Setup and Performance Testing of Lenovo Thinksystem DE6600F Systems with BeeGFS

Conor Elrick Simon Thompson

Contents

Revisions

November 2025

- Updated for BeeGFS 8
- Added some improvements to filesystem and BeeGFS tuning
- Added IOR performance data demonstrating full stripe write acceleration.
- Added MDTest performance data
- Fixed some spelling typos

Abstract

This document details the deployment of a Lenovo Thinksystem DE6600F[1] in a test benchmark system with storage drives configured using NVMe-over-fabrics technology. We outline all the steps that were required to configure the system, the clients and the drives initially to be ready for deployment of a parallel filesystem. We then detail the installation of a BeeGFS[2] parallel filesystem on our benchmark cluster using the configured setup. Finally, we will then finish by discussing some IO performance measurements on our created filesystem and discuss sizing considerations when looking to scale out to more systems.

[1] https://lenovopress.lenovo.com/lp1643-thinksystem-de6600f-and-de6600h-storage-arrays
[2] https://www.beegfs.io/

1 Overview

The Lenovo Thinksystem DE6600 series are storage systems designed to make use of current NVMe technology to provide both high capacity and performance of IO in the datacentre. The series consists of the DE6600F all-flash and the DE6600H hybrid storage arrays and are designed to be compatible with a wide range of connectivity options depending on what makes sense to use in the cluster.

Both the DE6600F and DE6600H models are 2U systems with bays for 24 2.5" SFF NVMe drives of capacity ranging from 1.9TB to 15.3TB. The total capacity can be increased by the inclusion of Lenovo Thinksystem DE600S, DE240S or DE120S storage expansions, detailed on the Lenovo Press page for the DE6600 systems[3].

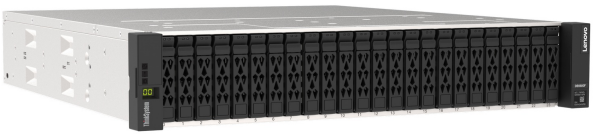

Figure 1: Lenovo Thinksystem DE6600F. Image taken from Lenovo Press[4].

In this document we will detail the installation of two DE6600F systems in our test benchmark centre starting with the network topology considerations and the initial deployment of the configuration options using the command line interface. We then discuss setting up the DE6600F systems as storage controllers for collections of network drives using NVMe-over-fabrics technology with the goal to build a cluster configuration compatible with BeeGFS.

With the cluster configured, we detail the setup of BeeGFS and build some test filesystems which we can mount on client nodes for some IO benchmarking. Finally, we finish by providing example benchmark results using a couple of standard applications and outline the sizing of potential scaled-out clusters based on both performance and capacity requirements.

2 BeeGFS Overview

In this setup, we will use BeeGFS version 8.1.0. Note there is a significant change in command syntax from earlier releases. In a BeeGFS cluster, nodes are designated into four categories as outlined in the online documentation[5]:

- Management Servers: For management of all of the other nodes in the cluster
- Metadata Servers: For writing/reading file metadata
- Storage Servers: For writing/reading data
- Client Servers: For mounting the BeeGFS filesystem and launching IO tasks

Note that these are not required to be exclusive groups - it is possible for a node to belong to more than one of these, running the services for each group it belongs to at the same time, provided it has sufficient resources. In particular, as the quick start guide[6] explains, the management service is not "performance-critical" as does not need to be run on a dedicated machine.

Each of the four server types above has associated with it a system service, but in addition for BeeGFS 7 and below there is also a fifth important service we will make use of in the deployment of beegfs which is the `helperd` service. The `helperd` service helps with logging and DNS lookups and is required to be started first when proceeding with the install in the manner detailed in the rest of the section.

In our benchmarking environment we will allocate the available storage servers according to the following:

- Management Server: `ece4709`
- Metadata Servers: `ece4709, ece4710`
- Storage Servers: `ece4709, ece4710`
- Client Servers: `ece4701-ece4708, ece4711-ece4712`

3 Hardware Configuration and Testing Environment

[3] https://lenovopress.lenovo.com/lp1643-thinksystem-de6600f-and-de6600h-storage-arrays
[4] https://lenovopress.lenovo.com/lp1643-thinksystem-de6600f-and-de6600h-storage-arrays
[5] https://doc.beegfs.io/latest/quick_start_guide/quick_start_guide.html
[6] https://doc.beegfs.io/latest/quick_start_guide/quick_start_guide.html

3.1 Infiniband network configuration of DE6600F systems

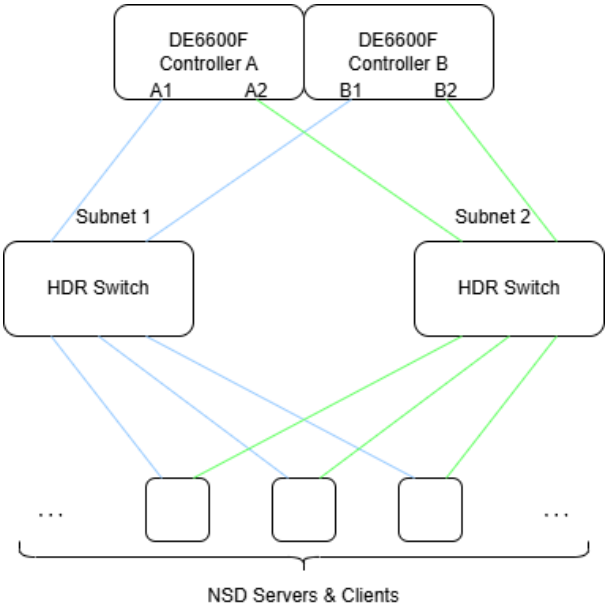

Figure 2: Hardware network setup for a single DE6600F system

Following the documentation available from NetApp online[7][8] the DE6600F are connected as shown in [Figure Above]. The first infiniband port on each of the controllers (labelled A1 and B1 in the diagram) are connected using HDR connections to a HDR ready switch with all connections on a single subnet. The second infiniband port on each of the controllers (labelled A2 and B2 in the diagram) are connected to a second HDR ready switch using a separate subnet to the one used with the connections in the first infiniband ports. Each of these switches are then connected with HDR connections to OSS servers again as outlined in [figure above] with each server connected to both switches using the matching subnet.

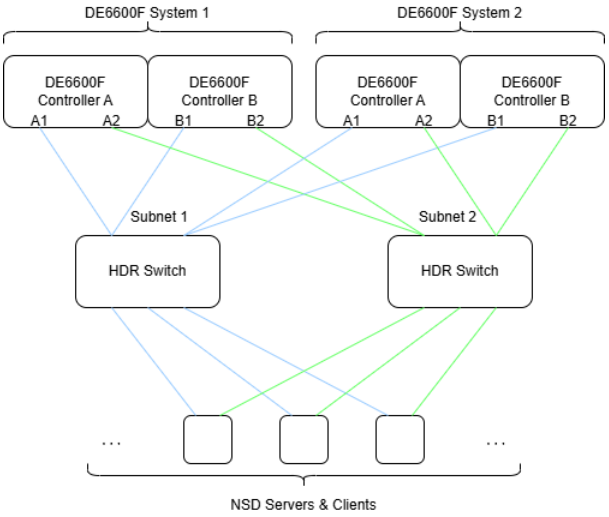

Figure 3: Hardware network setup for two DE6600F system

In our testing environment we extended the setup to use two DE6600F systems, each of which are connected to the HDR switches individually as outlined above for the single DE6600F system. All of the first infiniband ports across all four of the controllers are connected on one subnet to one of the switches, and all of the second infiniband ports across all four controllers are connected on a second subnet to the second of the switches.

[7] https://docs.netapp.com/us-en/e-series/install-hw-e2800-e5700/e2824-e5724-complete-setup-task.html#option-2-fabric-topology

[8] https://docs.netapp.com/us-en/e-series/config-linux/nvme-ib-worksheet-concept.html

4 Setup the DE6600F

Once we have the systems connected as outlined in the previous sections, we begin the setup of the systems using the linux terminal. For this we need to obtain SMcli from Lenovo Support page[9] on our management system.

In this section we will detail the setup of the DE6600F systems through the SMcli tool.

4.1 Install of SMcli

```
$ wget https://download.lenovo.com/storage/...<SMCLI_URL>
...
2024-09-03 11:44:15 (57.4 MB/s) - 'SMcli-01.81.54.9004.'zip saved [3877287/
    3877287]
$ mkdir SMcli
$ cd SMcli
$ unzip -q ../SMcli-01.81.54.9004.zip
$ ls
META-INF  MIB  SMcli-01.81.54.9004
$ chmod +x SMcli-01.81.54.9004/bin/SMcli
$ cd ../
```

The SMcli tool can be used to configure the storage array using a batch interface, it is often helpful to have the password for the storage array in a config file which means you do not have to enter the password for each cli command. Whilst you can pass the password to SMcli with the `-p` flag, this means the password will end up in the shell history.

```
echo "PASSWORD" > .depass
history -d -2
chmod 600 .depass
```

The second to last command above will remove the `echo` command from the shell history and the last command will set permissions so only the current user can read the contents of the password file.

4.2 Setup the base storage array

Once IP connectivity is possible to the management controller of the DE storage system, we need to setup the base configuration for the array. We can do this using the web interface or using the following batch configuration file:

```
on error stop;

show "Setting storageArray configuration...";
set storageArray userLabel="de4801";
// if not using NTP, sync the clocks
// set storageArray time;
set storageArray cacheBlockSize=32;
set storageArray cacheFlushStart=50;
set storageArray mediaScanRate=30;
set storageArray autoLoadBalancingEnable=false;
set storageArray hostConnectivityReporting disable;
set storageArray defaultHostType=28; // Linux DM-MP (Kernel 3.10 or later)

set storageArray autoSupport deliveryMethod=HTTPS proxyServer hostAddress="172.
    30.102.11" portNumber=3128;

set controller[a] DNSServers=(172.30.102.11);
set controller[b] DNSServers=(172.30.102.11);

set controller[a] NTPServers=(172.30.102.11);
```

[9] https://support.lenovo.com/us/en/solutions/HT514188


```
set controller[b] NTPServers=(172.30.102.11);

show "Setting controller NVMe-over-Fabrics hostports IPs...";
// HDR-100 4 ports will be 1a,1b,2a,2b, HDR 2 ports will be 2a,2b
set controller[a] hostPort["1a", "physical"] IPV4Address=172.30.120.31;
set controller[a] hostPort["1a", "virtual"] IPV4Address=172.30.120.32;
set controller[a] hostPort["1b", "physical"] IPV4Address=172.30.120.33;
set controller[a] hostPort["1b", "virtual"] IPV4Address=172.30.120.34;
set controller[b] hostPort["1a", "physical"] IPV4Address=172.30.120.35;
set controller[b] hostPort["1a", "virtual"] IPV4Address=172.30.120.36;
set controller[b] hostPort["1b", "physical"] IPV4Address=172.30.120.37;
set controller[b] hostPort["1b", "virtual"] IPV4Address=172.30.120.38;
show "Done.";
```

Note that we have setup the “a” (“b”) ports on each controller to be on the same subnet, and that the subnets for “a” and “b” are the same as each another. We apply this config with SMcli:

```
$ ./SMcli/SMcli-01.81.54.9004/bin/SMcli de4801a-mgt -u admin -P .depass  -k -f
    de4801-base.bat
Performing syntax check...

Syntax check complete.

Executing script...

Setting storageArray configuration...
Setting controller NVMe-over-Fabrics hostports IPs...
Done.
Script execution complete.

SMcli completed successfully.
```

Once this is complete we can test to make sure all the IB ports are configured correctly by pinging them:

```
for de in de4801 de4904; do
  echo $de;
  for ctrl in a b; do
    for ib in ib0 ib1; do
      ping -c1 -q ${de}${ctrl}-${ib}
    done
  done
  echo
done
```

4.3 Creating Volumes

On the DE6600F system, to take advantage of full stripe write acceleration (FSWA), RAID6 volume groups are required. For highest performance, 8+2P volumes using specific drives for the volume groups are required. More details on available RAID setups were presented in a proceeding section.

We apply the following batch configuration file to create RAID6 volume groups with the “central” 20 drives and RAID1 volume groups for the “outer” drives in the enclosure:

```
on error stop;

clear storageArray configuration volumeGroups;
clear storageArray recoveryMode;

show "Creating RAID6 volumes...";
create volumeGroup
  drives=(99,7 99,8 99,9 99,10 99,11 99,12 99,13 99,14 99,15 99,16)
```



```
  raidLevel=6 userLabel="de4801r6a01"
  drawerLossProtect=false trayLossProtect=false
  securityType=none dataAssurance=none;
create volumeGroup
  drives=(99,2 99,3 99,4 99,5 99,6 99,17 99,18 99,19 99,20 99,21)
  raidLevel=6 userLabel="de4801r6a02"
  drawerLossProtect=false trayLossProtect=false
  securityType=none dataAssurance=none;

show "Creating RAID1 volumes...";
create volumeGroup
  drives=(99,0 99,23)
  raidLevel=1 userLabel="de4801r1a01"
  drawerLossProtect=false trayLossProtect=false
  securityType=none dataAssurance=none;

create volumeGroup
  drives=(99,1 99,22)
  raidLevel=1 userLabel="de4801r1a02"
  drawerLossProtect=false trayLossProtect=false
  securityType=none dataAssurance=none;

show "Creating volumes on RAID6 volumeGroup...";
create volume volumeGroup="de4801r6a01" userLabel="de4801r6a01v01"
  capacity=57211GB owner=A segmentSize=32 cacheReadPrefetch=false
  dssPreAllocate=false securityType=none dataAssurance=none;
create volume volumeGroup="de4801r6a01" userLabel="de4801r6a01v02"
  capacity=57211GB owner=B segmentSize=32 cacheReadPrefetch=false
  dssPreAllocate=false securityType=none dataAssurance=none;

create volume volumeGroup="de4801r6a02" userLabel="de4801r6a02v01"
  capacity=57211GB owner=A segmentSize=32 cacheReadPrefetch=false
  dssPreAllocate=false securityType=none dataAssurance=none;
create volume volumeGroup="de4801r6a02" userLabel="de4801r6a02v02"
  capacity=57211GB owner=B segmentSize=32 cacheReadPrefetch=false
  dssPreAllocate=false securityType=none dataAssurance=none;

show "Creating volumes on RAID1 volumeGroup...";
create volume volumeGroup="de4801r1a01" userLabel="de4801r1a01v01"
  capacity=14302GB owner=A segmentSize=128 cacheReadPrefetch=false
  dssPreAllocate=false securityType=none dataAssurance=none;

create volume volumeGroup="de4801r1a02" userLabel="de4801r1a02v01"
  capacity=14302GB owner=A segmentSize=128 cacheReadPrefetch=false
  dssPreAllocate=false securityType=none dataAssurance=none;

show "Setting volume redundancy checks...";
set volume ["de4801r6a01v01"] redundancyCheckEnabled=true;
set volume ["de4801r6a01v02"] redundancyCheckEnabled=true;
set volume ["de4801r6a02v01"] redundancyCheckEnabled=true;
set volume ["de4801r6a02v02"] redundancyCheckEnabled=true;
set volume ["de4801r1a01v01"] redundancyCheckEnabled=true;
set volume ["de4801r1a02v01"] redundancyCheckEnabled=true;

show "Setting allVolumes configuration...";
set allVolumes cacheFlushModifier=10;
// set allVolumes cacheWithoutBatteryEnabled=false; // true==DANGER!;
set allVolumes mediaScanEnabled=true;
set allVolumes mirrorCacheEnabled=true;
set allVolumes readCacheEnabled=true;
set allVolumes writeCacheEnabled=true;
```



```
set allVolumes cacheReadPrefetch=false; // let GPFS do the prefetch show "Done
    .";
```

This config is then applied using SMcli:

```
$ ./SMcli/SMcli-01.81.54.9004/bin/SMcli de4801a-mgt -u admin -P .depass  -k -f
    de4801-volumes.bat
Performing syntax check...

Syntax check complete.

Executing script...…

Done.
Script execution complete.

SMcli completed successfully.
```

4.4 Mapping volumes to disks

Finally, we ensure the volumes are mapped to the hosts:

```
on error stop;

show "Creating hostgroup...";
create hostGroup userLabel="de4801_nsds";

show "Creating hosts...";
create host userLabel="ece4709" hostType=28 hostGroup="de4801_nsds";
create host userLabel="ece4710" hostType=28 hostGroup="de4801_nsds";
create host userLabel="ece4711" hostType=28 hostGroup="de4801_nsds";
create host userLabel="ece4712" hostType=28 hostGroup="de4801_nsds";

show "Creating NVMe-over-Fabrics initiators...";
create initiator identifier="nqn.2024-09.com.lenovo.eu.hpc:nvme:ece4709"
  userLabel="ece4709-iqn" host="ece4709" interfaceType=nvmeof;
create initiator identifier="nqn.2024-09.com.lenovo.eu.hpc:nvme:ece4710"
  userLabel="ece4710-iqn" host="ece4710" interfaceType=nvmeof;
create initiator identifier="nqn.2024-09.com.lenovo.eu.hpc:nvme:ece4711"
  userLabel="ece4711-iqn" host="ece4711" interfaceType=nvmeof;
create initiator identifier="nqn.2024-09.com.lenovo.eu.hpc:nvme:ece4712"
  userLabel="ece4712-iqn" host="ece4712" interfaceType=nvmeof;

show "Creating volume mappings...";
set volume ["de4801r6a01v01"] logicalUnitNumber=1  hostGroup="de4801_nsds";
set volume ["de4801r6a01v02"] logicalUnitNumber=2  hostGroup="de4801_nsds";
set volume ["de4801r6a02v01"] logicalUnitNumber=3  hostGroup="de4801_nsds";
set volume ["de4801r6a02v02"] logicalUnitNumber=4  hostGroup="de4801_nsds";
set volume ["de4801r1a01v01"] logicalUnitNumber=5  hostGroup="de4801_nsds";
set volume ["de4801r1a02v01"] logicalUnitNumber=6  hostGroup="de4801_nsds";

show "Done.";
```

Applied in a similar way as before with SMcli:

```
$ ./SMcli/SMcli-01.81.54.9004/bin/SMcli de4904a-mgt -u admin -P .depass  -k -f
    de4904-hosts.bat
Performing syntax check...

Syntax check complete.
```



```
Executing script...

Creating hostgroup...
Creating hosts...
Creating NVMe-over-Fabrics initiators...
Creating volume mappings...
Done.

Script execution complete.

SMcli completed successfully.
```

5 Setup the OSS servers

In this section we look at configuring the Infiniband hosts to allow for RDMA over IB communication.

5.1 Install IB Drivers

As part of Lenovo EveryScale clusters, Lenovo validates components and software drivers as part of an EveryScale cluster. This includes compatibility of NVIDIA OFED software. Refer to the Lenovo Support Site[10] for details of the latest Lenovo EveryScale Best Recipe.

We build and install the OFED stack including NVMe over Fabric support:

```
$ ./mlnxofedinstall --add-kernel-support --with-nvmf
Note: This program will create MLNX_OFED_LINUX TGZ for rhel9.4 under /tmp/
   MLNX_OFED_LINUX-24.07-0.6.1.0-5.14.0-427.
33.1.el9_4.x86_64 directory.
See log file /tmp/MLNX_OFED_LINUX-24.07-0.6.1.0-5.14.0-427.33.1.el9_4.x86_64/
   mlnx_iso.350131_logs/mlnx_ofed_iso.350131.log

Checking if all needed packages are installed...
Building MLNX_OFED_LINUX RPMS . Please wait...
...
You may need to update your initramfs before next boot. To do that, run:

   dracut -f
To load the new driver, run:
/etc/init.d/openibd restart
```

As instructed, we rebuild the initramfs:

```
dracut -f
```

Next we add the package for NVMe command line interface if not already installed:

```
# dnf -y install nvme-cli
Last metadata expiration check: 1:49:00 ago on Tue 03 Sep 2024 10:48:40 AM CEST.
Package nvme-cli-2.6-5.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

We can then check the mapping between the physical ports on the IB adapters with the network names.

```
$ systemctl restart openibd
$ ibdev2netdev
mlx5_0 port 1 ==> ibs2 (Up)
mlx5_1 port 1 ==> ibs1 (Up)
mlx5_2 port 1 ==> ibs3 (Up)
```

[10] https://support.lenovo.com/us/en/solutions/HT510136

5.2 Configuring networking for the IB interfaces

On our benchmark system that we are using for this testing, the interfaces are configured using Lenovo Confluent[11]. From the Confluent management host we can check the interfaces with:

```
$ nodeattrib ece4710 | grep net
ece4710: net.infiniband.interface_names: ibs1
ece4710: net.infiniband.ipv4_address: 172.30.119.70/20
ece4710: net.infiniband1.interface_names: ibs2
ece4710: net.infiniband1.ipv4_address: 172.30.119.90/20
ece4710: net.infiniband2.interface_names: ibs3
ece4710: net.infiniband2.ipv4_address: 172.30.119.110/20
```

If the address are not set, you can use for example:

```
$ nodeattrib ece4710 net.infiniband.interface_names=ibs1 net.infiniband.
    ipv4_address=172.30.119.70/20
ece4710: ibs1
ece4710: 172.30.119.70/20
```

Documentation for the confluent command line interface is available online[12]. On the hosts, we then reload configure these connections we set in confluent by running a script from confluent:

```
$ sh /etc/confluent/functions run_remote confignet

--------------------------------------------------------------------------------
Running 'confignet' from https://172.30.86.1/confluent-public/os/rocky-9.
    4-x86_64-LENOX/scripts/
Executing in /tmp/confluentscripts.PQ4JhtYR8
. firewalld.service - firewalld - dynamic firewall daemon
     Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; preset:
         enabled)
     Active: inactive (dead) since Tue 2024-09-03 12:40:35 CEST; 48s ago
   Duration: 18h 10min 41.393s
       Docs: man:firewalld(1)
    Process: 1903 ExecStart=/usr/sbin/firewalld --nofork --nopid $FIREWALLD_ARGS
         (code=exited, status=0/SUCCESS)
   Main PID: 1903 (code=exited, status=0/SUCCESS)
        CPU: 269ms

Sep 02 18:29:53 ece4710 systemd[1]: Starting firewalld - dynamic firewall daemon
    ...
Sep 02 18:29:53 ece4710 systemd[1]: Started firewalld - dynamic firewall daemon.
Sep 03 12:40:35 ece4710 systemd[1]: Stopping firewalld - dynamic firewall daemon
    ...
Sep 03 12:40:35 ece4710 systemd[1]: firewalld.service: Deactivated successfully.
Sep 03 12:40:35 ece4710 systemd[1]: Stopped firewalld - dynamic firewall daemon.
Connection successfully activated (D-Bus active path: /org/freedesktop/
    NetworkManager/ActiveConnection/8)
Connection successfully activated (D-Bus active path: /org/freedesktop/
    NetworkManager/ActiveConnection/9)
Connection successfully activated (D-Bus active path: /org/freedesktop/
    NetworkManager/ActiveConnection/10)
Connection successfully activated (D-Bus active path: /org/freedesktop/
    NetworkManager/ActiveConnection/11)
```

As the host has three IP addresses across three IB interfaces, we need to ensure that source based policy routing is configured. This ensures that only the interface that has the IP address assigned responds to ARP requests and ensures only that interface is used for traffic sourced from the IP address assigned.

[11] https://lenovopress.lenovo.com/lp1651-lenovo-hpc-ai-software-stack
[12] https://hpc.lenovo.com/users/documentation/

5.2.1 Setting up source policy routing

A Confluent script may be available which automatically configures source policy routing:

```
$ sh /etc/confluent/functions run_remote netdev-source-policy.sh
--------------------------------------------------------------------------------
Running 'netdev-source-policy.sh' from https://172.30.86.1/confluent-public/os/
    rocky-9.4-x86_64-LENOX/scripts/
Executing in /tmp/confluentscripts.XX0fMbPNo
Skip 169.254.95.0/24
Skip 172.30.80.0/20
Skip 127.0.0.0/8
172.30.112.0/20: Found 3 instances
ens4f0np0 - 172.30.87.70/20
 -> Not correct segment - SKIP
ibs1 - 172.30.119.70/20
Connection successfully activated (D-Bus active path: /org/freedesktop/
    NetworkManager/ActiveConnection/12)
net.ipv4.conf.ibs1.arp_filter = 1
ibs2 - 172.30.119.90/20
Connection successfully activated (D-Bus active path: /org/freedesktop/
    NetworkManager/ActiveConnection/13)
net.ipv4.conf.ibs2.arp_filter = 1
ibs3 - 172.30.119.110/20
Connection successfully activated (D-Bus active path: /org/freedesktop/
    NetworkManager/ActiveConnection/14)
net.ipv4.conf.ibs3.arp_filter = 1
lo - 127.0.0.1/8
 -> Not correct segment - SKIP
net.ipv4.conf.all.arp_filter = 1
net.ipv4.conf.default.arp_filter = 1
```

If not using Lenovo Confluent to manage the hosts, you will need to configure the IPoIB interfaces manually and setup source policy routing manually. Create a config file `/etc/sysctl.d/rpfilter.conf` with contents:

```
net.ipv4.conf.ibs1.arp_filter=1
net.ipv4.conf.ibs2.arp_filter=1
net.ipv4.conf.ibs3.arp_filter=1
net.ipv4.conf.all.arp_filter=1
net.ipv4.conf.default.arp_filter=1
```

There should be an entry for each IPoIB interface, these settings disable the Linux kernel from responding to ARP requests on L2 connected devices where the IP address is not assigned to the device, otherwise for example interface `ibs1` might respond to arp requests for a L2 connected interface on `ibs2`.

For each IPoIB interface, it is necessary to create some networking routing rules, for example:

```
nmcli con mod ibs1 ipv4.routing-rules "priority 501 iif ibs1 table 201"  #
    codespell:ignore iif
nmcli con mod ibs1 +ipv4.routing-rules "priority 501 from 172.30.119.70 table
    201"
nmcli con mod ibs1 ipv4.routes "172.30.112.0/20 table=201"

nmcli con mod ibs2 ipv4.routing-rules "priority 502 iif ibs2 table 202"  #
    codespell:ignore iif
nmcli con mod ibs2 +ipv4.routing-rules "priority 502 from 172.30.119.90 table
    202"
nmcli con mod ibs2 ipv4.routes "172.30.112.0/20 table=202"

nmcli con mod ibs3 ipv4.routing-rules "priority 503 iif ibs3 table 203"  #
    codespell:ignore iif
nmcli con mod ibs3 +ipv4.routing-rules "priority 503 from 172.30.119.110 table
    203"
```



```
nmcli con mod ibs3 ipv4.routes "172.30.112.0/20 table=203"
```

Note that the table number and priority needs to be different for each IB interface The interface should then be up/downed. We can then check the routing:

```
$ ip rule list
0:      from all lookup local
501:    from all iif ibs1 lookup 201 proto static  # codespell:ignore iif
501:    from 172.30.119.70 lookup 201 proto static
502:    from all iif ibs2 lookup 202 proto static  # codespell:ignore iif
502:    from 172.30.119.90 lookup 202 proto static
503:    from all iif ibs3 lookup 203 proto static  # codespell:ignore iif
503:    from 172.30.119.110 lookup 203 proto static
32766:  from all lookup main
32767:  from all lookup default
```

```
$ ip route show table 201
172.30.112.0/20 dev ibs1 proto static scope link metric 151
```

These rules apply source policy routing to ensure that only traffic sourced from the IP address on the interface goes out of the interface it is assigned to - otherwise the Linux kernel might send traffic from an IP out of another interface where L2 connected networks are configured.

5.3 Checking the installation

Check that the `nvme_core` module is loaded from the OFED installation:

```
$ modinfo nvme_core | grep filename
filename:       /lib/modules/5.14.0-427.33.1.el9_4.x86_64/extra/mlnx-nvme/host/
    nvme-core.ko
```

It is important that the `nvme-rdma` module is also loaded, check that it is coming from the OFED installation:

```
$ modinfo nvme_rdma | grep filename
filename:       /lib/modules/5.14.0-427.33.1.el9_4.x86_64/extra/mlnx-nvme/host/
    nvme-rdma.ko
```

And also configure it to load at boot time and load the module:

```
$ echo "nvme-rdma" >> /etc/modules-load.d/nvme-rdma.conf
$ modprobe nvme-rdma
$ lsmod | grep nvme_rdma
nvme_rdma              57344  0
nvme_fabrics           45056  1 nvme_rdma
rdma_cm               155648  2 nvme_rdma,rdma_ucm
ib_core               548864  9 rdma_cm,ib_ipoib,nvme_rdma,iw_cm,ib_umad,
    rdma_ucm,ib_uverbs,mlx5_ib,ib_cm
nvme_core             196608  7 nvme,nvme_rdma,nvme_fabrics
mlx_compat             20480  15 rdma_cm,ib_ipoib,mlxdevm,nvme,nvme_rdma,iw_cm,
    nvme_core,nvme_fabrics,ib_umad,ib_core,rdma_ucm,ib_uverbs,mlx5_ib,ib_cm,
    mlx5_core
```

At this point, it is advised to reboot the client host to ensure the MOFED nvme drivers are correctly loaded. Configure the host nqn on the node. Note that by default the nqn will have a UUID which might change if the host is redeployed (e.g. upgrading the OS or on a diskless node).

```
$ nvme show-hostnqn
nqn.2014-08.org.nvmexpress:uuid:db5b7106-2082-11ed-80ee-3a68dd8438cb
$ echo "nqn.2024-09.com.lenovo.eu.hpc:nvme:$(hostname -s)" > /etc/nvme/hostnqn
$ nvme show-hostnqn
nqn.2024-09.com.lenovo.eu.hpc:nvme:ece4710
```


5.4 Tuning disk parameters

It is necessary to tune some parameters with the IO subsystem, this is accomplished using udev rules, for example, create the file `/etc/udev/rules.d/99-nvme-disk-lenovo-de.rules`.

```
# Volumes from the DE

   ACTION=="add|change", SUBSYSTEM=="block", KERNEL=="nvme[0-9]*", ATTRS{model
   }=="LENOVO DE-Series*", \                SUBSYSTEMS=="nvme-subsystem", \
  ATTR{queue/max_sectors_kb}="1024", \
  ATTR{queue/read_ahead_kb}="0", \
  ATTR{queue/add_random}="0", \
  ATTR{queue/nomerges}="0", \
  ATTR{device/iopolicy}="round-robin"
```

To help with debugging rules, you could use for example the command `udevadm info -a /dev/disk/by-id/nvme-LENOVO_DE-Series_J7018733` which will show specific device information and the parent ATTRS which can be used to target the udev rules.

Once the udev rule is in place, trigger a udev rule update:

```
udevadm control --reload-rules
udevadm trigger --type=devices --action=change
```

You would then check the values of e.g. `read_ahead_kb`:

```
$ cd /sys/block/nvme12n2/queue
$ cat read_ahead_kb
0
```

6 Setup multipathing

There are two methods of configuring a DE6600F with NVMeoF to support multipathing:

- Native NVMe Multipathing
- device-multipath multipathing (dm-multipath)

In general with Red Hat Enterprise Linux 9, it is recommended to use Native NVMe multipathing.

Where the IO server has other storage attached (e.g. SAS attached DE6000), then it is recommended to use native NVMe multipathing to connect to the DE6600F and dm-multipath to connect to the DE6000. Whilst it is possible to use dm-multipath for both storage arrays, it is more complicated to configure dm-multipath to ensure rules apply specifically to the DE6000 or DE6600F storage arrays.

6.1 Using native NVMe multipathing

The OFED driver should enable native NVMe multipathing by default, we can check with:

```
$ cat /sys/module/nvme_core/parameters/multipath
Y
```

If using not enabled, we can update the grub arguments with:

```
grubby --update-kernel=ALL --args="nvme_core.multipath=Y"
```

Create `/etc/modprobe.d/nvme_core.conf` with contents:

```
options nvme_core multipath=Y
```

And finally run:

```
dracut -f
```


6.2 Discover volumes

We use nvme-cli to discover nvme connected targets on the network. Here we show an example with second DE6600F system (configured with hostname de4904):

```
# nvme discover -t rdma -a 172.30.121.59

Discovery Log Number of Records 4, Generation counter 0
=====Discovery Log Entry 0======
trtype:  rdma
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  0
trsvcid: 4420
subnqn:  nqn.1992-08.com.netapp:6000.6d039ea0005745cc0000000065b1a166
traddr:  172.30.121.59
eflags:  none
rdma_prtype: infiniband
rdma_qptype: connected
rdma_cms:    rdma-cm
rdma_pkey: 0x0000
=====Discovery Log Entry 1======
trtype:  rdma
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  1
trsvcid: 4420
subnqn:  nqn.1992-08.com.netapp:6000.6d039ea0005745cc0000000065b1a166
traddr:  172.30.128.22
eflags:  none
rdma_prtype: infiniband
rdma_qptype: connected
rdma_cms:    rdma-cm
rdma_pkey: 0x0000
...
```

To connect using a different source address, use one of the following:

```
nvme discover -t rdma -a 172.30.121.59 -w 172.30.119.90
nvme discover --transport rdma --traddr 172.30.121.59 --host-traddr 172.30.119.
   90
```

We can also attach using a single path to differentiate the subnets:

```
nvme connect --transport rdma --nqn nqn.1992-08.com.netapp:6000.
   6d039ea0005745cc0000000065b1a166 \
--traddr 172.30.121.59 --host-traddr 172.30.119.90 --queue-size 1024
   --ctrl-loss-tmo 3600
nvme connect --transport rdma --nqn nqn.1992-08.com.netapp:6000.
   6d039ea0005745cc0000000065b1a166 \
--traddr 172.30.128.22 --host-traddr 172.30.128.10 --queue-size 1024
   --ctrl-loss-tmo 3600
```

Once we have done this for all the IPoIB connections on the DE6600F systems, we can check using nvme cli that the devices are connected:

```
# nvme list
Node                  Generic               SN                   Model
                                       Namespace  Usage
   Format           FW Rev
```



```
--------------------- --------------------- --------------------
   ---------------------------------------- ----------
   ------------------------- ---------------- --------
/dev/nvme11n1         /dev/ng11n1           J7018733              LENOVO
   DE-Series                          0x1         58.97  TB /  58.97  TB      4
   KiB +  0 B   08810100
/dev/nvme11n2         /dev/ng11n2           J7018733              LENOVO
   DE-Series                          0x2         58.97  TB /  58.97  TB      4
   KiB +  0 B   08810100
/dev/nvme11n3         /dev/ng11n3           J7018733              LENOVO
   DE-Series                          0x3         58.97  TB /  58.97  TB      4
   KiB +  0 B   08810100
/dev/nvme11n4         /dev/ng11n4           J7018733              LENOVO
   DE-Series                          0x4         58.97  TB /  58.97  TB      4
   KiB +  0 B   08810100
/dev/nvme11n5         /dev/ng11n5           J7018733              LENOVO
   DE-Series                          0x5         14.74  TB /  14.74  TB      4
   KiB +  0 B   08810100
/dev/nvme11n6         /dev/ng11n6           J7018733              LENOVO
   DE-Series                          0x6         14.74  TB /  14.74  TB      4
   KiB +  0 B   08810100
```

Note that if the system has local NVMe drives, you may see these also in the output.

To check that the native multipath is working, use:

```
$ nvme list-subsys
...
nvme-subsys15 - NQN=nqn.1992-08.com.netapp:6000.6d039ea0005745cc0000000065b1a166
                hostnqn=nqn.2024-09.com.lenovo.eu.hpc:nvme:ece4710
                iopolicy=round-robin
\
 +- nvme18 rdma traddr=172.30.121.59,trsvcid=4420,host_traddr=172.30.119.90 live
 +- nvme17 rdma traddr=172.30.121.63,trsvcid=4420,host_traddr=172.30.119.90 live
 +- nvme16 rdma traddr=172.30.128.23,trsvcid=4420,host_traddr=172.30.128.10 live
 +- nvme15 rdma traddr=172.30.128.22,trsvcid=4420,host_traddr=172.30.128.10 live
...
```

Note that in this case, we see 4 paths to the NVMe subsystem with the traddr pointing to each of the IPoIB interfaces on the DE6600F system. These 4 paths correspond to the 4 IPoIB addresses (2 per canister) in the DE6600F controller. In this case, the DE6600F is configured with FC BUY7 (Lenovo ThinkSystem DE6600 HIC, 200GB NVMe Over ROCE/IB,2-Ports). Where the system is configured with FC BS3G (Lenovo ThinkSystem DE6400/6600 HIC, 100GB NVMe Over ROCE,2-Ports) and has 4 HICs installed, the system would be configured with 8 IPoIB addresses (4 per cannsiter).

Note: There is currently a limitation that the DE6600F can only be configured with 2 BS3G 100Gb IB HICs).

Whilst the NSD server has three IB interfaces, it is not possible to attach additional paths from the host to the DE6600F by using different `--host-traddr` addresses, trying to do this results in an error such as:

```
Failed to write to /dev/nvme-fabrics: Input/output error
```

i.e. it is not possible to use the NSD server's IPoIB address 172.30.119.70 to provide an additional 5th path to the DE6600F storage array.

Once the paths are connected, it is important to check that the transport protocol is correctly set to RDMA:

```
$ cat /sys/class/nvme/nvme11/transport
rdma
```

If `rdma` is not presented, ensure that the kernel module `nvme_rdma` is proprerly loaded and that it is coming from the installed OFED packages rather than an inbox kernel driver.

Native NVMe multipathing currently supports two IO policies, i.e. how IOs are sent to the device. These are `numa` and `round-robin`. By default, the policy is likely `numa` which is correct in the case of local NVMe devices, however for the DE6600F, `round-robin` is preferred.

Checking the policy:

```
$ cat /sys/class/nvme-subsystem/nvme-subsys11/iopolicy
numa
```

When using round-robin, the native NVMe multipathing should still use Asstemtric Namespace Access (ANA), this is similar to ALUA with SCSI systems and should ensure that optimised IO paths are used for data. This is important when using a DE6600F as each canister will be the "owner" of a volume and hence the preferred IO path. Note that even when an IO is sent to the non-owner, the DE6600F passes the IO internally between canisters.

To make udev automatically set the device to round-robin, create the file

`/etc/udev/rules.d/71-nvme-io-policy-lenovo-de.rules`

```
SUBSYSTEM!="nvme-subsystem", GOTO="end_rule"
ACTION!="add|change", GOTO="end_rule"
ATTRS{subsystype}!="nvm", GOTO="end_rule"

# skip if not a DE series
ATTRS{model}!="*DE-Series*", GOTO="end_rule"

# Change the io policy
ATTR{iopolicy}="round-robin"

LABEL="end_rule"
```

To apply run:

```
$ udevadm control --reload-rules
$ udevadm trigger --type=devices --action=change
$ cat /sys/class/nvme-subsystem/nvme-subsys11/iopolicy
round-robin
```

Check other local devices, for example:

```
$ cat /sys/class/nvme-subsystem/nvme-subsys1/iopolicy
numa
```

To automatically be able to use `nvme connect-all`, create `/etc/nvme/discovery.conf`:

```
# Used for extracting default parameters for discovery
#
# Example:
# --transport=<trtype> --traddr=<traddr> --trsvcid=<trsvcid> --host-traddr=<
    host-traddr> --host-iface=<host-iface>

--transport rdma --traddr 172.30.121.59 --host-traddr 172.30.119.90 --queue-size
     1024 --ctrl-loss-tmo 3600
--transport rdma --traddr 172.30.128.22 --host-traddr 172.30.128.10 --queue-size
     1024 --ctrl-loss-tmo 3600
--transport rdma --traddr 172.30.121.63 --host-traddr 172.30.119.90 --queue-size
     1024 --ctrl-loss-tmo 3600
--transport rdma --traddr 172.30.128.23 --host-traddr 172.30.128.10 --queue-size
     1024 --ctrl-loss-tmo 3600
```

Enable and start the systemd service to connect the devices on each boot:


```
$ systemctl enable nvmf-autoconnect.service
Created symlink /etc/systemd/system/default.target.wants/nvmf-autoconnect.
   service → /usr/lib/systemd/system/nvmf-autoconnect.service.
$ systemctl start nvmf-autoconnect.service
```

6.3 Using dm-multipath

As an alternative to using native NVMe multipathing, it is also possible to use device-mapper multipath (dm-multipath). Whilst on paper peak performance may be lower than using native NVMe multipathing, in practice it is likely to make little difference with the DE6600F system.

Reverting the options we highlighted at the start of the section:

```
$ grubby --update-kernel=ALL --args="nvme_core.multipath=N"
$ reboot
$ cat /sys/module/nvme_core/parameters/multipath
N

$ dnf -y install device-mapper-multipath
```

We then modify the dm multipath configuration file located at `/etc/multipath.conf`:

```
defaults {
        user_friendly_names yes
        find_multipaths yes
}

blacklist {
}

#
# Lenovo DE6600 series NVMeoF devices
#
devices {
        device {
                vendor                  "NVME"
                product                 "LENOVO DE-Series"
                path_grouping_policy    "group_by_prio"
                # path_selector         "queue-length 0"
                failback                "immediate"
                no_path_retry           30
        }
}
```

Once we have this, we can enable the dm multipath service on the host:

```
systemctl enable --now multipathd.service
multipath -r
```

Then we can check the paths are all showing correctly with:

```
$ multipath -ll
...
mpathp (eui.0000044e66e47007d039ea00005834c0) dm-41 NVME,LENOVO DE-Series
size=14T features='1 queue_if_no_path' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=50 status=active
| |- 16:32770:6:6 nvme16n6 259:83  active ready running
| `- 17:32771:6:6 nvme17n6 259:93  active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  |- 15:2:6:6     nvme15n6 259:73  active ready running
  `- 18:3:6:6     nvme18n6 259:103 active ready running
```



```
...
```

Note two paths with different priority - 50 will be the “active” controller for the volume and then 1 target IP per IPoIB interface.

Note that in `nvme list`, that each volume/controller will appear multiple times when not using NVMe native multipathing:

```
[root@ece4710 ~]# nvme list
Node                  Generic               SN                   Model
                                          Namespace  Usage
   Format           FW Rev
--------------------- --------------------- --------------------
   ---------------------------------------- ----------
   -------------------------- ---------------- --------
/dev/nvme0n1          /dev/ng0n1            PHAB209600VJ3P8EGN   SSDPF2KE032T9L
                             0x1          3.20  TB /   3.20  TB      4 KiB +  0
   B   2CV1L031
/dev/nvme10n1         /dev/ng10n1           PHAB1525025V3P8EGN   SSDPF2KE032T9L
                             0x1          3.20  TB /   3.20  TB      4 KiB +  0
   B   2CV1L031
/dev/nvme12n1         /dev/ng12n1             J7018733             LENOVO
   DE-Series                          0x1         61.43  TB /  61.43  TB      4
   KiB +  0 B   08810200
/dev/nvme12n2         /dev/ng12n2             J7018733             LENOVO
   DE-Series                          0x2         61.43  TB /  61.43  TB      4
   KiB +  0 B   08810200
/dev/nvme12n3         /dev/ng12n3             J7018733             LENOVO
   DE-Series                          0x3         61.43  TB /  61.43  TB      4
   KiB +  0 B   08810200
/dev/nvme12n4         /dev/ng12n4             J7018733             LENOVO
   DE-Series                          0x4         61.43  TB /  61.43  TB      4
   KiB +  0 B   08810200
/dev/nvme12n5         /dev/ng12n5             J7018733             LENOVO
   DE-Series                          0x5         15.36  TB /  15.36  TB      4
   KiB +  0 B   08810200
/dev/nvme12n6         /dev/ng12n6             J7018733             LENOVO
   DE-Series                          0x6         15.36  TB /  15.36  TB      4
   KiB +  0 B   08810200
/dev/nvme13n1         /dev/ng13n1             J7018733             LENOVO
   DE-Series                          0x1         61.43  TB /  61.43  TB      4
   KiB +  0 B   08810200
/dev/nvme13n2         /dev/ng13n2             J7018733             LENOVO
   DE-Series                          0x2         61.43  TB /  61.43  TB      4
   KiB +  0 B   08810200
/dev/nvme13n3         /dev/ng13n3             J7018733             LENOVO
   DE-Series                          0x3         61.43  TB /  61.43  TB      4
   KiB +  0 B   08810200
/dev/nvme13n4         /dev/ng13n4             J7018733             LENOVO
   DE-Series                          0x4         61.43  TB /  61.43  TB      4
   KiB +  0 B   08810200
/dev/nvme13n5         /dev/ng13n5             J7018733             LENOVO
   DE-Series                          0x5         15.36  TB /  15.36  TB      4
   KiB +  0 B   08810200
/dev/nvme13n6         /dev/ng13n6             J7018733             LENOVO
   DE-Series                          0x6         15.36  TB /  15.36  TB      4
   KiB +  0 B   08810200
/dev/nvme14n1         /dev/ng14n1             J7018733             LENOVO
   DE-Series                          0x1         61.43  TB /  61.43  TB      4
   KiB +  0 B   08810200
/dev/nvme14n2         /dev/ng14n2             J7018733             LENOVO
   DE-Series                          0x2         61.43  TB /  61.43  TB      4
```



```
   KiB +  0 B   08810200
/dev/nvme14n3         /dev/ng14n3           J7018733             LENOVO
   DE-Series                        0x3         61.43  TB /  61.43  TB      4
   KiB +  0 B   08810200
/dev/nvme14n4         /dev/ng14n4           J7018733             LENOVO
   DE-Series                        0x4         61.43  TB /  61.43  TB      4
   KiB +  0 B   08810200
```

6.4 Validation of Setup

In order to validate our devices are setup correctly, we can run some performance tests using FIO[13]. On the system `ece4709` we can download and run the FIO benchmark, writing to the dm multipath devices located in `/dev/mapper`:

```
$ wget https://github.com/axboe/fio/archive/refs/tags/fio-3.38.tar.gz
$ tar -xvzf fio-3.38.tar.gz
$ cd fio-fio-3.38
$ ./configure
$ make
$ fio --filename=/dev/mapper/mpathr:/dev/mapper/mpathq:/dev/mapper/mpaths:/dev/
   mapper/mpatht --direct=1  --bs=128k --ioengine=libaio --iodepth=512 --runtime
   =120 --numjobs=16 --time_based --group_reporting --name=throughput-test-job
   --eta-newline=1 --rw=read / write

...
Write: WRITE: bw=14.8GiB/s (15.9GB/s), 14.8GiB/s-14.8GiB/s (15.9GB/s-15.9GB/s),
   io=2666GiB (2863GB), run=180008-180008msec
Read: READ: bw=22.5GiB/s (24.2GB/s), 22.5GiB/s-22.5GiB/s (24.2GB/s-24.2GB/s), io
   =2707GiB (2907GB), run=120189-120189msec
```

7 BeeGFS Installation

In this section we will outline the steps required to setup BeeGFS to make use of the DE6600F as an IO controller.

7.1 Partitioning and Formatting of Volumes

Before we proceed with installing the BeeGFS services, we first need to partition and install a filesystem on the storage devices. On our test servers we have device multipath enabled, so we can list available multipath devices with:

```
$ multipath -ll
...
de4801r1a01v01 (eui.000001ec66ded794d039ea00005501c3) dm-26 NVME,LENOVO
   DE-Series
size=14T features='0' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=50 status=active
| |- 16:32772:6:5 nvme16n6 259:58 active ready running
| |- 17:32775:6:5 nvme17n6 259:69 active ready running
| |- 18:32776:6:5 nvme18n6 259:75 active ready running
| `- 19:32777:6:5 nvme19n6 259:86 active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  |- 12:2:6:5     nvme12n6 259:27 active ready running
  |- 13:3:6:5     nvme13n6 259:36 active ready running
  |- 14:6:6:5     nvme14n6 259:42 active ready running
  `- 15:9:6:5     nvme15n6 259:51 active ready running
de4801r1a02v01 (eui.000001ee66ded798d039ea00005501c3) dm-27 NVME,LENOVO
   DE-Series
```

[13] https://github.com/axboe/fio/releases/tag/fio-3.38


```
size=14T features='0' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=50 status=enabled
| |- 12:2:4:6     nvme12n4 259:25 active ready running
| |- 13:3:4:6     nvme13n4 259:34 active ready running
| |- 14:6:4:6     nvme14n4 259:43 active ready running
| `- 15:9:4:6     nvme15n4 259:53 active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  |- 16:32772:4:6 nvme16n4 259:61 active ready running
  |- 17:32775:4:6 nvme17n4 259:65 active ready running
  |- 18:32776:4:6 nvme18n4 259:77 active ready running
  `- 19:32777:4:6 nvme19n4 259:83 active ready running
mpathq (eui.000007686903aac2d039ea00005501c3) dm-22 NVME,LENOVO DE-Series
size=56T features='0' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=50 status=active
| |- 12:2:2:1     nvme12n2 259:26 active ready running
| |- 13:3:2:1     nvme13n2 259:33 active ready running
| |- 14:6:2:1     nvme14n2 259:40 active ready running
| `- 15:9:2:1     nvme15n2 259:50 active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  |- 16:32772:2:1 nvme16n2 259:59 active ready running
  |- 17:32775:2:1 nvme17n2 259:64 active ready running
  |- 18:32776:2:1 nvme18n2 259:72 active ready running
  `- 19:32777:2:1 nvme19n2 259:80 active ready running
mpathr (eui.000003a76903aa74d039ea00005501b5) dm-24 NVME,LENOVO DE-Series
size=56T features='0' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=50 status=active
| |- 16:32772:1:4 nvme16n1 259:57 active ready running
| |- 17:32775:1:4 nvme17n1 259:70 active ready running
| |- 18:32776:1:4 nvme18n1 259:76 active ready running
| `- 19:32777:1:4 nvme19n1 259:82 active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  |- 12:2:1:4     nvme12n1 259:24 active ready running
  |- 13:3:1:4     nvme13n1 259:32 active ready running
  |- 14:6:1:4     nvme14n1 259:41 active ready running
  `- 15:9:1:4     nvme15n1 259:52 active ready running
mpaths (eui.000007696903ac26d039ea00005501c3) dm-23 NVME,LENOVO DE-Series
size=56T features='0' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=50 status=active
| |- 12:2:3:3     nvme12n3 259:28 active ready running
| |- 13:3:3:3     nvme13n3 259:37 active ready running
| |- 14:6:3:3     nvme14n3 259:47 active ready running
| `- 15:9:3:3     nvme15n3 259:49 active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  |- 16:32772:3:3 nvme16n3 259:56 active ready running
  |- 17:32775:3:3 nvme17n3 259:68 active ready running
  |- 18:32776:3:3 nvme18n3 259:73 active ready running
  `- 19:32777:3:3 nvme19n3 259:84 active ready running
mpatht (eui.000003a96903ac61d039ea00005501b5) dm-25 NVME,LENOVO DE-Series
size=56T features='0' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=50 status=active
| |- 16:32772:5:2 nvme16n5 259:62 active ready running
| |- 17:32775:5:2 nvme17n5 259:67 active ready running
| |- 18:32776:5:2 nvme18n5 259:74 active ready running
| `- 19:32777:5:2 nvme19n5 259:81 active ready running
`-+- policy='round-robin 0' prio=10 status=enabled
  |- 12:2:5:2     nvme12n5 259:29 active ready running
  |- 13:3:5:2     nvme13n5 259:39 active ready running
  |- 14:6:5:2     nvme14n5 259:45 active ready running
  `- 15:9:5:2     nvme15n5 259:48 active ready running
...
```


To summarise the above output, we have dm devices of 56T with multipath mapping `mpathq-mpatht` corresponding to our two sets of 10 drives in RAID 8+2P configuration that we intend to use for data. We then have 2 devices of 14T capacity that we will use for file metadata. Each device has a multipath name `/dev/mapper/X` and an associated multipath entry for this server `/dev/dm-X`. Attempting to proceed with formatting commands on the latter will result in "Device or resource busy" errors as these are controlled by the dm multipath service.

We can start by creating partition tables for the devices if one does not exist already:

```
[root@ece4709 ~]# fdisk /dev/mapper/mpathq1
Welcome to fdisk (util-linux 2.37.4).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
The size of this disk is 55.9 TiB (61429843443712 bytes). DOS partition table
    format cannot be used on drives for volumes larger than

Created a new DOS disklabel with disk identifier 0x5965f6e1.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (256-4294967295, default 256):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (256-4294967295, default
    4294967295):

Created a new partition 1 of type 'Linux' and of size 16 TiB.
```

If we want to use a partition table, we can create an xfs on top of that device. Alternatively we can build a fs on thee device directly. We do this on each of the 4 devices intended for storage:

```
mkfs.xfs -d su=32k,sw=8 -l version=2,su=32k /dev/mapper/mpatht -f
```

Here it was important to align the filesystem block size with the RAID configuration. In this case we use a block size of 32k to match what we have set in the RAID stripe size and using `sw=8` since there will be 8 devices with non-parity data on an 8+2P RAID setup.

Depending on the filesystem block size used, this command above took between 8 and 30 mins in our setup. Once we have done this for all 4 devices, we can mount the created filesystems:

```
[root@ece4709 ~]# mkdir /mnt/de4904-1
[root@ece4709 ~]# mkdir /mnt/de4904-2
[root@ece4710 ~]# mkdir /mnt/de4904-3
[root@ece4710 ~]# mkdir /mnt/de4904-4
[root@ece4709 ~]# mount /dev/mapper/mpathq /mnt/de4904-1
[root@ece4709 ~]# mount /dev/mapper/mpatht /mnt/de4904-2
[root@ece4710 ~]# mount /dev/mapper/mpathl /mnt/de4904-3
[root@ece4710 ~]# mount /dev/mapper/mpathm /mnt/de4904-4
[root@ece4709 ~]# df
Filesystem                     1K-blocks      Used   Available Use% Mounted on
...
/dev/mapper/mpatht             59987994624  373469368  59614525256   1% /mnt/
    de4801-2
/dev/mapper/mpathq             59987994624  373469240  59614525384   1% /mnt/
    de4801-1
...
[root@ece4710 ~]# df
Filesystem                     1K-blocks      Used   Available Use% Mounted on
...
```



```
/dev/mapper/mpathm              59987994624   373469240   59614525384    1% /mnt/
    de4801-4
/dev/mapper/mpathl              59987994624   373469240   59614525384    1% /mnt/
    de4801-3
```

Where in the last lines we have used `df` to ensure the created filesystems are mounted correctly. To ensure these mount on each boot, we label each partition using:

```
[root@ece4709 ~]# xfs_admin -L "de4904-1" /dev/mapper/mpathq
label = "de4904-1"
[root@ece4709 ~]# xfs_admin -L "de4904-2" /dev/mapper/mpatht
label = "de4904-2"
```

And then add entries to the `/etc/fstab` file on each of the storage servers of the form:

```
LABEL=de4904-1     /mnt/de4904-1    xfs    defaults 0 0
```

On each mounted filesystem we create a directory `beegfs` to be used for storing BeeGFS configuration data and an empty file which is different for each filesystem as a sanity check to make sure the filesystems are the same across the storage nodes.

```
[root@ece4709 ~]# cd /mnt/de4904-2/
[root@ece4709 de4904-4]# mkdir beegfs
[root@ece4709 de4904-4]# touch two.txt
```

Once this is complete, we do a similar setup with the devices intended for metadata. The BeeGFS documentation recommends that metadata filesystems be created as `ext4` filesystems[14] as they "handles small file workloads (common on a BeeGFS metadata server) significantly faster than other local Linux file systems".

```
[root@ece4709 ~]# fdisk /dev/mapper/de4801r1a01v01

Welcome to fdisk (util-linux 2.37.4).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0xe68a2059.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (256-3749183487, default 256):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (256-3749183487, default
    3749183487):

Created a new partition 1 of type 'Linux' and of size 14 TiB.

[root@ece4710 ~]# mkfs.ext4 -i 8192 -I 4096 -J size=400 -Odir_index,filetype /
    dev/mapper/de4801r1a01v01p1

[root@ece4710 ~]# mkdir /mnt/de4904-md-1
[root@ece4709 ~]# mkdir /mnt/de4904-md-2
[root@ece4710 ~]# mount /dev/mapper/de4801r1a02v01p1 /mnt/de4904-md-2/
[root@ece4709 ~]# mount /dev/mapper/de4801r1a01v01p1 /mnt/de4904-md-1/
[root@ece4709 ~]# df -h
Filesystem                      1K-blocks      Used   Available Use% Mounted on
...
```

[14] https://doc.beegfs.io/latest/advanced_topics/metadata_tuning.html#metadata_tuning


```
/dev/mapper/de4801r1a01v01p1  7.0T  259M  6.3T   1% /mnt/de4801-md-1
/dev/mapper/mpatht             56T  357G   56T   1% /mnt/de4801-2
/dev/mapper/mpathq             56T  357G   56T   1% /mnt/de4801-1
...
[root@ece4710 ~]# df -h
Filesystem                    1K-blocks      Used   Available Use% Mounted on
...
/dev/mapper/de4801r1a02v01p1  7.0T  259M  6.3T   1% /mnt/de4801-md-2
/dev/mapper/mpathm             56T  357G   56T   1% /mnt/de4801-4
/dev/mapper/mpathl             56T  357G   56T   1% /mnt/de4801-3
...
```

7.2 Installing BeeGFS packages

In order to access the RedHat 9 BeeGFS packages, we first add the repo info file into a directory that `dnf` manages and then refresh the dnf list of packages:

```
[root@ece4709 ~]# cat /etc/yum.repos.d/beegfs.repo
[beegfs]
name=BeeGFS 8.1 (rhel9)

baseurl=https://www.beegfs.io/release/beegfs_8.1/dists/rhel9

gpgkey=https://www.beegfs.io/release/beegfs_8.1/gpg/GPG-KEY-beegfs
gpgcheck=1
enabled=1

[root@ece4709 ~]# dnf update --refresh
BeeGFS 8.1 (rhel9)                                              29 kB/s | 2.9 kB
       00:00
...
```

We then install only the packages required for each of the servers as outlined in[^1]:

- ece4709: `dnf install beegfs-mgmtd beegfs-meta libbeegfs-ib beegfs-storage`
- ece4710: `dnf install beegfs-meta libbeegfs-ib beegfs-storage`
- Client nodes: `dnf install beegfs-client beegfs-utils`

Before we proceed with the setup of all the BeeGFS services, we need to create an authentication file to allow communication between the servers.

```
[root@ece4709 ~]# dd if=/dev/random of=/etc/beegfs/conn.auth bs=128 count=1
1+0 records in
1+0 records out
128 bytes copied, 3.7272e-05 s, 3.4 MB/s
[root@ece4709 ~]# chown root:root /etc/beegfs/conn.auth
[root@ece4709 ~]# chmod 400 /etc/beegfs/conn.auth
[root@ece4709 beegfs]# scp /etc/beegfs/conn.auth ece4710:/etc/beegfs/conn.auth
...
```

Where in the last line we copied this auth file to the other storage server. We then further need to copy this to all of the client servers in a similar way.

7.3 Setup of the BeeGFS Services

7.3.1 Setup of BeeGFS Management Service

To configure the management service, we invoke the included setup script and we choose to store the mgmt config in a local disk fs:

```
[root@ece4709 ~]# /opt/beegfs/sbin/beegfs-mgmtd --init
Created new database version 3 at "/var/lib/beegfs/mgmtd.sqlite".
```



```
[root@ece4709 beegfs]# vim /etc/beegfs/beegfs-mgmtd.toml
[root@ece4709 beegfs]# systemctl restart beegfs-mgmtd
[root@ece4709 beegfs]# systemctl status beegfs-mgmtd.service
. beegfs-mgmtd.service - BeeGFS Management Server
     Loaded: loaded (/usr/lib/systemd/system/beegfs-mgmtd.service; disabled;
        preset: disabled)
     Active: active (running) since Mon 2025-10-13 12:47:43 CEST; 2s ago
       Docs: https://doc.beegfs.io
   Main PID: 67585 (beegfs-mgmtd)
      Tasks: 66 (limit: 1644848)
     Memory: 38.0M
        CPU: 17ms
     CGroup: /system.slice/beegfs-mgmtd.service
             67585 /opt/beegfs/sbin/beegfs-mgmtd --log-target=journald

Oct 13 12:47:43 ece4709 systemd[1]: Starting BeeGFS Management Server...
Oct 13 12:47:43 ece4709 beegfs-mgmtd[67585]: Failed to load license verification
    library: /opt/beegfs/lib/libbeegfs_license.so: cannot open shared object
   file: No such file or directory
Oct 13 12:47:43 ece4709 beegfs-mgmtd[67585]: Initializing licensing library
   failed. Licensed features will be unavailable: License verification library
   not loaded.
Oct 13 12:47:43 ece4709 beegfs-mgmtd[67585]: gRPC server running with TLS
   disabled
Oct 13 12:47:43 ece4709 beegfs-mgmtd[67585]: Waiting for shutdown signal ...
Oct 13 12:47:43 ece4709 systemd[1]: Started BeeGFS Management Server.
```

Where as part of this setup we needed to edit the created config file to point to our auth file created in the previous step:

```
[root@ece4709 ~]# cat /etc/beegfs/beegfs-mgmtd.toml | grep auth-file
auth-file = "/etc/beegfs/conn.auth"
```

7.3.2 Setup of BeeGFS Metadata Service

We will proceed by configuring first the metadata service and second the storage service. On each of the metadata nodes we will run the setup script:

```
[root@ece4710 ~]# /opt/beegfs/sbin/beegfs-setup-meta -p /mnt/de4801-md-2/beegfs/
   beegfs_meta/ -s 13 -m ece4709
Preparing storage directory: /mnt/de4801-md-2/beegfs/beegfs_meta/
 * Creating format.conf file...
 * Creating server numeric ID file: /mnt/de4801-md-2/beegfs/beegfs_meta//
    nodeNumID
Updating config file: /etc/beegfs/beegfs-meta.conf
 * Setting management host: ece4709
 * Setting storage directory in config file...
 * Disabling usage of uninitialized storage directory in config file...
 * Fetching the underlying device...
Underlying device detected: /dev/mapper/de4801r6a02v01
Fetching UUID of the file system on that device...
Found UUID 091c1160-003d-4bed-8a56-b5f28e4478ff
Writing UUID to config file...
 * Setting usage of extended attributes to: true
All done.
[root@ece4710 ~]# systemctl restart beegfs-meta
[root@ece4710 ~]# systemctl status beegfs-meta
. beegfs-meta.service - BeeGFS Metadata Server
     Loaded: loaded (/usr/lib/systemd/system/beegfs-meta.service; enabled;
        preset: disabled)
     Active: active (running) since Mon 2025-10-13 12:52:33 CEST; 2s ago
```



```
       Docs: http://www.beegfs.com/content/documentation/
   Main PID: 72391 (beegfs-meta/Mai)
      Tasks: 394 (limit: 1644848)
     Memory: 243.5M
        CPU: 818ms
     CGroup: /system.slice/beegfs-meta.service
             72391 /opt/beegfs/sbin/beegfs-meta cfgFile=/etc/beegfs/beegfs-meta.
                conf runDaemonized=false

Oct 13 12:52:34 ece4710 beegfs-meta[72391]: RegDGramLis [Heartbeat incoming] >>
   Root (by Heartbeat): 13
Oct 13 12:52:34 ece4710 beegfs-meta[72391]: Main [Register node] >> Node
   registration successful.
Oct 13 12:52:34 ece4710 beegfs-meta[72391]: Main [NodeConn (acquire stream)] >>
   Connected: beegfs-mgmtd@172.30.87.69:8008 >
Oct 13 12:52:34 ece4710 beegfs-meta[72391]: Main [App] >> Registration and
   management info download complete.
Oct 13 12:52:34 ece4710 beegfs-meta[72391]: Main [DGramLis] >> Listening for UDP
    datagrams: any Port 8005
Oct 13 12:52:34 ece4710 beegfs-meta[72391]: Main [ConnAccept] >> Listening for
   RDMA connections: Port 8005
Oct 13 12:52:34 ece4710 beegfs-meta[72391]: Main [ConnAccept] >> Listening for
   TCP connections: Port 8005
Oct 13 12:52:34 ece4710 beegfs-meta[72391]: Main [App] >> Version: 8.1.0
Oct 13 12:52:34 ece4710 beegfs-meta[72391]: Main [App] >> LocalNode: beegfs-meta
    node_meta_13 [ID: 13]
Oct 13 12:52:34 ece4710 beegfs-meta[72391]: Main [App] >> Usable NICs: ibs3(RDMA
   ) ibs2(RDMA) ibs1(RDMA) ibs3(TCP) ibs2(TCP>
```

Here we have used

- the `-s` flag to ensure a unique ID is associated with each metadata server
- the `-m` flag to specify the hostname (or ip address) of the mgt server

Again need to specify an auth file to the service, this time to `/etc/beegfs/beegfs-meta.conf`:

```
[root@ece4709 ~]# cat /etc/beegfs/beegfs-meta.conf | grep connAuthFile
connAuthFile                 = /etc/beegfs/conn.auth
...
```

7.3.3 Setup of BeeGFS Storage Service

In our setup, we have two storage servers and four dm storage devices so we will use two devices on each server. If high availability is a requirement, we could also have setup buddy mirroring which is detailed in the online documentation[15].

We proceed in the installation by running the included setup script:

```
[root@ece4709 beegfs]# /opt/beegfs/sbin/beegfs-setup-storage -p /mnt/de4801-1/
   beegfs -s 4 -i 301 -m ece4709
Preparing storage target directory: /mnt/de4801-1/beegfs
 * Creating format.conf file...
 * Creating chunks directory...
 * Creating buddymir directory...
 * Creating target numeric ID file: /mnt/de4801-1/beegfs/targetNumID
 * Creating server numeric ID file: /mnt/de4801-1/beegfs/nodeNumID
Updating config file: /etc/beegfs/beegfs-storage.conf
 * Setting management host: ece4709
 * Appending to target directory list in config file...
 * Disabling usage of uninitialized storage targets in config file...
 * Fetching the underlying device...
```

[15] https://doc.beegfs.io/latest/advanced_topics/mirroring.html


```
Underlying device detected: /dev/mapper/de4801r6a01v01
Fetching UUID of the file system on that device...
Found UUID 4cb8319b-dc05-4c45-9bc0-a17af12d7689
Appending UUID to config file...
All done.
[root@ece4710 ~]# /opt/beegfs/sbin/beegfs-setup-storage -p /mnt/de4801-3/beegfs
    -s 14 -i 1303 -m ece4709
Preparing storage target directory: /mnt/de4801-3/beegfs
 * Creating format.conf file...
 * Creating chunks directory...
 * Creating buddymir directory...
 * Creating target numeric ID file: /mnt/de4801-3/beegfs/targetNumID
 * Creating server numeric ID file: /mnt/de4801-3/beegfs/nodeNumID
Updating config file: /etc/beegfs/beegfs-storage.conf
 * Setting management host: ece4709
 * Appending to target directory list in config file...
 * Disabling usage of uninitialized storage targets in config file...
 * Fetching the underlying device...
Underlying device detected: /dev/mapper/de4801r6a01v02
Fetching UUID of the file system on that device...
Found UUID c0530044-d65a-4969-8257-f4dfa72d387f
Appending UUID to config file...
All done.
[root@ece4710 ~]# /opt/beegfs/sbin/beegfs-setup-storage -p /mnt/de4801-4/beegfs
    -s 14 -i 1304 -m ece4709
Preparing storage target directory: /mnt/de4801-4/beegfs
 * Creating format.conf file...
 * Creating chunks directory...
 * Creating buddymir directory...
 * Creating target numeric ID file: /mnt/de4801-4/beegfs/targetNumID
 * Creating server numeric ID file: /mnt/de4801-4/beegfs/nodeNumID
Updating config file: /etc/beegfs/beegfs-storage.conf
 * Setting management host: ece4709
 * Appending to target directory list in config file...
 * Disabling usage of uninitialized storage targets in config file...
 * Fetching the underlying device...
Underlying device detected: /dev/mapper/de4801r6a02v02
Fetching UUID of the file system on that device...
Found UUID 4b8f0f2c-3ac9-4290-9806-a3ad6e7e58c7
Appending UUID to config file...
All done.
[root@ece4709 mapper]# /opt/beegfs/sbin/beegfs-setup-storage -p /mnt/de4801-2/
    beegfs -s 4 -i 302 -m ece4709 -f
Preparing storage target directory: /mnt/de4801-2/beegfs
 * Creating format.conf file...
 * Creating chunks directory...
 * Creating buddymir directory...
 * Creating target numeric ID file: /mnt/de4801-2/beegfs/targetNumID
 * Creating server numeric ID file: /mnt/de4801-2/beegfs/nodeNumID
Updating config file: /etc/beegfs/beegfs-storage.conf
 * Setting management host: ece4709
 * Appending to target directory list in config file...
 * Disabling usage of uninitialized storage targets in config file...
 * Fetching the underlying device...
Underlying device detected: /dev/mapper/de4801r6a01v02
Fetching UUID of the file system on that device...
Found UUID c0530044-d65a-4969-8257-f4dfa72d387f
Appending UUID to config file...
All done.
```


Where we have used:

- the `-s` flag to specify that both of the devices on each server have the same server ID
- the `-i` flag to give a unique id to each storage target
- the `-m` flag to specify the management server

Once we have specified the auth file in `/etc/beegfs/beegfs-storage.conf`, we adjust the size of the IO operations in the storage config, making sure to align the writeSize properly with the filesystem blocksize and RAID stripe size:

```
tuneBindToNumaZone            =
tuneFileReadAheadSize         = 0m
tuneFileReadAheadTriggerSize  = 4m
tuneFileReadSize              = 1m
tuneFileWriteSize             = 256k
tuneFileWriteSyncSize         = 0m
```

Once we have done this we can start the storage service on both of the storage nodes:

```
[root@ece4709 mapper]# systemctl restart beegfs-storage
[root@ece4709 mapper]# systemctl status beegfs-storage
. beegfs-storage.service - BeeGFS Storage Server
     Loaded: loaded (/usr/lib/systemd/system/beegfs-storage.service; enabled;
        preset: disabled)
     Active: active (running) since Mon 2025-10-13 13:26:31 CEST; 7s ago
       Docs: http://www.beegfs.com/content/documentation/
   Main PID: 84144 (beegfs-storage/)
      Tasks: 33 (limit: 1644848)
     Memory: 27.8M
        CPU: 16ms
     CGroup: /system.slice/beegfs-storage.service
             84144 /opt/beegfs/sbin/beegfs-storage cfgFile=/etc/beegfs/
                beegfs-storage.conf runDaemonized=false

Oct 13 13:26:31 ece4709 beegfs-storage[84144]: Main [RegDGramLis] >> Listening
   for UDP q:qdatagrams: any Port 8003
Oct 13 13:26:31 ece4709 beegfs-storage[84144]: RegDGramLis [
   addOrUpdateNodeUnlocked] >> Updating alias for local node:  -> node_stora>
Oct 13 13:26:31 ece4709 beegfs-storage[84144]: Main [Register node] >> Node
   registration successful.
Oct 13 13:26:31 ece4709 beegfs-storage[84144]: Main [InternodeSyncer.cpp:619] >>
    Storage targets registration successful.
Oct 13 13:26:31 ece4709 beegfs-storage[84144]: Main [Sync results] >> Nodes
   added: 2 (Type: beegfs-meta)
Oct 13 13:26:31 ece4709 beegfs-storage[84144]: Main [App] >> Registration and
   management info download complete.
Oct 13 13:26:31 ece4709 beegfs-storage[84144]: Main [DGramLis] >> Listening for
   UDP datagrams: any Port 8003
Oct 13 13:26:31 ece4709 beegfs-storage[84144]: Main [ConnAccept] >> Listening
   for RDMA connections: Port 8003
Oct 13 13:26:31 ece4709 beegfs-storage[84144]: Main [ConnAccept] >> Listening
   for TCP connections: Port 8003
Oct 13 13:26:31 ece4709 beegfs-storage[84144]: Main [App] >> 0 sessions restored
   .
```

7.3.4 Setup of BeeGFS Client Service

Finally, we configure the client service to allow us to mount the created beegfs filesystem on all of our client servers. To include RDMA support, we need to rebuild the client application to use the installed OFED.

```
$ cat /etc/beegfs/beegfs-client-autobuild.conf
...
```



```
buildArgs=-j8 OFED_INCLUDE_PATH=/usr/src/ofa_kernel/default/include/
...
[root@ece4709 mapper]# /etc/init.d/beegfs-client rebuild
- BeeGFS module autobuild
$OFED_INCLUDE_PATH = []
$OFED_INCLUDE_PATH = [/usr/src/ofa_kernel/default/include/]
$OFED_INCLUDE_PATH = [/usr/src/ofa_kernel/default/include/]
Building beegfs client module
feature detection gives: -DKERNEL_HAS_INODE_CTIME -DKERNEL_HAS_INODE_MTIME
    -DKERNEL_HAS_DENTRY_SUBDIRS -DKERNEL_HAS_SCHED_SIG_H
    -DKERNEL_HAS_LINUX_STDARG_H -DKERNEL_HAS_STATX -DKERNEL_HAS_KREF_READ
    -DKERNEL_HAS_FILE_DENTRY -DKERNEL_HAS_SUPER_SETUP_BDI_NAME
    -DKERNEL_HAS_KERNEL_READ -DKERNEL_HAS_SKWQ_HAS_SLEEPER
    -DKERNEL_HAS_CURRENT_TIME_SPEC64 -DKERNEL_WAKE_UP_SYNC_KEY_HAS_3_ARGUMENTS
    -DKERNEL_HAS_IOV_ITER_KVEC_NO_TYPE_FLAG_IN_DIRECTION -DKERNEL_HAS_PROC_OPS
    -DKERNEL_HAS_SOCKPTR_T -DKERNEL_HAS_SOCK_SETSOCKOPT_SOCKPTR_T_PARAM
    -DKERNEL_HAS_TIME64 -DKERNEL_HAS_KTIME_GET_TS64
    -DKERNEL_HAS_KTIME_GET_REAL_TS64 -DKERNEL_HAS_KTIME_GET_COARSE_REAL_TS64
    -DKERNEL_HAS_GENERIC_FILE_SPLICE_READ -DKERNEL_HAS_SETATTR_PREPARE
    -DKERNEL_HAS_GET_ACL -DKERNEL_HAS_SET_ACL -DKERNEL_HAS_SET_ACL_NS_INODE
    -DKERNEL_HAS_USER_NS_MOUNTS -DKERNEL_HAS_FOPS_ITERATE
    -DKERNEL_HAS_XATTR_HANDLERS_INODE_ARG -DKERNEL_HAS_CPU_IN_THREAD_INFO
feature detection gives: -DKERNEL_HAS_INODE_CTIME -DKERNEL_HAS_INODE_MTIME
    -DKERNEL_HAS_DENTRY_SUBDIRS -DKERNEL_HAS_SCHED_SIG_H
    -DKERNEL_HAS_LINUX_STDARG_H -DKERNEL_HAS_STATX -DKERNEL_HAS_KREF_READ
    -DKERNEL_HAS_FILE_DENTRY -DKERNEL_HAS_SUPER_SETUP_BDI_NAME
    -DKERNEL_HAS_KERNEL_READ -DKERNEL_HAS_SKWQ_HAS_SLEEPER
    -DKERNEL_HAS_CURRENT_TIME_SPEC64 -DKERNEL_WAKE_UP_SYNC_KEY_HAS_3_ARGUMENTS
    -DKERNEL_HAS_IOV_ITER_KVEC_NO_TYPE_FLAG_IN_DIRECTION -DKERNEL_HAS_PROC_OPS
    -DKERNEL_HAS_SOCKPTR_T -DKERNEL_HAS_SOCK_SETSOCKOPT_SOCKPTR_T_PARAM
    -DKERNEL_HAS_TIME64 -DKERNEL_HAS_KTIME_GET_TS64
    -DKERNEL_HAS_KTIME_GET_REAL_TS64 -DKERNEL_HAS_KTIME_GET_COARSE_REAL_TS64
    -DKERNEL_HAS_GENERIC_FILE_SPLICE_READ -DKERNEL_HAS_SETATTR_PREPARE
    -DKERNEL_HAS_GET_ACL -DKERNEL_HAS_SET_ACL -DKERNEL_HAS_SET_ACL_NS_INODE
    -DKERNEL_HAS_USER_NS_MOUNTS -DKERNEL_HAS_FOPS_ITERATE
    -DKERNEL_HAS_XATTR_HANDLERS_INODE_ARG -DKERNEL_HAS_CPU_IN_THREAD_INFO
Skipping BTF generation for /opt/beegfs/src/client/client_module_8/build/../
    source/beegfs.ko due to unavailability of vmlinux
$OFED_INCLUDE_PATH = [/usr/src/ofa_kernel/default/include/]
$OFED_INCLUDE_PATH = []

[root@ece4709 mapper]# systemctl restart beegfs-client
[root@ece4709 mapper]# systemctl status beegfs-client
. beegfs-client.service - Start BeeGFS Client
     Loaded: loaded (/usr/lib/systemd/system/beegfs-client.service; enabled;
         preset: disabled)
     Active: active (exited) since Mon 2025-10-13 13:32:27 CEST; 6s ago
    Process: 91104 ExecStart=/opt/beegfs/sbin/beegfs-client start (code=exited,
        status=0/SUCCESS)
   Main PID: 91104 (code=exited, status=0/SUCCESS)
        CPU: 199ms

Oct 13 13:32:26 ece4709 systemd[1]: Starting Start BeeGFS Client...
Oct 13 13:32:26 ece4709 beegfs-client[91104]: Starting BeeGFS Client:
Oct 13 13:32:26 ece4709 beegfs-client[91104]: - Loading BeeGFS modules
Oct 13 13:32:26 ece4709 beegfs-client[91104]: - Mounting directories from /etc/
    beegfs/beegfs-mounts.conf
Oct 13 13:32:26 ece4709 beegfs-client[91133]: + mount --internal -t beegfs
    --source beegfs_nodev --target /mnt/beegfs -orw,nosuid,cfg>
Oct 13 13:32:27 ece4709 systemd[1]: Finished Start BeeGFS Client.
```


Once this is rebuild we proceed like the previous services - using the setup script, editing the config to point to the copied authfile and then starting the service:

```
[root@ece4701 ~]# /opt/beegfs/sbin/beegfs-setup-client -m ece4709
Updating config file: /etc/beegfs/beegfs-client.conf
 * Setting management host: ece4709
All done.
[root@ece4701 ~]# cat /etc/beegfs/beegfs-client.conf | grep connAuthFile
connAuthFile = /etc/beegfs/conn.auth

[root@ece4701 ~]# systemctl start beegfs-client
[root@ece4701 ~]# systemctl status beegfs-client
. beegfs-client.service - Start BeeGFS Client
     Loaded: loaded (/usr/lib/systemd/system/beegfs-client.service; enabled;
         preset: disabled)
     Active: active (exited) since Mon 2025-01-20 11:04:07 CET; 4h 49min ago
    Process: 3052880 ExecStart=/etc/init.d/beegfs-client start (code=exited,
       status=0/SUCCESS)
   Main PID: 3052880 (code=exited, status=0/SUCCESS)
        CPU: 70ms

Jan 20 11:04:07 ece4701 systemd[1]: Starting Start BeeGFS Client...
```

7.4 Validation of BeeGFS install

With our services running we can check the BeeGFS filesystem is mounted under `df`:

```
[root@ece4709 ~]# df
Filesystem                      1K-blocks        Used    Available Use% Mounted
   on
beegfs_nodev                    224T             1.4T    223T      1%   /mnt/
   beegfs
```

Further, we check that the filesystem is healthy using `beegfs health df`:

```
[root@ece4709 ~]# beegfs health df
----------------
Metadata Targets
----------------
ID    TYPE  ALIAS           NODE  STORAGE_POOL  CAP_POOL  SPACE   SPACE_USED
          SPACE_FREE  INODES  INODES_USED    INODES_FREE
m:3   meta  target_meta_3   m:3   (n/a)         Normal    7.0TiB  258.1MiB (0.00
   %)  7.0TiB      1.9G    66.1k (0.00%)  1.9G
m:13  meta  target_meta_13  m:13  (n/a)         Normal    7.0TiB  258.2MiB (0.00
   %)  7.0TiB      1.9G    66.1k (0.00%)  1.9G

---------------
Storage Targets
---------------
ID      TYPE     ALIAS                NODE  STORAGE_POOL  CAP_POOL  SPACE
   SPACE_USED        SPACE_FREE  INODES  INODES_USED   INODES_FREE
s:301   storage  target_0-68ECE1E7-4  s:4   s:1           Normal    55.9TiB  356
   .2GiB (0.62%)  55.5TiB     300.0M  13.0 (0.00%)  300.0M
s:302   storage  target_1-68ECE1E7-4  s:4   s:1           Normal    55.9TiB  356
   .2GiB (0.62%)  55.5TiB     300.0M  13.0 (0.00%)  300.0M
s:1303  storage  target_0-68ECE248-E  s:14  s:1           Normal    55.9TiB  356
   .2GiB (0.62%)  55.5TiB     300.0M  13.0 (0.00%)  300.0M
s:1304  storage  target_1-68ECE248-E  s:14  s:1           Normal    55.9TiB  356
   .2GiB (0.62%)  55.5TiB     300.0M  13.0 (0.00%)  300.0M
```

And that each of our server is pingable and the storage servers are using the correct protocol:


```
[root@ece4709 ~]#  beegfs node list --with-nics --tls-disable
m:3   meta         node_meta_3               rdma:ibs3 (172.30.119.109:8005)
                                             rdma:ibs2 (172.30.119.89:8005)
                                             rdma:ibs1 (172.30.119.69:8005)
                                             ethernet:ibs3 (172.30.119.109:8005)
                                             ethernet:ibs2 (172.30.119.89:8005)
                                             ethernet:ibs1 (172.30.119.69:8005)
                                             ethernet:ens4f0np0 (172.30.87.69:
                                                8005)
                                             ethernet:enp0s20f0u1u6 (169.254.95.
                                                120:8005)
m:13  meta         node_meta_13              rdma:ibs3 (172.30.119.110:8005)
                                             rdma:ibs2 (172.30.119.90:8005)
                                             rdma:ibs1 (172.30.119.70:8005)
                                             ethernet:ibs3 (172.30.119.110:8005)
                                             ethernet:ibs2 (172.30.119.90:8005)
                                             ethernet:ibs1 (172.30.119.70:8005)
                                             ethernet:ens4f0np0 (172.30.87.70:
                                                8005)
                                             ethernet:enp0s20f0u1u6 (169.254.95.
                                                120:8005)
s:4   storage      node_storage_4            rdma:ibs3 (172.30.119.109:8003)
                                             rdma:ibs2 (172.30.119.89:8003)
                                             rdma:ibs1 (172.30.119.69:8003)
                                             ethernet:ibs3 (172.30.119.109:8003)
                                             ethernet:ibs2 (172.30.119.89:8003)
                                             ethernet:ibs1 (172.30.119.69:8003)
                                             ethernet:ens4f0np0 (172.30.87.69:
                                                8003)
                                             ethernet:enp0s20f0u1u6 (169.254.95.
                                                120:8003)
s:14  storage      node_storage_14           rdma:ibs3 (172.30.119.110:8003)
                                             rdma:ibs2 (172.30.119.90:8003)
                                             rdma:ibs1 (172.30.119.70:8003)
                                             ethernet:ibs3 (172.30.119.110:8003)
                                             ethernet:ibs2 (172.30.119.90:8003)
                                             ethernet:ibs1 (172.30.119.70:8003)
                                             ethernet:ens4f0np0 (172.30.87.70:
                                                8003)
                                             ethernet:enp0s20f0u1u6 (169.254.95.
                                                120:8003)
mg:1  management   management                ethernet:ens4f0np0 (172.30.87.69:
   8008,8010)
                                             ethernet:enp0s20f0u1u6 (169.254.95.
                                                120:8008,8010)
                                             ethernet:ibs2 (172.30.119.89:8008,
                                                8010)
                                             ethernet:ibs1 (172.30.119.69:8008,
                                                8010)
                                             ethernet:ibs3 (172.30.119.109:8008,
                                                8010)
                                             ethernet:lo (127.0.0.1:8008,8010)
```

8 Performance Summary

In this section we will demonstrate the IO performance we were able to achieve with the DE6600F system setup as described in the previous sections. We will use both IOR[16] and MDTest[17] as standard performance

[16] https://github.com/hpc/ior
[17] https://github.com/hpc/ior

measurements. For each benchmark we had up to 10 nodes available as compute clients for creating IO tasks.

8.1 IOR

IOR is a tool we will use for measuring sequential read and write performance of the storage arrays. IO tasks are created by MPI tasks running on each of the client nodes which are synchronised with MPI collective communication.

8.1.1 IOR Setup

IOR was compiled with openmpi4.1.7 installed via dnf, and was build with gpfs libraries included.

```
dnf -y install openmpi-devel openmpi

mpipath=$(dirname $(rpm -ql openmpi | grep mpiexec | grep -v man))
export PATH=${mpipath}:${PATH}

wget https://github.com/hpc/ior/releases/download/4.0.0/ior-4.0.0.tar.gz
tar -xvzf ~/ior-4.0.0.tar.gz
cd ior-4.0.0
./configure --prefix=/root/ce/ior-install
make
make install
```

For the benchmarking presented we used IOR with the "File-per-process" option using POSIX api. Additional flags[18] are included for performing fsync operation after write tasks are complete and for adding barriers and delays between read/write operations and iterations.

IOR was run using the following script with the number of ranks, hosts, transfer size and total file size adapted in each run.

```
RUNNING=/root/RUNNING

if [ -f $RUNNING ]; then
  echo "A job is already running. Please wait until the file $RUNNING does not
     exist."
  exit
fi

touch $RUNNING
DATE=$(date +%s)
TASKS=200
BLOCK=50g
XFER=variable
OUTFILE=ior.$TASKS.$BLOCK.$XFER.$DATE
HOSTFILE=hostfile_10n

mpiexec -np $TASKS --allow-run-as-root -hostfile $HOSTFILE -map-by NODE /root/ce
   /ior-latest-install/bin/ior -a POSIX -C -Q 1 -i20 -r -w -b$BLOCK -t1m -v
   --posix.odirect -e -g -d 10 -F -o /mnt/beegfs/ior | tee -a $OUTFILE

rm -f $RUNNING
```

8.2 IOR Performance Results

8.2.1 Single Client

The following plots demonstrate the maximum achieved read/write throughput performance measured with IOR using multiple MPI tasks on a single client with a total size per client of 10GB. For each point on the plots, the maximum recorded throughput (in GB/s) across 5 iterations was recorded. For this measurement, we used a blocksize in our underlying filesystem/RAID array which allowed FSWA to activate.

[18] https://ior.readthedocs.io/en/latest/userDoc/options.html

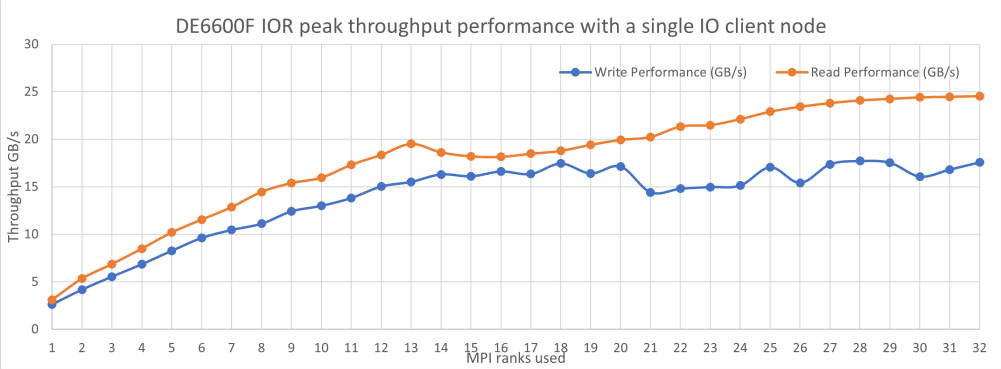

Figure 4: Plot of IOR throughput performance for a single client node.

8.2.2 Multiple Clients

The following is the peak write and read bandwidth measurements from running IOR with 10 clients, 10 tasks per node, an 8M transfer size and doing 20 iterations of each step using our setup with a single DE6600F controller. We split the results into two tables, one where we have used an xfs blocksize and RAID segment size of 128k (giving a full array stripe size of 1M) and one where we used an xfs blocksize/RAID segment size of 32k (giving a full array stripe size of 256k).

For 128k XFS blocksize / RAID Segment size:

Max Write (MiB/s)	Max Read (MiB/s)	Max Write (GB/s)	Max Read (GB/s)
13268.05	40129.03	13.91256	42.07834

For 32k XFS blocksize / RAID Segment size:

Max Write (MiB/s)	Max Read (MiB/s)	Max Write (GB/s)	Max Read (GB/s)
18538.39	38088.51	19.25016	38.08851

During the testing, we discovered that the full stripe write acceleration was not triggering with the 128k blocksize. The reason for this is suspected to be with interference from the xfs journal writes. The XFS journal will perform journal write operations of size 256k which is known to interfere with the full stripe write acceleration learning algorithm if the full stripe write blocks are a different size from the journal size of 256k. For an 8+2p RAID array, this corresponds to having to tune the underlying filesystem blocksize to 32k.

A downsize to this is that this is much smaller than what was known to be the optimal blocksize/RAID segment size for achieving the best performance of the system. Indeed in this case we see about a 10% in read performance when lowering the blocksize from 128k to 32k, so the increase in write performance appears to come with a penalty to read performance. Depending on the customer's preferences, either choice in blocksize could be preferable.

8.3 MDTEST

We also performed some MDTEST benchmarks to measure the performance of small file io on the setup. With BeeGFS we have metadata targets and storage targets, each with there own underlying filesystem. For small files these are always written to the storage, and tuning the inode size will not suddenly allow these to be saved in inodes. For this reason, we do not see a difference in mdtest performance using {512, 3k, 15k} file sizes.


```
Command line used: /root/ce/ior-latest-install/bin/mdtest '-n' '10000' '-u' '-w'
    '15360' '-e' '15360' '-a' 'POSIX' '-i' '3' '-P' '-v' '-d' '/mnt/beegfs/ior/'
SUMMARY rate (in ops/sec): (of 3 iterations)
100 tasks, 1000000 files/directories

   Operation                     Max            Min           Mean        Std
      Dev
   ---------                     ---            ---           ----
      -------
   Directory creation          17041.976      15895.888      16566.997
      597.687
   Directory stat              22378.544      21080.298      21874.309
      695.944
   Directory rename            43192.180      39938.916      41282.520
      1698.896
   Directory removal           11324.411      10949.542      11136.512
      187.436
   File creation               70478.849      59807.378      64120.521
      5622.025
   File stat                   26928.766      24804.322      25957.675
      1073.886
   File read                   89427.658      76935.310      84844.948
      6878.664
   File removal                66130.209      57400.724      62562.354
      4577.779
   Tree creation                  55.580         47.700         51.906
      3.967
   Tree removal                    0.618          0.597          0.610
      0.011

SUMMARY time (in ms/op): (of 3 iterations)
   Operation                     Max            Min           Mean        Std
      Dev
   ---------                     ---            ---           ----
      -------
   Directory creation             62.909         58.679         60.414
      2.215
   Directory stat                 47.438         44.686         45.747
      1.480
   Directory rename               25.038         23.152         24.250
      0.980
   Directory removal              91.328         88.305         89.812
      1.512
   File creation                  16.720         14.189         15.673
      1.321
   File stat                      40.316         37.135         38.569
      1.613
   File read                      12.998         11.182         11.841
      1.005
   File removal                   17.421         15.122         16.043
      1.216
   Tree creation                   0.021          0.018          0.019
      0.002
   Tree removal                    1.675          1.619          1.639
      0.031
```

Additionally, we performed some MDTEST benchmarks using a setup closer to the recommended setup in the beegfs documentation[19] for 0 size files to expose the maximum performance of the filesystem:

[19] https://doc.beegfs.io/latest/advanced_topics/benchmark.html


```
Command line used: /root/ce/ior-latest-install/bin/mdtest '-I' '2000' '-i' '3' '
   -F' '-D' '-P' '-N' '1' '-t' '-u' '-b' '8' '-d' '/mnt/beegfs/ior/'
320 tasks, 640000 files/directories

SUMMARY rate (in ops/sec): (of 3 iterations)
   Operation                     Max            Min           Mean        Std
      Dev
   ---------                     ---            ---           ----
      -------
   Directory creation          18801.173      14331.246      16013.305
      2431.496
   Directory stat              24732.736      24161.254      24494.310
      297.261
   Directory rename            44408.948      42317.334      43266.048
      1059.242
   Directory removal           11386.220      10951.147      11096.524
      250.885
   File creation               76567.873      72076.538      74886.325
      2449.024
   File stat                   27720.287      27227.847      27479.364
      246.391
   File read                  119071.827     115225.026     117717.002
      2160.857
   File removal                74583.539      71345.497      73306.974
      1724.256
   Tree creation                  31.658          7.299         22.603
      13.328
   Tree removal                    1.136          1.092          1.117
      0.023

SUMMARY time (in ms/op): (of 3 iterations)
   Operation                     Max            Min           Mean        Std
      Dev
   ---------                     ---            ---           ----
      -------
   Directory creation             44.658         34.040         40.543
      5.697
   Directory stat                 26.489         25.877         26.131
      0.319
   Directory rename               15.124         14.412         14.798
      0.360
   Directory removal              58.441         56.208         57.695
      1.288
   File creation                   8.879          8.359          8.552
      0.285
   File stat                      23.505         23.088         23.291
      0.209
   File read                       5.554          5.375          5.438
      0.101
   File removal                    8.970          8.581          8.734
      0.208
   Tree creation                   0.137          0.032          0.068
      0.060
   Tree removal                    0.916          0.880          0.895
      0.018
```


9 Acknowledgements

This work was done in collaboration with ThinkParq who provided insights into how to achieve maximum performance from a BeeGFS setup.

The authors would like to thank the following people from Lenovo for their contributions to this project:

- Gilad Berman
- Steve Eiland
- Michael Julien
- Olivier Lagrasse
- Qudus Mayowa

