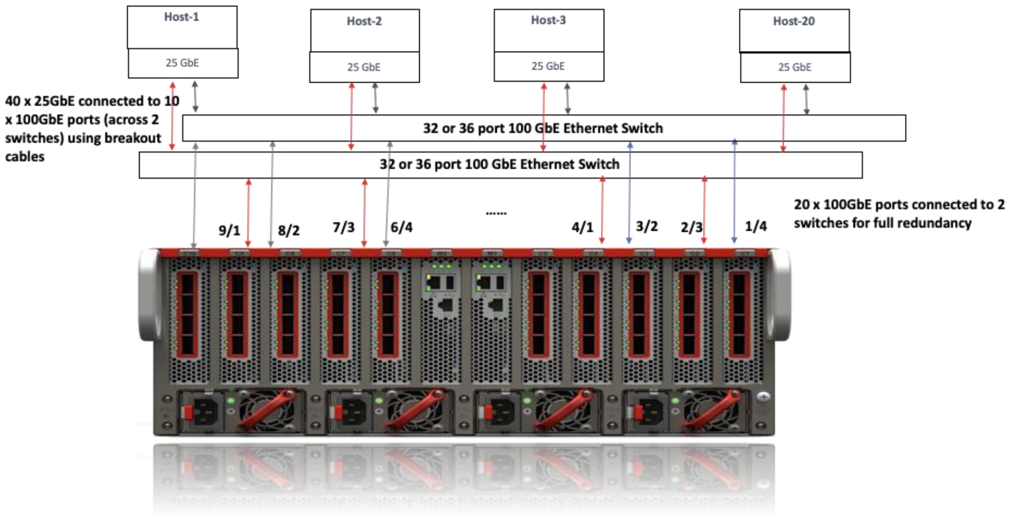
This section describes the settings and network topology required to provide HA client access to volumes on the Pavilion HFA. An understanding of networking and high availability concepts is assumed, and past experience with Linux multipathing is useful. Multiple NICs, switches, and cabling are required to support this multipathing, the same as with any other storage multipath technology.

A path is a connection between a server and the underlying storage. The path can be severed due to many reasons such as a faulty ethernet switch port, a cable disconnection, power supply issues, or even NIC port failures. When only one path exists between the client and the Pavilion HFA, a single failure anywhere along it will sever connectivity until the path is restored. To guard against this, for mission critical systems a multipath network configuration is recommended. Multiple NICs in the client are connected to separate switches and those switches are connected to different controllers on the Pavilion HFA. This "multipathing" ensures that the client can stay connected even when a single failure occurs.



## High Availability for the cluster using redundant switches

For high availability cluster design, you may approach the Pavilion HFA as you would any other SAN-type system. All the usual concerns about removing cluster-wide single points of failure (SPOFs), such as redundant, separately served power distribution units, dual switches, careful cabling, etc. apply. However, because the Pavilion HFA can provide significantly higher throughput than other AFAs, careful consideration should be applied to network and client topology to ensure maximum throughput. Below is a sample HA architecture with a 25G Ethernet infrastructure:


Host-1
25 GbE
Host-2
25 GbE
Host-3
25 GbE
Host-20
25 GbE
40 x 25GbE connected to 10 x 100GbE ports (across 2 switches) using breakout cables
32 or 36 port 100 GbE Ethernet Switch
32 or 36 port 100 GbE Ethernet Switch
9/1
8/2
7/3
6/4
......
4/1
3/2
2/3
1/4
20 x 100GbE ports connected to 2 switches for full redundancy

For more detail, please see the "Pavilion HFA Networking Best Practices and Implementation Guide" which contains a more detailed overview of general HA network topologies and configurations. The "Rack-Scale" configuration therein described is implemented here.

The Pavilion HFA has 20 IO controllers in a full configuration. Each of these controllers can provide approximately 5 GB/s of throughput, and this number should be used as a starting point for architectural designs that are performance critical.

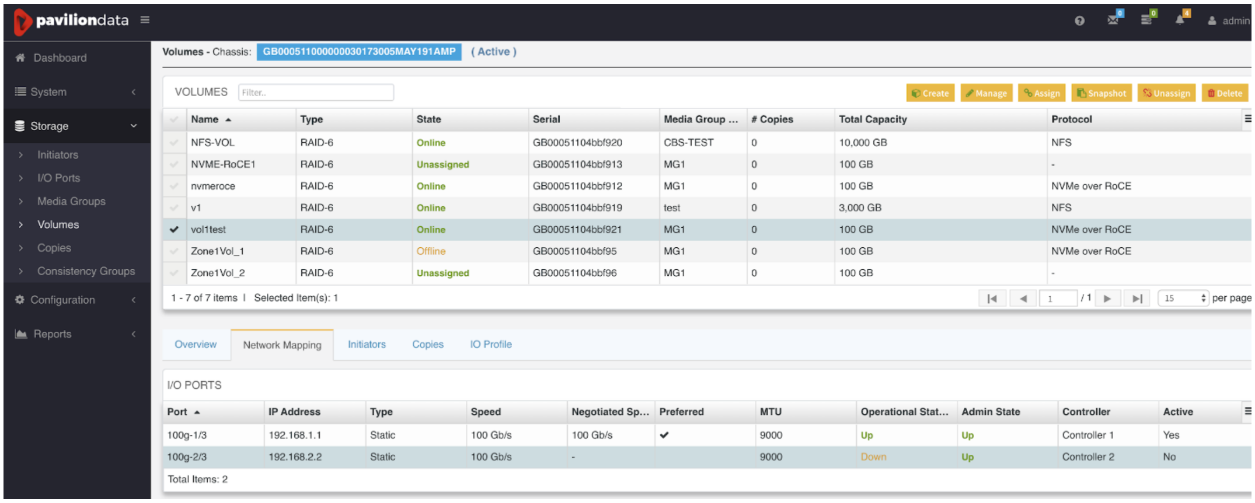
### Multi-Path I/O Support

Pavilion's multi-path I/O support (dual paths) allows for uninterrupted data availability even under full line card, storage controller, NIC, or cabling failure. The use of industry standard multi-path NVMe-oF allows the operating system to transparently route around the failure to get to data, with the application needing no changes whatsoever.

## Multipath configuration for volumes on the Pavilion HFA

First, create a volume as described earlier in this document, using the CLI, the web GUI, or the RESTful interface.

At the next step, where you assign the volume to a controller interface port, instead of selecting a single interface (100g-XXX/Y), select two of them (only two-way multipath is supported). Only controllers in the same zone can be configured as an HA pair. These two interfaces (separate IPs on separate NICs) will form an active/standby pair. After clicking "OK" to finish the connection, use the "Network Mapping" tab at the bottom of the GUI to identify the active one as "Preferred."



Volumes - Chassis: GB00051100000000301730005MAY191AMP ( Active )

VOLUMES

| | Name | Type | State | Serial | Media Group ... | # Copies | Total Capacity | Protocol |
|---|---|---|---|---|---|---|---|---|
| | NFS-VOL | RAID-6 | Online | GB00051104bbf920 | CBS-TEST | 0 | 10,000 GB | NFS |
| | NVME-RoCE1 | RAID-6 | Unassigned | GB00051104bbf913 | MG1 | 0 | 100 GB | - |
| | nvmeroce | RAID-6 | Online | GB00051104bbf912 | MG1 | 0 | 100 GB | NVMe over RoCE |
| | v1 | RAID-6 | Online | GB00051104bbf919 | test | 0 | 3,000 GB | NFS |
| ✓ | vol1test | RAID-6 | Online | GB00051104bbf921 | MG1 | 0 | 100 GB | NVMe over RoCE |
| | Zone1Vol_1 | RAID-6 | Offline | GB00051104bbf95 | MG1 | 0 | 100 GB | NVMe over RoCE |
| | Zone1Vol_2 | RAID-6 | Unassigned | GB00051104bbf96 | MG1 | 0 | 100 GB | - |

1 - 7 of 7 items | Selected Item(s): 1 1 / 1 15 per page

I/O PORTS

| Port | IP Address | Type | Speed | Negotiated Sp... | Preferred | MTU | Operational Stat... | Admin State | Controller | Active |
|---|---|---|---|---|---|---|---|---|---|---|
| 100g-1/3 | 192.168.1.1 | Static | 100 Gb/s | 100 Gb/s | ✓ | 9000 | Up | Up | Controller 1 | Yes |
| 100g-2/3 | 192.168.2.2 | Static | 100 Gb/s | - | | 9000 | Down | Up | Controller 2 | No |

Total Items: 2

When choosing two interfaces to combine into a RoCE HA pair, common practice is to have the primary controller use its primary interface, and the standby controller use its secondary interface (because it will be idle and unused until failure).

## Linux device mapper multipath (dm-multipath)

Device mapper multipathing (or dm-multipath) is a Linux native multipath tool which allows you to configure multiple I/O paths between clients and storage arrays to behave as a single, highly-available link. These I/O paths are physical connections that should include separate cables, switches, and controllers. Multipathing aggregates the I/O paths, creating a new device that is used by the operating system instead of either of the individual paths to access storage.

To create a multipath (HA) connection to a Pavilion volume, first the volume must be configured to add a secondary controller to handle connectivity in case of primary failure, and then Linux must be configured to combine these primary and secondary controller paths to the same volume into a single, unified, HA device node.

### Multipath installation on the client

Install the device mapper multipath extension using the Yum command on any client that will be using it:

```
# yum install -y device-mapper-multipath
```

Enable and start multipathed service. It will continue to be active through reboots.

```
# systemctl enable multipathd.service
# systemctl start multipathd.service
```

### Multipath configuration on the client

This section describes configuring the multipath.conf file as well as modifying UDEV to enable automatic pairing of multipath devices by “`Device Serial`” or “`subnqn`”.

The device mapper multipathing uses the configuration file /etc/multipath.conf for the configuration. If you make any changes to this file the multipath command must be run in order to reconfigure the multipathed devices.

The easiest way to create this file is to use the “`mpathconf`” utility. If there is an existing configuration file mpathconf will edit it, if no such file exists it will copy a default version.

```
    # mpathconf --enable --with_multipathd y --with_chkconfig y
```

Add the following configuration to “/etc/multipath.conf” using an editor and restart the multipath service to load the changes:

```
    defaults {
                uid_attribute ID_WWN
                user_friendly_names yes
                find_multipaths yes
                no_path_retry "queue"
    }

    blacklist {
                devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
                devnode "^(td|hd|vd)[a-z]"
                devnode "^dcssblk[0-9]*"
    }
    blacklist_exceptions {
                devnode "nvme*"
                property "(ID_WWN|SCSI_IDENT_.*|ID_SERIAL|DEVTYPE)"
    }
    devices {
                device {
                            vendor "NVME"
                            product ".*"
                            uid_attribute "ID_WWN"
                            path_checker "none"
                }
                device {
                            vendor "PVL-*"
                            path_grouping_policy "failover"
                            path_selector "queue-length 0"
                            path_checker "directio"
                            hardware_handler "1 alua"
                            prio "alua"
                            prio_args "exclusive_pref_bit"
                            failback manual
                            rr_weight "priorities"
                            fast_io_fail_tmo 25
                }
    }
    multipaths {
    }
```

This configuration tells the multipath daemon to look for Pavilion volumes which appear on the client and try and match them by the “Device Serial” into multipath pairs, while not touching standard local devices.

To give the multipath daemon access to the “Device Serial” field of attached Pavilion volumes, add the rule below to the file “/lib/udev/rules.d/60-persistent-storage.rules” and reload it.

```
# echo 'KERNEL=="nvme*[0-9]n*[0-9]", ENV{DEVTYPE}=="disk", ATTRS{wwid}=="?*",
ENV{ID_WWN}="$attr{wwid}"' >> /lib/udev/rules.d/60-persistent-storage.rules
```

```
      # udevadm control --reload-rules
```

At this point, the multipath daemon will automatically assemble Pavilion volumes connected into a multipath device under the `/dev/mapper/mpathXXX`.

To form the multipath volume pair, you will need to connect to the primary and standby interfaces you chose in the prior steps using the "nvme connect" command as described in the basic "`Connecting a volume to a client`" section above. Note that in this case, the two "`nvme connect`" commands will have the same "`-n <Device Serial>`" but different IP addresses. For example:

```
      # nvme connect -a 192.168.10.14 -t rdma -n GB00001404bbf91
      # nvme connect -a 192.168.10.15 -t rdma -n GB00001404bbf91
```

To check that the multipath daemon has merged the two new NVMe over RoCE devices into a single `/dev/mapper/mpathXX` file, use "`lsblk`" or the "`multipath -ll`" command:

```
      # lsblk
      NAME MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
      sda      8:0      0 298.1G  0 disk
      ├─sda1            8:1         0   512M  0 part  /boot
      ├─sda2            8:2         0  15.8G  0 part  [SWAP]
      └─sda3            8:3         0 281.9G  0 part  /
      sr0         11:0        1  1024M  0 rom
      nvme0n1  259:0 0   200G  0 disk
      └─mpathg 253:0          0   200G  0 mpath
      nvme1n1  259:1 0   200G  0 disk
      └─mpathg 253:0          0   200G  0 mpath

      # multipath -ll
      mpathi (eui.36323862313537662d616161372d3434) dm-2 NVME,PVL-
      MX09S0P2L2C1-F100TP0TY1
      size=200G features='1 queue_if_no_path' hwhandler='0' wp=rw
      |-+- policy='service-time 0' prio=0 status=enabled
      | `- 5:0:1:0 nvme5n1 259:5 failed faulty running
      `-+- policy='service-time 0' prio=1 status=active
        `- 4:0:1:0 nvme4n1 259:4 active ready running
```

These multipath configurations, once done, will persist across reboots and integrate automatically with the automatic NVMe-over-RoCE reconnection utility "PDS" described above.

If for some reason you need to destroy the multipath setup and disconnect the individual NVMe over RoCE volumes, use “`multipath -f mpathXX`” and “`nvme disconnect`” as follows:

```
    # multipath -f mpathg
    # nvme disconnect -d nvme0n1
    # nvme disconnect -d nvme1n1
```

# Deploying IBM Spectrum Scale

## Installation

IBM Spectrum Scale comes in a self-extracting bundle, “`Spectrum-Scale-Standard-5.0.x.y-x86_64-Linux-install`”.

Copy this package to one of the cluster node’s /tmp/ folder (copying might change executable permission, change it using chmod +x <package-name>)

To get the list of all packages, run following command:

```
Spectrum-Scale-Standard-5.0.x.y-x86_64-Linux-install –manifest
```

Invoke the above mentioned self-extracting package

```
./Spectrum-Scale-Standard-5.0.x.y-x86_64-Linux-install
```

Once the extraction is done, the extracted packages can be located at `/usr/lpp/mmfs/5.0.x.y/`

Verify all the following packages are present:

```
    gpfs.base-5.0.*.rpm
    gpfs.gpl-5.0.*.noarch.rpm
    gpfs.msg.en_US-5.0.*.noarch.rpm
    gpfs.gskit-8.0.50.*.rpm
    gpfs.license*.rpm
    gpfs.ext-5.0.*.rpm
    gpfs.compression-5.0.*.rpm
    gpfs.crypto-5.0.*.rpm
    gpfs.adv-5.0.*.rpm
```

Optional packages

```
    gpfs.java-5.0.*.rpmvgpfs.callhome-ecc-client-5.0.*.rpm
    gpfs.callhome-5.0.*.rpm
    gpfs.gui-5.0.*.rpm
    gpfs.kafka-5.0.*.x86_64.rpm(Red Hat Enterprise Linux x86_64 only)
    gpfs.librdkafka-5.0.*.x86_64.rpm(Red Hat Enterprise Linux x86_64 only)
    gpfs.gss.pmcollector-5.0.*.rpm
    gpfs.gss.pmsensors-5.0.*.rpm
```

Following steps to be used in installing the Standard edition

```
cd /usr/lpp/mmfs/5.0.x.y/gpfs_rpms
rpm -ivh gpfs.base*.rpm gpfs.gpl*rpm gpfs.license.std*.rpm gpfs.gskit*rpm
gpfs.msg*rpm gpfs.ext*rpm gpfs.compression*rpm
```

Following steps to be used in installing the advanced edition

```
cd /usr/lpp/mmfs/5.0.x.y/gpfs_rpms
rpm -ivh gpfs.base*.rpm gpfs.gpl*rpm gpfs.license.adv*.rpm gpfs.gskit*rpm
gpfs.msg*rpm     gpfs.ext*rpm     gpfs.compression*rpm     gpfs.adv*rpm
gpfs.crypto*rpm
```

Build GPFS portability layer using following command

```
/usr/lpp/mmfs/bin/mmbuildgpl --build-package
```

The package can be located at the following location:

```
/root/rpmbuild/RPMS/x86_64/gpfs.gplbin-3.10.0-229.el7.x86_64-5.0.0-
0.x86_64.rpm
```

Install above RPM and set environment variable $PATH to include this location

```
Export $PATH=$PATH:/usr/lpp/mmfs/bin/:
```

## Deployment

- Determine cluster nodes nd1 to ndX; Dedicate 2 nodes as quorum managers
  - Make sure password-less access to all nodes from quorum managers

- Steps to create IBM Spectrum Scale Cluster
  - At quorum manager node, create node-list file with following content (ndX represent the name of the cluster nodes):

```
cat node-list
nd1:quorum
nd2
nd3
nd4:quorum-manager
nd5:quorum-manager
nd5
..
..
ndX
```

  - Run following command to create cluster

```
mmcrcluster -N nodes-list --ccr-enable -r/usr/bin/ssh -R /usr/bin/scp -C
PavSpectrum
```

- Install the license for all nodes in the cluster

- Run command "`mmlscluster`" to see the status and info about the cluster

- Run "`mmstartup -a`" to start GPFS daemon on all cluster nodes
  a. Check status of GPFS daemon on all cluster node "mmgetstate -a"
  b. Add the `NSDDEVICE` file on all the nodes which will qualify NVMe device as the block device accepted by GPFS.

- Steps to create NSD & File System
  - At quorum manager create NSD creation stanza file

```
cat pavnsd.cfg
%nsd: nsd=pavNSD000  device=/dev/nvme0n1  usage=dataAndMetadata
%nsd: nsd=pavNSD001  device=/dev/nvme1n1  usage=dataAndMetadata
%nsd: nsd=pavNSD002  device=/dev/nvme2n1  usage=dataAndMetadata
%nsd: nsd=pavNSD003  device=/dev/nvme3n1  usage=dataAndMetadata
%nsd: nsd=pavNSD004  device=/dev/nvme4n1  usage=dataAndMetadata
%nsd: nsd=pavNSD005  device=/dev/nvme5n1  usage=dataAndMetadata
```

  - Run the following command to create NSD

```
mmcrnsd -F pavnsd.cfg -v no
```

  - Run the following command to create file system named `pavfs`

```
mmcrfs pavfs -F pavnsd.cfg -A yes -B 128K -D posix -i 4k -j
scatter -m 2 -M 2 -Q yes -r 1 -R 2 -T /gpfs  -t Z -z no --
inode-limit 500M --filesetdf --perfileset-quota -S yes
```

  - Mount `pavfs` using following commands on all cluster nodes

```
mmmount pavfs -a
```

  - See the status and information about the `pavfs` with following command

```
mmlsfs pavfs
```

## Spectrum Scale Performance Measurements from All NVMe solution

In order to baseline the performance characteristics, a series of IO benchmarks were performed using the tool “FIO” using a setup of up to 30 Spectrum Scale servers (dual socket E5-26xxv3 CPUs with 256GB DDR4 and ConnectX-5 NICs) and 2 Pavilion HFAs. The performance of the cluster scaled up as new volumes/systems added to the cluster. The tests were conducted using both DIO & buffered I/O options, on a file system that was more than 90% full. Following is the table for empirical performance numbers.

| **30 node cluster (DIO) with 2 Pavilion HFAs** | | | | |
|---|---|---|---|---|
| **I/O** | RandRead (in IOPS) | RandWrite (in IOPS) | Seq read in GBps | Seq Write in GBps |
| **Direct** | 26M | 6M | 200GBps | 150GBps |
| **Buffered** | 6M | 3M | 200GBps | 150GBps |

In a big Spectrum Scale solution cluster with 10 of Pavilion HFA systems, performance could reach more than 130M random read IOPS and 1TB/s Sequential read throughput.

## Hierarchical Storage Management (HSM) solution using Pavilion HFA and JBOD/RBODs

### Introduction to HSM

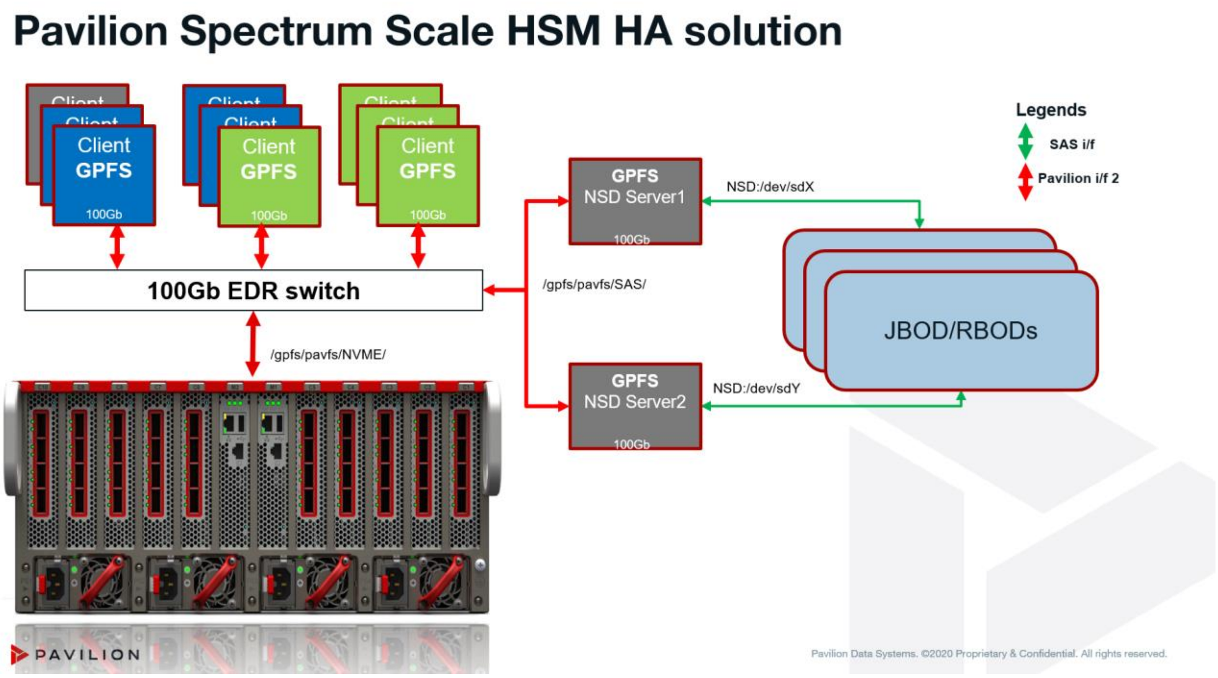
HSM is a data management technique that moves data (File, blocks and objects) between higher performance storage tier (i.e. NVMe, NVMe-oF etc.) and lower performance storage tier (i.e. Disks, JBOD and/or RBOD). It is somewhat similar to Processor Cache and DRAM. Data on the higher performance storage tier is usually an active data set and data on lower performance storage tier devices is usually a passive data set. All HSMs work on the basis of user configured policies and depending on those policies, portions of the dataset (file, block or objects) will move to/from high performance storage and low performance storage.

### Pavilion HFA Spectrum Scale HSM HA Architecture

Spectrum Scale HSM can be easily implemented using Pavilion HFA and lower performance JBOD/RBODs. The HSM depending on the policy set by users automatically optimizes usage by moving data between NVMe storage (i.e. NVMe-oF) provided by Pavilion and SAS HDD storage provided by the JBOD/RBODs.
Spectrum scale implements HSM using NSD server based implementation for accessing HDDs. The SAS HDD based JBOD/RBODs are directly connected to the NSD servers, which provide the services to the GPFS clients, when they need data stored in those JBOD/RBODs.

In the presented solution, the filesystem was created with 3 different storage pools which are as follows:

- System pool: Provides the storage for metadata which is hosted on the NVMe-oF storage devices carved out of Pavilion HFA
- NVMeoF Pool: Provides the storage for Hot data and this pool is created with the NSDs; this is again carved out of Pavilion HFA
- SAS pool: Which consists of NSDs; created out of SAS JBOD/RBOD.


Pavilion Spectrum Scale HSM HA solution

## Pavilion HFA Spectrum Scale HSM Policies

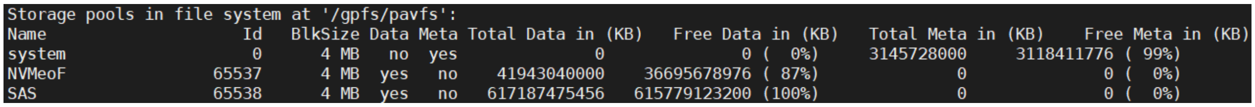
Spectrum Scale solution provides policy-based rules to be applied on files for following set of operations:

- File placements
- Snapshot placements
- Group pools
- File migrations
- File deletions
- File exclusions
- File lists
- File restores

- External storage pool definitions
- External list definitions

In the above set of policies there is a policy definition for File migration, which is being used for HSM and can be developed to create the active data always in high performance NVMe storage. The files that are no longer being accessed in succession can be moved to a lower performance tier of storage (i.e. HDDs, JBODs or RBODs).

For the current solution following pools have been created:



```
Storage pools in file system at '/gpfs/pavfs':
Name                    Id   BlkSize Data Meta Total Data in (KB)   Free Data in (KB)   Total Meta in (KB)    Free Meta in (KB)
system                   0      4 MB   no  yes                0              0 (  0%)     3145728000      3118411776 ( 99%)
NVMeoF               65537      4 MB  yes   no      41943040000    36695678976 ( 87%)              0               0 (  0%)
SAS                  65538      4 MB  yes   no     617187475456   615779123200 (100%)              0               0 (  0%)
```

Spectrum Scale presents an interface to implement HSM by setting-up the file migration policies. The policies are based on SQL syntax and can be modified by the Spectrum Scale administrators. The policy provides criteria for the selection of a file or a group of files. The files thus selected are migrated to/from two different storage pools as described in the policy. As described in the Spectrum scale documentation following is the syntax for file migration policy:

```
RULE ['RuleName'] [WHEN TimeBooleanExpression]
  MIGRATE
     [COMPRESS ({'yes' | 'no' | 'z' | 'lz4' | 'zfast' | 'alphae' |
'alphah'})]
     [FROM POOL 'FromPoolName']
     [THRESHOLD
(HighPercentage[,LowPercentage[,PremigratePercentage]])]
     [WEIGHT (WeightExpression)]
  TO POOL 'ToPoolName'
     [LIMIT (OccupancyPercentage)]
     [REPLICATE (DataReplication)]
     [FOR FILESET ('FilesetName'[,'FilesetName']...)]
     [SHOW (['String'] SqlExpression)]
     [SIZE (numeric-sql-expression)]
     [ACTION (SqlExpression)]
     [WHERE SqlExpression]
```

Following example shows a sample migration policy based on the file size and access time for any of the files on the NVMeoF storage pool (*pavhsmmig.pol*):

```
define(ACCESS_AGE,(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME)))
RULE ['default'] SET POOL 'NVMeoF'
RULE ['NVMeoFPlacement'] SET POOL 'NVMeoF'
    FOR FILESET ('NVMeoF')
RULE ['SASPlacement'] SET POOL 'SAS'
    FOR FILESET ('SAS')
RULE ['MigrateFilesPolicy'] MIGRATE From POOL 'NVMeoF'
```

```
TO POOL 'SAS'
WHERE (
    FILE_SIZE > 5242880
    AND
    ACCESS_AGE > 2
)
RULE ['BackMigrateFilesPolicy'] MIGRATE From POOL 'SAS'
TO POOL 'NVMeoF'
WHERE (ACCESS_AGE < 2)
```

This policy should be recorded in a file (say *pavhsmmig.pol*) and can be applied with the `mmchpolicy` command. The above policy will direct Spectrum Scale to move any file on the NVMeoF storage pool to SAS storage pool where the file has not been accessed for 3 days and the file size is greater than 5MB (P.S. Policy named `MigrateFilesPolicy`).

```
# mmchpolicy pavhsmfs pavhsmmig.pol -I yes
```

The policy can be changed at runtime by editing the above mentioned policy file and applying it again with the `mmchpolicy` and applying using the `mmapplypolicy` command.

```
# mmapplypolicy pavhsmfs -P pavhsmmig.pol -I yes
```

For a dry run of the policy, the following command can be used:

```
# mmapplypolicy pavhsmfs -P pavhsmmig.pol -I test
```

**Note**: The Migration Policies mentioned above could be much more extensive depending on the application and usage of the overall solution.

## Performance Expectations

The addition of the JBOD/RBODs to the Pavilion HFA Spectrum Scale HSM solution, provides a huge boost to the storage capacity of the overall solution. For the best performance out of this HSM solution, the entire active dataset should be present in the NVMeoF Pool (which is the current file placement policy); and the first access to the data stored in SAS JBOD/RBOD pool, will incur the penalty of moving the data from lower performance tier to the higher performance tier but the subsequent accesses to that data will be similar to accessing files in the NVMeoF storage pool.

The performance of this solution will be similar to the All NVMe solution (shown in the previous section) with correct sizing and policies.

# For Further Information

For more detailed implementation assistance, please contact your Sales representative or use our support email support@paviliondata.io or phone centers available at https://paviliondata.io/support .