



NVMe over RoCE: Best Practices and Implementation Guide

Version 2.6

This document is intended to give a brief overview of NVM Express (NVMe) and provide a best practice for implementation of NVMe over RoCE (NVMeoF), followed by a step-by-step guide to configuring a client to use NVMe-oF with the Pavilion Hyperparallel Flash Array (HFA) under Linux. The user is expected to be familiar with typical system administration tasks, SAN/NAS terminology, and the general architecture of the Pavilion Hyperparallel Flash Array.



Summary	3
Key Recommendations.....	4
NVMe and NVMeOF.....	4
Configuring the Pavilion HFA for NVMeOF	6
Configuring a controller to use the RoCE protocol	7
Creating and Assigning Volumes on the Pavilion HFA	8
Host Configuration:.....	11
Host OS Compatibility Matrix:	11
Supported Client Network Adapters.....	11
RDMA Initiator Configuration (In-Box Drivers)	12
RDMA Initiator Configuration (Mellanox OFED Drivers).....	15
Connecting Volumes to Clients	17
Making Pavilion volumes persistent across reboots.....	20
Cluster-wide High Availability, High Performance Architectures	21
High Availability Features	21
Multipath Support	22
High Availability for the cluster using redundant switches	23
Multipath configuration for volumes on the Pavilion HFA	24
Linux device mapper multipath (dm-multipath).....	25
Testing and troubleshooting NVMe over RoCE hosts	28
Open source tools for testing	28
File systems Tested	29
Troubleshooting host side issues	30
NVMe over RoCE Performance Measurements	32
For Further Information	32

Summary

This document is intended to give guidance and present best practices for constructing a high performance, highly available NVMe over Fabrics (NVMe-oF) based compute and storage environment with the Pavilion Hyperparallel Flash Array (HFA).



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

- RDMA capable switch, configured for RDMA over Converged Ethernet
- 100, 40, or 25 Gbit networking (100Gb for best performance), MTU 9216
- Converged Ethernet NIC (Mellanox ConnectX-4 or newer) with MTU of 9000
- 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 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 NVMeOF

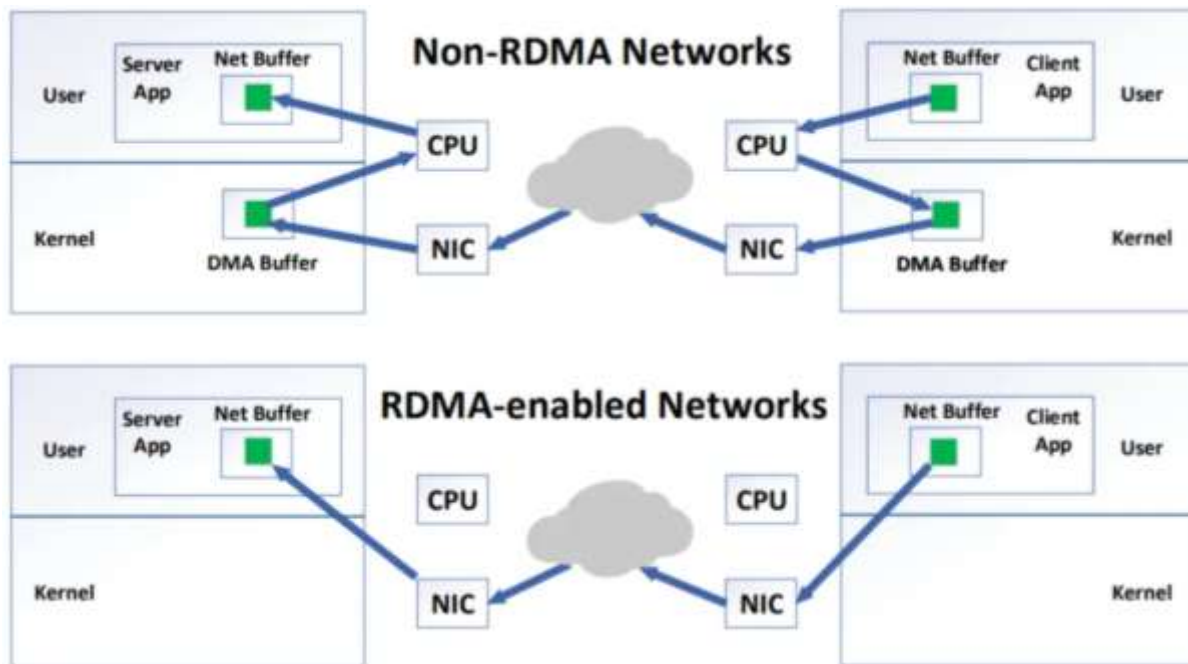
Non-Volatile Memory Express (NVMe) is an advanced protocol used to access flash storage on PCI Express bus. “Non-Volatile” stands for persistent storage, while “Express” refers to the fact that the data travels over the PCI Express (PCIe) interface on the computer's motherboard. This gives the drive 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-oF 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, however, is not supported by standard switches and NICs. 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.



InfiniBand (IB), an interconnect designed for HPC workloads, has had RDMA support built-in since its initial stages. IB does not see major deployment in general purpose data centers and is more commonly found in academic and supercomputing clusters.

Ethernet did not initially have support for RDMA, but today most modern high speed network switches (such as all high-end switches from Cisco, Arista, Dell, Mellanox, and others) and NICs at 25, 40, and 100Gbit (such as the Mellanox Connect-X series) support RDMA over Converged Ethernet (RoCE). Ethernet switches which support RDMA need to support high throughput, lossless packet forwarding (vs. the potentially lossy standard Ethernet) and communicate with attached devices to automatically throttle connections under congestion. The administrator often doesn't need to do a thing to ensure these lossless connections on modern switches, so it does not typically increase management overhead.



Legacy network infrastructures do not support RDMA. Please verify switches and NICs support RoCE (see the “Networking Best Practices Guide” for several tested models).

For legacy networks, NFS may be used instead. See the “NFS Best Practices Guide” for complete details on configuring the network and Pavilion HFA this way.

Both RDMA over Converged Ethernet (RoCE) and InfiniBand are supported via in-box drivers for Linux, or via add-on software called “OFED” which will be discussed later in this document. High-availability via standard multipath methods is fully supported, allowing for a resilient, always-on storage infrastructure using RoCE.

Configuring the Pavilion HFA for NVMeOF

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, or RoCE. 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, a process which is often done only once per controller on installation. After configuring it to use the RoCE 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

Configuring a controller to use the RoCE protocol

The following snippet shows the sequence of commands used in the CLI to enable the RoCE 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

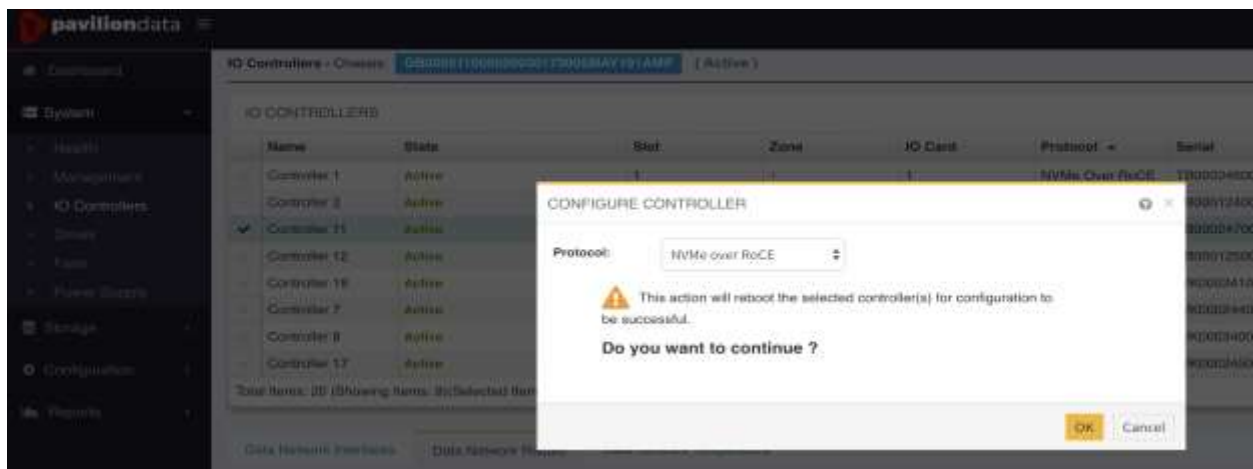
Creation of task was successful
Monitor task : 2d04159f-a094-4e86-93de-cd935e240f93
Command format : show task id [id]
```



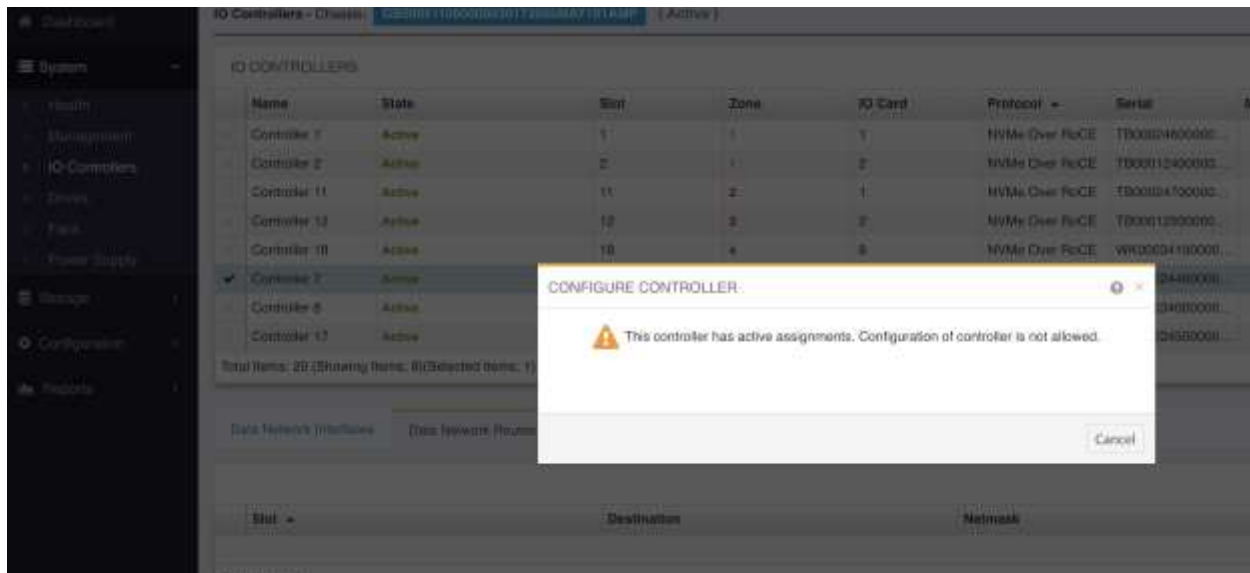
The Pavilion HFA utilizes an asynchronous queue for configuration changes. When a long-running configuration change is requested, a background process (“task”) is created to perform the request and control returned to the user immediately.

The “Creation of task was successful” messages indicate that the command is

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”:



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.



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 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

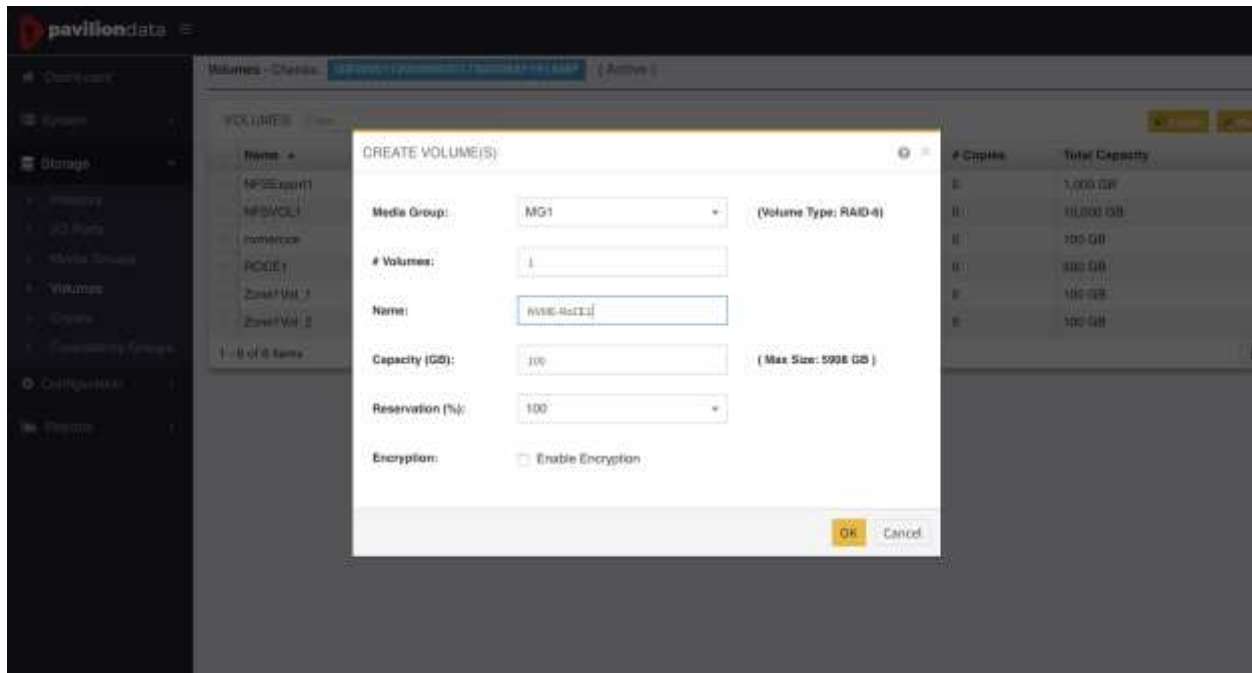
Create a volume using the CLI with a single command as shown below.

```
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.



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

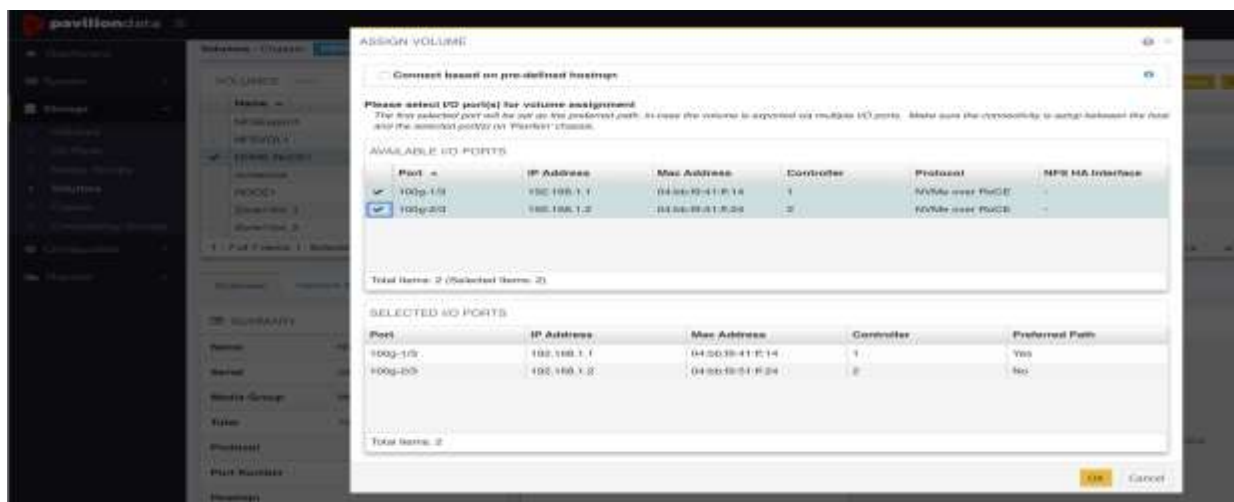
Creation of task was successful
Monitor task : 7bc94647-118d-46fc-a3ed-467d21efcb64
Command format : show task id [id]

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:



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):

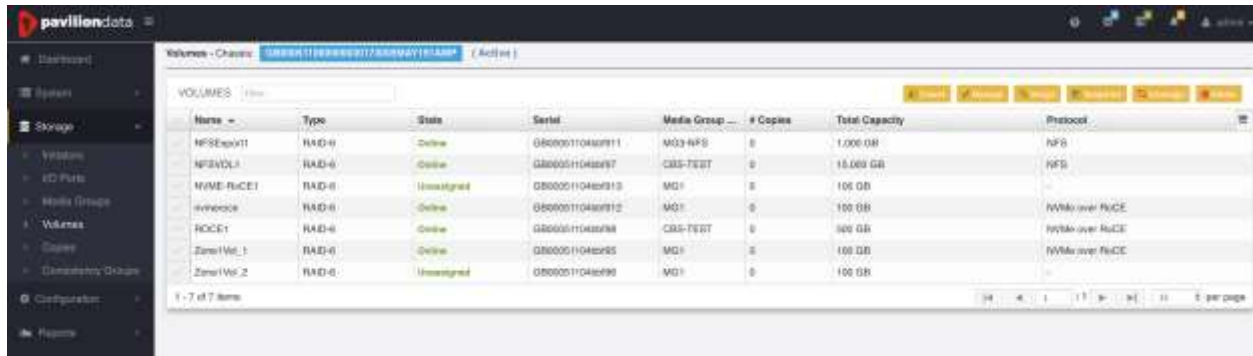
```
admin@GB0...AMP> switch show
Switched to show namespace

(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:



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 NVMeOF 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*
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.



For most users running CentOS 7.5 or newer, the inbox drivers shown here will be the preferred method unless the additional Mellanox utilities are desired. (for such things as upgrading NIC firmware, etc.). For earlier CentOS releases, the Mellanox drivers may provide a better user experience.

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.



As stated earlier, for most users running CentOS 7.5 or newer, the inbox drivers are sufficient and there is no need to install the Mellanox OFED drivers. For users who require using an earlier CentOS release or kernel, the OFED drivers can enable NVMe-over-RoCE support when the inbox drivers are buggy or even missing completely (CentOS 6.x, etc.).



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-nvme`” option specified. **NOTE: It is very important to have the “`--with-nvme`” 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-nvme
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 -orelative

# 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 volumes to connect to 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=(GB00000001 GB00000002)
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

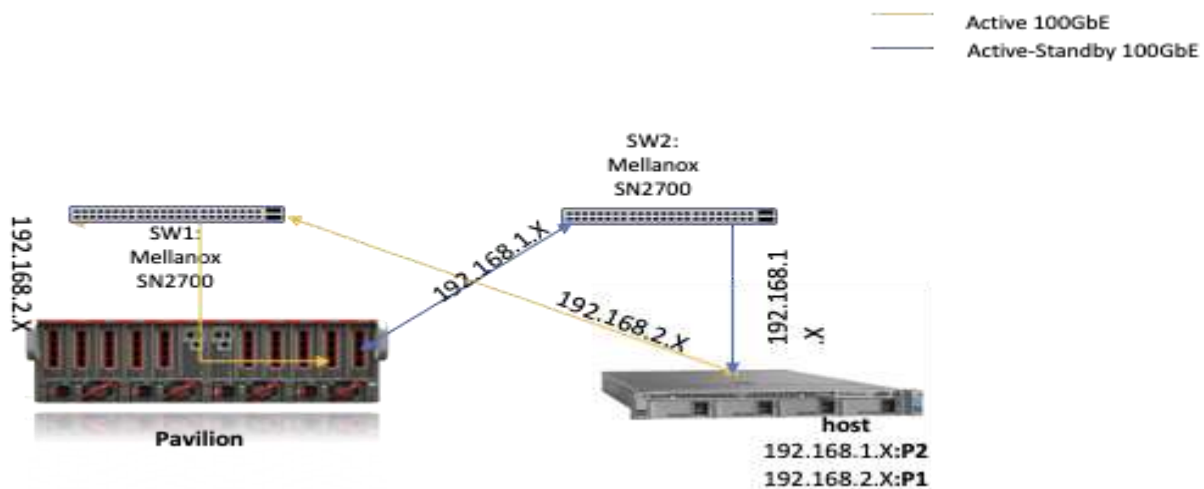
Pavilion's patented architecture employs a high-speed PCIe network to connect all of the internal components, including I/O Line Cards, the NVMe Drive Array, and the Supervisor Modules. This fabric is fully-redundant and is implemented on dual-redundant swappable PCIe switch cards contained in the chassis.

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.

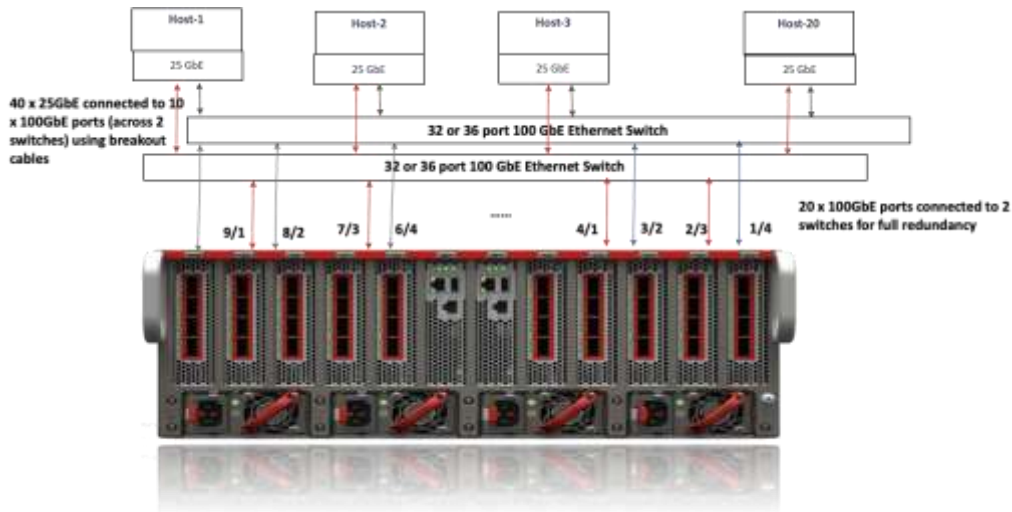
This section describes the settings and network topology required to provide HA client access to volumes on the Pavilion HFA. An understanding of networking and high availability concepts is assumed, and past experience with Linux multipathing is useful. Multiple NICs, switches, and cabling are required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



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 4 GB/s of throughput, and this number should be used as a starting point for architectural designs that are performance critical.

For a 25G Ethernet client environments, where individual clients require approximately 1GB/s of bandwidth (NoSQL databases, containers, etc.), Pavilion recommends assigning up to 4 volumes to each of the Pavilion HFA controllers. This will allow for up to 80 clients, each with their own private volumes.

In HPC-type environments (HFT, engineering analysis, drug discovery, etc.) utilizing 100 G for client connectivity, assigning a single volume to each controller (with HA configurations as described in prior sections) supports 20 clients with optimal bandwidth availability.

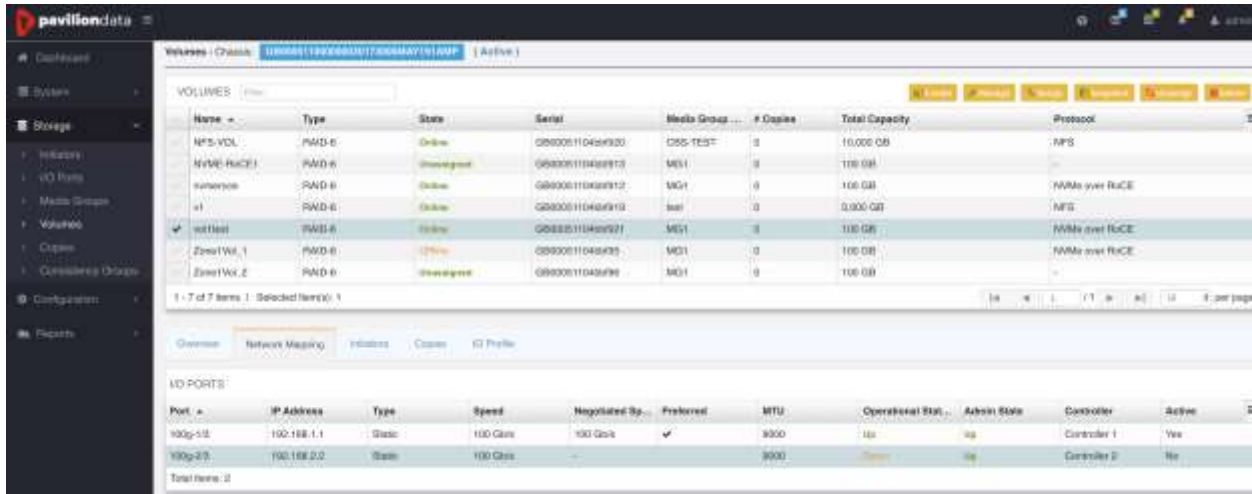
Multi-Path I/O Support

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.”



The screenshot shows the Pavilion GUI with the 'VOLUMES' section selected. Below it, the 'IO PORTS' section is visible, showing a table of network interfaces.

Name	Type	Status	Serial	Media Group	# Copies	Total Capacity	Protocol
hps-vol	RWD-6	Online	059005110400030	OS-TEST	0	10,000 GB	NFS
hps-vol-1	RWD-6	Overprovisioned	059005110400013	MS1	0	100 GB	NVMe over RoCE
hps-vol-2	RWD-6	Online	059005110400012	MG1	0	100 GB	NVMe over RoCE
hps-vol-3	RWD-6	Online	059005110400010	MG1	0	0,000 GB	NFS
hps-vol-4	RWD-6	Online	059005110400021	MS1	0	100 GB	NVMe over RoCE
hps-vol-5	RWD-6	Overprovisioned	059005110400005	MG1	0	100 GB	NVMe over RoCE
hps-vol-6	RWD-6	Overprovisioned	059005110400006	MG1	0	100 GB	NVMe over RoCE

Port	IP Address	Type	Speed	Negotiated Sp...	Preferred	MTU	Operational Stat...	Admin State	Controller	Active
100g-1.1	100.100.1.1	Gigabit	100 Gbps	100 Gbps	✓	9000	Up	Up	Controller 1	Yes
100g-2.2	100.100.2.2	Gigabit	100 Gbps	---		9000	Down	Up	Controller 2	No



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 multipathd 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
```

Testing and troubleshooting NVMe over RoCE hosts

Open source tools for testing

FIO

Flexible IO tester (FIO) fio is a tool that will spawn a number of threads or processes doing a particular type of I/O action as specified by the user. The typical use of fio is to write a job file matching the I/O load one wants to simulate.

EZFIO

ezFIO is a script-based tool optimized for demonstrating the sustained performance of NVMe storage devices over a series of varying, enterprise-class workloads. It wraps the cross-platform open-source tool FIO within an easy to use GUI (Windows) or CLI (Windows and Linux).



IOZONE

iozone does a benchmarking on different types of file system performance metrics. It measures the performance of file systems by running filesystem operations including read, write, re-read, re-write, read backwards read strided, random read, and so on.

Bonnie++

Bonnie++ is a benchmark suite that is aimed at performing a number of simple tests of hard drive and file system performance. Bonnie++ allows you to benchmark how your file systems perform with respect to data read and write speed, the number of seeks that can be performed per second, and the number of file metadata operations that can be performed per second.

File systems Tested

Pavilion HFA has been tested with the following file systems

Extended File System (ext 3 & 4): These file systems are very robust, ext 3 was the default file system for RHEL release 3 and 4. Ext4 is successor of ext3 and supports features like large filesystems (upto 16 TiB) better performance and increased reliability and is default file system for most linux distributions.

XFS: XFS is highly scalable and high-performance file journaling file system and is supported by most Linux distributions. XFS is default file system on RHEL, CentOS 7 and many other distributions. XFS file system can be expanded while mounted and active and cannot be reduced in size.

Btrfs: b-tree file system based on copy-on-write (Copy-on-write is a mechanism, when a write request is made, the data are copied into a new storage area, and then the original data are modified).

Glusterfs: Gluster is a scalable, distributed file system that aggregates disk storage resources from multiple servers into a single global namespace which can scale upto several petabytes of data and has the ability to handle thousands of clients.

GPFS: General parallel file system also known as IBM Spectrum scale is a high-performance clustered file system. Spectrum Scale provides concurrent high-speed file access to applications executing on multiple nodes of clusters.

BGFS: BeeGFS is open source file system and is available for free with basic features. BeeGFS is a parallel cluster file system spreading data across multiple disks & server(s).

LustreFS : Lustre is a type of parallel distributed file system, generally used for large-scale cluster computing. Lustre file systems have high performance capabilities and open licensing and is often used in supercomputers. Lustre file systems are scalable and can be part of multiple computer clusters with tens of thousands of client nodes, tens of petabytes (PB) of



storage on hundreds of servers, and more than a terabyte per second (TB/s) of aggregate I/O throughput.

Troubleshooting host side issues

NVMe Discover/Connect:

When trying to discover nvme target, If no parameters are given, then 'nvme discover' will attempt to find a /etc/nvme/discovery.conf file to use to supply a list of Discovery commands to run. If no /etc/nvme/discovery.conf file exists, the command will quit with an error. Otherwise, a specific Discovery Controller should be specified using the --transport, --traddr, and if necessary, the --trsvcid flags. A Discoveryrequest will then be sent to the specified Discovery Controller.

Below are the most frequently encountered error messages on the host side while trying to discover/connect NVMe target.

Error-1 : Failed to write to /dev/nvme-fabrics: Connection timed out

Solution: This is a very generic error from nvme stack, we need to check dmesg to get more info. Error message under dmesg

```
dmesg shows error like below
[1024681.236520] nvme nvme0: rdma_resolve_addr wait failed (-110).
```

Problem might be

- a. Wrong IP address of the target in the command
- b. Check if controller is reachable.

Error-2 : "Failed to write to /dev/nvme-fabrics: Invalid argument"

By default MOFED installs a dummy nvme_rdma module. You can see in lsmod/modinfo that it doesn't have any dependencies.

```
#ofed_info -s
MLNX_OFED_LINUX-4.5-1.0.1.0:
# » modinfo nvme_rdma
filename: /lib/modules/3.10.0-957.el7.x86_64/extra/mlnx-
ofa_kernel/drivers/nvme/host/nvme-rdma.ko
version: 2.0.0
license: Dual BSD/GPL
description: nvme-rdma dummy kernel module
```

Solution: Uninstalled the MOFED and installed it with --with-nvmf flag

Error-3: Failed to open /dev/nvme-fabrics: No such file or directory

Solution: One of the probable reasons for this error could be: nvme drivers not loaded

```
#lsmod|grep nvme
<<<< No drivers are loaded
# modprobe nvme_rdma --- load the drivers using modprobe
#lsmod|grep nvme
nvme_rdma 28408 0
nvme_fabrics 19997 1 nvme_rdma
nvme_core 58852 2 nvme_fabrics,nvme_rdma
rdma_cm 59673 6 rpcrdma,ib_srp,nvme_rdma,ib_iser,rdma_ucm,ib_isert
ib_core 242235 15
rdma_cm,ib_cm,iw_cm,rpcrdma,mlx4_ib,ib_srp,ib_ucm,nvme_rdma,ib_iser,ib_srpt,i
b_umad,ib_uverbs,rdma_ucm,ib_ipoib,ib_isert
```

Solution: Load the nvme_rdma module with -v option.

```
# modprobe nvme_rdma
```

Error-4: Connection reset by peer

Solution: Check if controller is reachable by following ping command

```
Ping < ip address of target> -s 8750 -M do
```

If ping is also successful, then check for mlx4_ib or mlx5_ib driver.

```
# lsmod | grep mlx5_ib
# lsmod | grep mlx4_ib
# <<<<mlx4_ib/ mlx5_ib were not loaded,
```

even though network connectivity was fine between host and target, RDMA connections were failing to establish.

Solution: Load the mlx_ib module to fix the issue

```
# modprobe mlx4_ib
# modprobe mlx5_ib
```

Error-5 : Discover and connect failing from an rdma host

All the nvme and mlx modules are loaded, targets are pingable, jumbo frames are enabled.

Solution: rdma/infiniband stack in centos 7.5 kernel (3.10.0-862.11.6.el7.x86_64 – released on August 15th) is broken. This has been reported in the Red hat and centos community.

Centos - <https://bugs.centos.org/view.php?id=15193>

Redhat - <https://access.redhat.com/solutions/3568891>

Solution: Upgrade the kernel.



NVMe over RoCE Performance Measurements

In order to baseline the various performance characteristics, a series of IO benchmarks were performed using the tool “FIO” using a setup of up to 20 servers (dual socket E5-2690v3 CPUs with 64GB DDR4 and ConnectX-4 NICs). The tests were conducted on fully pre-conditioned volumes. Here are the numbers from those tests.

I/O (Jobs X QD X Block size)	2 Card Config (4 servers)	5 Card Config (10 servers)	10 Card Config (20 servers)
Random Read (16X32X4K)	3.4M IOPS	8.4M IOPS	16.9M IOPS
Random write (16x32x4K)	0.9M IOPS	2.2M IOPS	4.5M IOPS
Seq Read (1 X 1024 X 128K)	14.4GBps	37.1GBps	75.3GBps
Seq Write (1 X 1024 X 128K)	12.8GBps	32.3GBps	64.8GBps
Read latency (1 X 1 x 4K)	118 us		
Write latency (1 X 1 X 4K)	47 us		

For Further Information

We hope this Best Practices guide has been helpful to you in understanding NVMe-oF, RoCE, and the most performant and highly available way of configuring them on the Pavilion HFA. 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>.