Technical Report

BeeGFS with NetApp E-Series

Solution Deployment

Abdel Sadek and Joe McCormick, NetApp
August 2020 | TR-4755

Abstract

This document provides an overview of how to deploy BeeGFS parallel file systems with NetApp® E-Series storage systems. This guide includes a basic building block that is composed of a hybrid E-Series instance as the back end for metadata and storage targets. It also discusses the advanced building block in which metadata uses a separate EF-Series all-flash array.

NetApp®

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Solution Overview

With a competitive price-to-performance ratio, NetApp® E-Series storage systems make an excellent choice for BeeGFS back-end storage. E-Series also supports a wide range of high-performing SAN protocols, making it a flexible choice for most use cases. E-Series supports the following protocols:

- InfiniBand (IB) running iSCSI Extensions for RDMA (iSER)
- IB running the SCSI RDMA Protocol (SRP)
- NVMe over IB (NVMe/IB)
- NVMe over RDMA over Converged Ethernet (NVMe/RoCE)
- NVMe over Fibre Channel (NVMe/FC)
- SAS
- FC
- iSCSI

1.1 Solution Technology

BeeGFS is a parallel file system with an architecture that is based on four main services:

- **Management service.** Registers and monitors all other services.
- **Storage service.** Stores the distributed user file contents known as data chunk files.
- **Metadata service.** Keeps track of the file system layout, directory and file attributes, and so on.
- **Client service.** Mounts the file system to access the stored data.

Note: For more information about BeeGFS, see the links that are provided in the section “Where to Find Additional Information.”

NetApp E-Series storage systems provide a solid back-end target for both the storage service and the metadata service. This document describes the setup and scale-out of BeeGFS configurations based on E-Series building blocks.

This document does not cover BeeGFS On Demand (BeeOND). Detailed installation steps are available at the BeeGFS official wiki.

To implement this solution, you must verify the compatibility of all components (operating systems, multipathing, host bus adapters [HBAs], switches, and so on). To verify the components, see the NetApp Interoperability Matrix Tool (IMT).

For information about host setup, see SANtricity Software Express Configuration for Linux in the E-Series and SANtricity 11 Documentation Center.

1.2 Use Case Summary

This solution applies to the following use cases:

- **Basic building block.** For this use case, both metadata and data reside on the same E-Series hybrid storage system.
- **Advanced building block.** For this use case, metadata resides on an EF-Series all-flash system, and data resides on an E-Series system with HDDs.

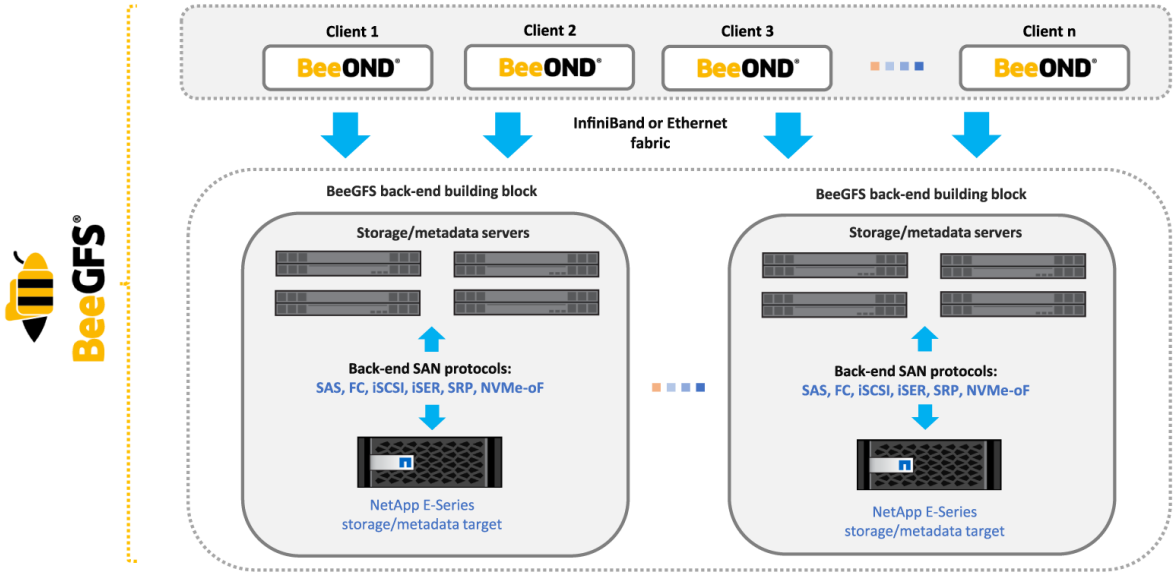
2 Basic Building Block

2.1 Technology Requirements

As shown in Figure 1, the basic back-end building block comprises a NetApp E-Series hybrid storage system with both SSDs and HDDs for data. Volumes that are created from the SSDs store metadata, and volumes that are created from the HDDs serve as storage. You can connect an E-Series storage system directly to up to four servers running the storage service, with at least one of the servers also running the metadata service.

Adding a switch to the back end increases the potential number of servers for each storage system. Connecting the back end directly without a switch helps to lower your user costs.

Figure 1) BeeGFS basic building block on E-Series systems.



Hardware Requirements

In the following deployment procedures, 100Gb IB host channel adapters (HCAs) connect each server (metadata or storage) to the E-Series array. However, you can use any other block protocol that E-Series supports. For simplicity in this use case, metadata and management services are run on one server, and the storage service is run on another two servers. There are also servers running the client service. Table 1 lists the hardware components that are required to implement the use case.

Table 1) Hardware requirements.

BeeGFS Node ID	Name	BeeGFS Role	Hardware Specifications
–	`ictm1621c1`	BeeGFS metadata and storage targets	• Model: NetApp E5760 • NetApp SANtricity® OS version: 11.60.2R1 • Drives (60 total): 4 SSDs, 56 HDDs
1	`ictm1624h3`	BeeGFS management and metadata services	• Memory: 96GB • CPU: Intel Xeon Gold 6132 12C @ 3.0GHz

BeeGFS Node ID	Name	BeeGFS Role	Hardware Specifications
			• HCAs: 2x Mellanox MCX556A-ECAT (100Gbps)
1	`ictm1624h4`	BeeGFS storage service	• Memory: 96GB • CPU: Intel Xeon Gold 6132 12C @ 3.0GHz • HCAs: 2x Mellanox MCX556A-ECAT (100Gbps)
2	`ictm1624h5`	BeeGFS storage service	• Memory: 96GB • CPU: Intel Xeon Gold 6132 12C @ 3.0GHz • HCAs: 2x Mellanox MCX556A-ECAT (100Gbps)

–	`ictm1624sw1`
	`ictm1624sw2`
	`ictm1625sw1`
	`ictm1625sw2`
		Front-end and back-end IB switches	• Mellanox MSB7800 (100Gbps)
			• Software version: 3.8.1054

–	`ictm1625h1a` `ictm1625h2-11`	Servers running the client service (compute nodes)	• Memory: 96GB • CPU: Intel Xeon Gold 6132 12C @ 3.0GHz • HCAs: 1x Mellanox MCX556A-ECAT (100Gbps)

Note: BeeGFS node IDs must be unique only for each service type.

Software Requirements

Table 2 lists the software components that are required to implement the basic building block use case.

Table 2) Software requirements.

Software	Version
Operating system	Red Hat Enterprise Linux Server 7.7
NetApp SANtricity OS release	11.60.2R1
BeeGFS	7.1.5

2.2 Deployment Procedures

BeeGFS deployment with E-Series follows four main steps:

1. Set up the E-Series storage array.
2. Set up the server back end.
3. Set up the front end and client-servers.
4. Set up BeeGFS.

Set Up the E-Series Storage Array

Follow the SANtricity online help to create volumes for the metadata and storage services. After you have created the volumes, follow these steps:

1. Use SANtricity System Manager to assign the volumes to the appropriate servers, as follows:

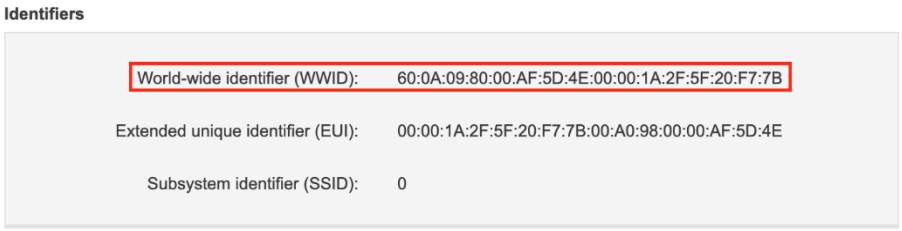
a. Assign volumes for the storage service to storage servers.

b. Assign the volume for the metadata service to the metadata server.

Name	Status	Assigned To	LUN	Pool/ Volume Group	Reported Capacity (GiB)	Allocated Capacity (GiB)
metadata_01	Optimal	Host ictm1624h3	1	Volume Group metadata_01	6200.00	6200.00
storage_01_tgt_01	Optimal	Host ictm1624h4	1	Volume Group storage_01_tgt_01	109422.00	109422.00
storage_01_tgt_02	Optimal	Host ictm1624h4	2	Volume Group storage_01_tgt_02	109422.00	109422.00
storage_02_tgt_03	Optimal	Host ictm1624h5	1	Volume Group storage_02_tgt_03	109422.00	109422.00
storage_02_tgt_04	Optimal	Host ictm1624h5	2	Volume Group storage_02_tgt_04	109422.00	109422.00

The preceding screenshot shows the volume-to-host assignments that are used for this deployment.

2. For subsequent steps, you need the worldwide identifier (WWID) of each volume to set up user-friendly names. To identify the WWID, select a volume and click View/Edit Settings and look under Identifiers.



This screenshot shows the WWID for the volume `metadata_01`.

Set Up Back-End Storage Connections and User-Friendly Names

For detailed instructions on how to set up storage connections, see the Linux Express Guide for the protocol that you are using to connect BeeGFS servers (metadata and storage) to E-Series storage systems.

To simplify the association of BeeGFS metadata and storage targets to their corresponding volumes on the E-Series storage array, use these steps to configure user-friendly names.

Note: In the following example, the volume `metadata_01` has the WWID `600A098000AF5D4E00001A2F5F20F77B`, as shown in the previous section.

1. Edit the file `/etc/multipath.conf`.

   Note: You must prefix `3` to the WWID.

```
defaults {
            user_friendly_names yes
}
multipaths {
                multipath {
                            wwid 3600a098000af5d4e00001a2f5f20f77b
                            alias metadata_01
                }
}
```

2. Restart the multipath service.

```
#  systemctl restart multipathd
```

3. After you have followed the procedure on all storage and metadata servers, Device Mapper Multipathing shows the user-friendly names of the devices:

   a. View the metadata server information.

```
[root@ictm1624h3 ~]# multipath -ll
metadata_01 (3600a098000af5d4e00001a2f5f20f77b) dm-3 NETAPP  ,INF-01-00
size=6.1T features='4 queue_if_no_path pg_init_retries 50 retain_attached_hw_handle' hwhandler='1
alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| |- 19:0:0:1 sdb 8:16 active ready running
| `- 20:0:0:1 sdc 8:32 active ready running
`-+- policy='service-time 0' prio=10 status=enabled
  |- 21:0:0:1 sdd 8:48 active ready running
  `- 22:0:0:1 sde 8:64 active ready running
```

b. View the first storage server information.

```
[root@ictm1624h4 ~]# multipath -ll
storage_01_tgt_01 (3600a098000af5d4e00001a335f20f7e2) dm-7 NETAPP  ,INF-01-00
size=107T features='4 queue_if_no_path pg_init_retries 50 retain_attached_hw_handle' hwhandler='1
alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| |- 19:0:0:1 sdb 8:16  active ready running
| `- 20:0:0:1 sdd 8:48  active ready running
`-+- policy='service-time 0' prio=10 status=enabled
  |- 21:0:0:1 sdf 8:80  active ready running
  `- 22:0:0:1 sdh 8:112 active ready running
storage_01_tgt_02 (3600a098000af5b500000160d5f20f6c6) dm-6 NETAPP  ,INF-01-00
size=107T features='4 queue_if_no_path pg_init_retries 50 retain_attached_hw_handle' hwhandler='1
alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| |- 21:0:0:2 sdg 8:96  active ready running
| `- 22:0:0:2 sdi 8:128 active ready running
`-+- policy='service-time 0' prio=10 status=enabled
  |- 19:0:0:2 sdc 8:32  active ready running
  `- 20:0:0:2 sde 8:64  active ready running
```

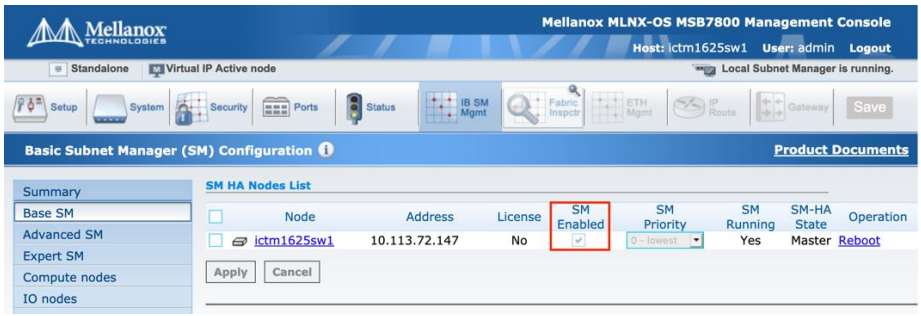
c. View the second storage server information.

```
[root@ictm1624h5 ~]# multipath -ll
storage_02_tgt_03 (3600a098000af5b500000160f5f20f6cc) dm-7 NETAPP  ,INF-01-00
size=107T features='4 queue_if_no_path pg_init_retries 50 retain_attached_hw_handle' hwhandler='1
alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| |- 21:0:0:2 sdg 8:96  active ready running
| `- 22:0:0:2 sdi 8:128 active ready running
`-+- policy='service-time 0' prio=10 status=enabled
  |- 19:0:0:2 sdc 8:32  active ready running
  `- 20:0:0:2 sde 8:64  active ready running
storage_02_tgt_04 (3600a098000af5d4e00001a315f20f78e) dm-6 NETAPP  ,INF-01-00
size=107T features='4 queue_if_no_path pg_init_retries 50 retain_attached_hw_handle' hwhandler='1
alua' wp=rw
|-+- policy='service-time 0' prio=50 status=active
| |- 19:0:0:1 sdb 8:16  active ready running
| `- 20:0:0:1 sdd 8:48  active ready running
`-+- policy='service-time 0' prio=10 status=enabled
  |- 21:0:0:1 sdf 8:80  active ready running
  `- 22:0:0:1 sdh 8:112 active ready running
```

Set Up Front-End Server and Client Connectivity

Note: If you already have an IB subnet manager configured or running, the following steps are not necessary.

1. Complete the following steps with the GUI:
   a. Open a web browser and navigate to the switch IP (https://<switch-ip>).
   b. Log in, then go to IB SM Mgmt and select Base SM.
   c. Select the SM Enabled option.
   d. Click Save.



1. On all nodes (management, metadata, storage, and clients), set up the IP addresses on the IB ports. The IB drivers are part of the inbox `rdma-core` driver that comes with the OS. The following example shows the procedure on a single server:

   a. Start and enable the RDMA service with IP over IB (IPoIB) enabled.

```
[root@ictm1625h1a ~]# cat /etc/rdma/rdma.conf | grep -i IPoIB
# Load IPoIB
IPOIB_LOAD=yes
[root@ictm1625h1a ~]# systemctl enable rdma.service
[root@ictm1625h1a ~]# systemctl start rdma.service
[root@ictm1625h1a ~]# systemctl status rdma.service
```

   b. Verify that the IB port or ports are up.

```
[root@ictm1624h4 ~]# ibstatus mlx5_0
Infiniband device''mlx5_'' port 1 status:
        default gid:     fe80:0000:0000:0003:b859:9f03:00d5:4e0a
        base lid:        0x6
        sm lid:          0x1
        state:           4: ACTIVE
        phys state:      5: LinkUp
        rate:            100 Gb/sec (4X EDR)
        link_layer:      InfiniBand
```

   c. Assign the IP address; the exact configuration steps might vary depending on your Linux version. The following is for an example Red Hat 7.7 ifcfg file:

```
[root@ictm1624h4 ~]# cat /etc/sysconfig/network-scripts/ifcfg-ib0
DEVICE=ib0
BOOTPROTO=none
ONBOOT=yes
IPADDR=192.168.3.24
PREFIX=24
MTU=
TYPE=InfiniBand
```

3. Verify that the following conditions are true:
   - The hosts can ping each other's IPs and host names (if you want DNS and it is set up).
   - Volumes are assigned to the appropriate hosts.
   - IB IPs can ping each other.

If the configuration meets all the requirements, you are ready to set up BeeGFS.

Set Up BeeGFS

Before you install BeeGFS on each node, you must add the BeeGFS package repository for your Linux distribution. You can find repository files for Red Hat, SUSE, and Debian (including Ubuntu) on the BeeGFS Download and Installation of Packages wiki. After you have added the repository, install the packages that are listed in Table 3 on the appropriate servers by using your package manager.

Table 3) Required packages for BeeGFS.

Node	Packages
Management server	`beegfs-mgmtd`
Metadata servers	`beegfs-meta libbeegfs-ib`
Storage servers	`beegfs-storage libbeegfs-ib`
Clients	`beegfs-client beegfs-helperd beegfs-utils`

Note: The `libbeegfs-ib` package is required for storage and metadata servers only when you use IB.

Set Up the Management Service

To set up the management service, complete the following steps:

1. Set up the management location: `# /opt/beegfs/sbin/beegfs-setup-mgmtd -p <management_data_location>`.

```
ictm1624h3:~# /opt/beegfs/sbin/beegfs-setup-mgmtd -p /data/beegfs/mgmt/
```

2. Start the management service and enable it to start at boot.

```
ictm1624h3:~# systemctl start beegfs-mgmtd.service
ictm1624h3:~# systemctl enable beegfs-mgmtd.service
ictm1624h3:~# systemctl status beegfs-mgmtd.service
```

Set Up the Metadata Service

To set up the metadata service, complete the following steps:

1. Format the mapped volume:
   `# mkfs.ext4 -i 2048 -I 512 -J size=400 -Odir_index,filetype <mapped volume>.`

```
ictm1624h3:~ # mkfs.ext4 -i 2048 -I 512 -J size=400 -Odir_index,filetype /dev/mapper/metadata_01
```

2. Mount each mapped volume:
   `# mount -onoatime,nodiratime,nobarrier <mapped volume> <mount point>.`

```
ictm1624h3:~ # mount -onoatime,nodiratime,nobarrier /dev/mapper/metadata_01 /mnt/metadata_01
```

3. Add the following line to `/etc/fstab` to automatically mount when the server boots:

```
/dev/mapper/metadata_01 /mnt/metadata_01 ext4 rw,noatime,nodiratime,nobarrier,_netdev 0 0
```

   Note: To confirm that the network stack is initialized before the OS attempts to mount the volumes, NetApp recommends that you include the `_netdev` mount parameter in `fstab`.

4. Configure the metadata service:
   `# /opt/beegfs/sbin/beegfs-setup-meta -p <mount point> -s <node id> -m <management_node_hostname_or_ip>`.

```
ictm1624h3:~# /opt/beegfs/sbin/beegfs-setup-meta -p /mnt/metadata_01 -s 1 -m ictm1624h3
```

   Note: In this example, the metadata service is configured as BeeGFS node ID 1 (as shown in Table 1).

5. NetApp recommends that you specify which network interface or interfaces other services can use to contact this metadata service. Create a text file under `/etc/beegfs`, with the interface names to be used listed one per line. In the following example, `ib0` and `ib1` are used. The absolute path of the file must be specified in the configuration file `/etc/beegfs-meta.conf` by using the `connInterfacesFile` parameter.

```
ictm1624h3:~ # cat /etc/beegfs/connInterfacesFile.conf
ib0
ib1
ictm1624h3:~ # cat /etc/beegfs/beegfs-meta.conf | grep connInterfacesFile
connInterfacesFile = /etc/beegfs/connInterfacesFile.conf
```

6. Start the metadata service and enable it to start at boot.

```
ictm1625h3:~# systemctl start beegfs-meta.service
ictm1625h3:~# systemctl enable beegfs-meta.service
ictm1625h3:~# systemctl status beegfs-meta.service
```

Set Up the Storage Service

To configure the storage service, complete the following steps:

1. Format each mapped volume:
   `# mkfs.xfs -d su=128k,sw=8 -l version=2,su=128k <mapped volume>.`

   Note: `su` is the segment size of the volume, and `sw` is the number of disks in the RAID group, not including the parity disks.

```
ictm1624h4:~# mkfs.xfs -d su=128k,sw=8 -l version=2,su=128k /dev/mapper/storage_01_tgt_01
```

2. Mount each mapped volume:
   `# mount -`
   `onoatime,nodiratime,logbufs=8,logbsize=256k,largeio,inode64,swalloc,allocsiz`
   `e=131072k,nobarrier <mapped volume> <mount point>.`

```
ictm1624h4:~ # mount -
onoatime,nodiratime,logbufs=8,logbsize=256k,largeio,inode64,swalloc,allocsize=131072k,nobarrier
/dev/mapper/storage_01_tgt_01 /mnt/storage_01_tgt_01
```

3. Add each mount point to `/etc/fstab` to automatically mount when the server boots.

```
/dev/mapper/storage_01_tgt_01 /mnt/storage_01_tgt_01 xfs
rw,noatime,nodiratime,logbufs=8,logbsize=256k,largeio,inode64,swalloc,allocsize=131072k,nobarrier,_
netdev 0 0
```

   Note: To confirm that the network stack is initialized before the OS attempts to mount the volumes, NetApp recommends that you include the `_netdev` mount parameter in `fstab`.

4. Configure the storage service:
   `# /opt/beegfs/sbin/beegfs-setup-storage -p <mount point> -s <node id> -i`
   `<target id > -m <management_node_hostname_or_ip>.`

   This example includes two BeeGFS storage nodes with two storage targets each, run for each target:

   - Storage01 with BeeGFS node ID 1:

```
ictm1624h4:~ # /opt/beegfs/sbin/beegfs-setup-storage -p /mnt/storage_01_tgt_02 -s 1 -i 101 \
-m ictm1624h3
ictm1624h4:~ # /opt/beegfs/sbin/beegfs-setup-storage -p /mnt/storage_01_tgt_02 -s 1 -i 102 \
-m ictm1624h3
```

   - Storage02 with BeeGFS node ID 2:

```
ictm1624h5:~ # /opt/beegfs/sbin/beegfs-setup-storage -p /mnt/storage_02_tgt_03 -s 2 -i 203 \
-m ictm1624h3
ictm1624h5:~ # /opt/beegfs/sbin/beegfs-setup-storage -p /mnt/storage_02_tgt_04 -s 2 -i 204 \
-m ictm1624h3
```

5. NetApp recommends that you specify which network interface or interfaces other services can use to contact the storage services. Create a text file under `/etc/beegfs`, with the interface names to be used listed one per line. In the following example, `ib0` and `ib1` are used. The absolute path of the file must be specified in the configuration file `/etc/beegfs-storage.conf` by using the `connInterfacesFile` parameter.

```
ictm1624h4:~ # cat /etc/beegfs/connInterfacesFile.conf
ib0
ib1
ictm1624h4:~ # cat /etc/beegfs/beegfs-storage.conf | grep connInterfacesFile
connInterfacesFile            = /etc/beegfs/connInterfacesFile.conf
```

6. Start the storage services on each node and enable them to start at boot.

```
ictm1624h4:~ # systemctl start beegfs-storage.service
ictm1624h4:~ # systemctl enable beegfs-storage.service
ictm1624h4:~ # systemctl status beegfs-storage.service
```

Set Up the Client Service

To set up the client service, complete the following steps on each node where BeeGFS should be mounted:

1. Ensure that the installed `kernel-devel` package matches the current kernel version (see `uname -r`).

   Note: Because the client kernel module requires this match so that it can build correctly, this issue is probably the most commonly encountered in setting up the BeeGFS client service.

2. If you are using IB, append `BEEGFS_OPENTK_IBVERBS=1` to the `buildArgs=-j8` line in `/etc/beegfs/beegfs-client-autobuild.conf`.

```
buildArgs=-j8 BEEGFS_OPENTK_IBVERBS=1
```

   Note: If you are using the as-delivered OpenFabrics Enterprise Distribution (OFED) driver, you also must use `OFED_INCLUDE_PATH` to specify the corresponding header include path. This example uses the inbox RDMA core package that comes with the operating system. Therefore, this parameter is not included in the preceding example.

3. NetApp recommends that you specify on each client which network interface or interfaces other services can use to contact this service. Create a text file under `/etc/beegfs`, with the interface names to be used listed one per line. In this example, `ib0` and `ib1` are used. The absolute path of the file must be specified in the configuration file `/etc/beegfs-client.conf`.

```
ictm1625h2:~ # cat /etc/beegfs/connInterfacesFile.conf
ib0
ib1
ictm1625h2:~ # cat /etc/beegfs/beegfs-client.conf | grep connInterfacesFile
connInterfacesFile            = /etc/beegfs/connInterfacesFile.conf
```

4. Configure the client service:
   `# /opt/beegfs/sbin/beegfs-setup-client -m <management_node_hostname_or_ip>`.

```
ictm1625h2:~ # /opt/beegfs/sbin/beegfs-setup-client -m ictm1624h3
```

   Note: This command mounts the BeeGFS file system by default on `/mnt/beegfs`. You can change the mount point in `/etc/beegfs/beegfs-mounts.conf`.

5. Start the client services on each node and enable them to start at boot.

```
ictm1625h2:~ # systemctl start beegfs-client.service
ictm1625h2:~ # systemctl enable beegfs-client.service
ictm1625h2:~ # systemctl status beegfs-client.service
```

   Note: The first time that you start the BeeGFS client service on each node might take a few minutes because the BeeGFS client kernel module must be built (including support for IB if requested).

Verify Setup

To verify the setup, run the following commands on any client:

1. List the management node.

```
[root@ictm1625h1a ~]# beegfs-ctl --listnodes --nodetype=management --details
ictm1624h3.ict.englab.netapp.com [ID: 1]
```

```
   Ports: UDP: 8008; TCP: 8008
   Interfaces: ib0(TCP) ib2(TCP)
Number of nodes: 1
```

2. List the metadata nodes.

```
[root@ictm1625h1a ~]# beegfs-ctl --listnodes --nodetype=meta --details
ictm1624h3.ict.englab.netapp.com [ID: 1]
   Ports: UDP: 8005; TCP: 8005
   Interfaces: ib0(RDMA) ib0(TCP) ib2(RDMA) ib2(TCP)

Number of nodes: 1
Root: 1
```

3. List the storage nodes.

```
[root@ictm1625h1a ~]# beegfs-ctl --listnodes --nodetype=storage --details
ictm1624h4.ict.englab.netapp.com [ID: 1]
   Ports: UDP: 8003; TCP: 8003
   Interfaces: ib0(RDMA) ib0(TCP) ib2(RDMA) ib2(TCP)
ictm1624h5.ict.englab.netapp.com [ID: 2]
   Ports: UDP: 8003; TCP: 8003
   Interfaces: ib0(RDMA) ib0(TCP) ib2(RDMA) ib2(TCP)

Number of nodes: 2
```

4. List the client nodes.

```
[root@ictm1625h1a ~]# beegfs-ctl --listnodes --nodetype=client --details
62EB-5F208A12-ictm1625h1a.ict.englab.netapp.com [ID: 1]
   Ports: UDP: 8004; TCP: 0
   Interfaces: ib0(RDMA) ib0(TCP) ib1(TCP) ib1(RDMA)
2DA54-5F20A581-ictm1625h3.ict.englab.netapp.com [ID: 2]
   Ports: UDP: 8004; TCP: 0
   Interfaces: ib0(RDMA) ib0(TCP) ib1(TCP) ib1(RDMA)
2BB7F-5F20A581-ictm1625h5.ict.englab.netapp.com [ID: 3]
   Ports: UDP: 8004; TCP: 0
   Interfaces: ib0(RDMA) ib0(TCP) ib1(TCP) ib1(RDMA)
2BA1B-5F20A581-ictm1625h4.ict.englab.netapp.com [ID: 4]
   Ports: UDP: 8004; TCP: 0
   Interfaces: ib0(RDMA) ib0(TCP) ib1(TCP) ib1(RDMA)
1C52C-5F20A581-ictm1625h2.ict.englab.netapp.com [ID: 5]
   Ports: UDP: 8004; TCP: 0
   Interfaces: ib0(RDMA) ib0(TCP) ib1(TCP) ib1(RDMA)
2C1E0-5F20A581-ictm1625h6.ict.englab.netapp.com [ID: 6]
   Ports: UDP: 8004; TCP: 0
   Interfaces: ib0(RDMA) ib0(TCP) ib1(TCP) ib1(RDMA)
2BD5B-5F20A58D-ictm1625h7.ict.englab.netapp.com [ID: 7]
   Ports: UDP: 8004; TCP: 0
   Interfaces: ib0(RDMA) ib0(TCP) ib1(TCP) ib1(RDMA)
2BA9C-5F20A58D-ictm1625h8.ict.englab.netapp.com [ID: 8]
   Ports: UDP: 8004; TCP: 0
   Interfaces: ib0(RDMA) ib0(TCP) ib1(TCP) ib1(RDMA)
2BB0B-5F20A58E-ictm1625h11.ict.englab.netapp.com [ID: 9]
   Ports: UDP: 8004; TCP: 0
   Interfaces: ib0(RDMA) ib0(TCP) ib1(TCP) ib1(RDMA)
2BCEF-5F20A58E-ictm1625h10 [ID: 10]
   Ports: UDP: 8004; TCP: 0
   Interfaces: ib0(RDMA) ib0(TCP) ib1(TCP) ib1(RDMA)
2B7EE-5F20A58D-ictm1625h9.ict.englab.netapp.com [ID: 11]
   Ports: UDP: 8004; TCP: 0
   Interfaces: ib0(RDMA) ib0(TCP) ib1(TCP) ib1(RDMA)

Number of nodes: 11
```

5. Display the connections that the client is actually using.

```
[root@ictm1625h1a ~]# beegfs-net

mgmt_nodes
=============
ictm1624h3.ict.englab.netapp.com [ID: 1]
```

```
   Connections: TCP: 1 (192.168.3.23:8008);

meta_nodes
=============
ictm1624h3.ict.englab.netapp.com [ID: 1]
   Connections: RDMA: 1 (192.168.3.23:8005);

storage_nodes
=============
ictm1624h4.ict.englab.netapp.com [ID: 1]
   Connections: RDMA: 2 (192.168.3.24:8003);
ictm1624h5.ict.englab.netapp.com [ID: 2]
   Connections: RDMA: 2 (192.168.3.25:8003);
```

6. Verify connectivity to the different services.

```
[root@ictm1625h1a ~]# beegfs-check-servers
Management
==========
ictm1624h3.ict.englab.netapp.com [ID: 1]: reachable at 192.168.3.23:8008 (protocol: TCP)

Metadata
==========
ictm1624h3.ict.englab.netapp.com [ID: 1]: reachable at 192.168.3.23:8005 (protocol: TCP)

Storage
==========
ictm1624h4.ict.englab.netapp.com [ID: 1]: reachable at 192.168.3.24:8003 (protocol: TCP)
ictm1624h5.ict.englab.netapp.com [ID: 2]: reachable at 192.168.3.25:8003 (protocol: TCP)
```

7. Display the free space and inodes on the storage and metadata targets.

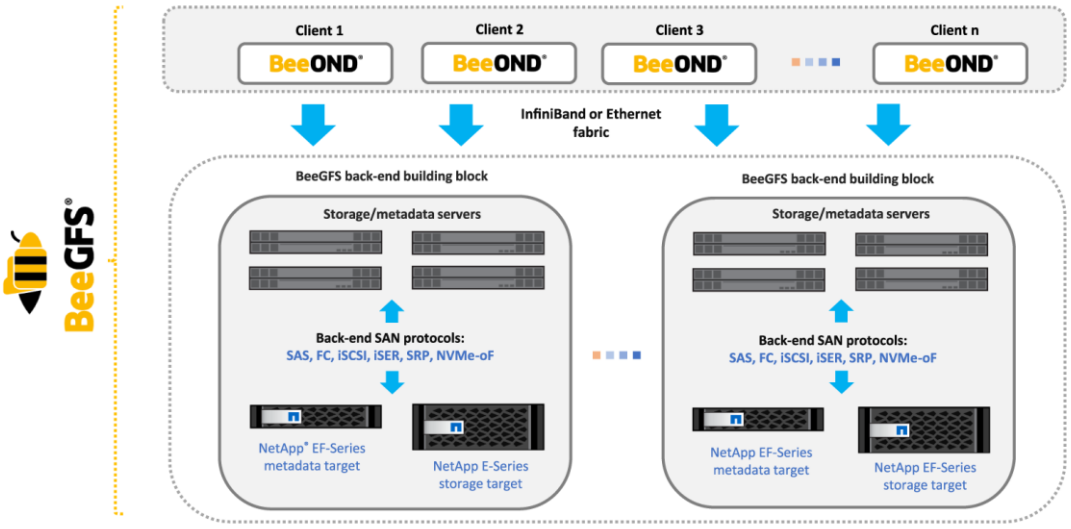
```
[root@ictm1625h1a ~]# beegfs-df
METADATA SERVERS:
TargetID   Cap. Pool        Total         Free    %      ITotal       IFree    %
========   =========        =====         ====    =      ======       =====    =
       1      normal    4648.7GiB    4648.4GiB 100%     3100.0M     3099.9M 100%

STORAGE TARGETS:
TargetID   Cap. Pool        Total         Free    %      ITotal       IFree    %
========   =========        =====         ====    =      ======       =====    =
     101      normal  109420.0GiB  109420.0GiB 100%     2188.4M     2188.4M 100%
     102      normal  109420.0GiB  109420.0GiB 100%     2188.4M     2188.4M 100%
     203      normal  109420.0GiB  109420.0GiB 100%     2188.4M     2188.4M 100%
     204      normal  109420.0GiB  109420.0GiB 100%     2188.4M     2188.4M 100%
```

3 Advanced Building Block

In the advanced building block use case, the metadata resides on a separate NetApp EF-Series all-flash array, in this case, an EF570 system. The data for the storage services remains on an E-Series HDD-based target, in this case, an E5760 system, as shown in Figure 2. The models can vary depending on your performance and capacity requirements.

Figure 2) BeeGFS advanced building block on E-Series systems.



To set up the advanced building block use case, you can use the same deployment procedures as for the basic building block use case.

4 Conclusion

This document provides instructions on how to quickly and easily set up a BeeGFS configuration with NetApp E-Series as back-end storage. These instructions focus primarily on functionality and interoperability.

This document can be referenced in any other document that addresses detailed subjects such as tuning parameters for performance or any special use cases of BeeGFS with E-Series.

Appendix A: Optimize BeeGFS Service Startup with E-Series Volumes

The information in this appendix might help you troubleshoot issues in which BeeGFS metadata or storage services are in a failed state after the OS boots. When using file systems that reside on remote mount points (such as NetApp E-Series volumes) with BeeGFS storage and metadata services, the services must start after the remote file systems are available. To achieve this order, override the `systemd` unit file for each BeeGFS metadata or storage service to add `remote-fs.target` to the `Requires` and `After` directives.

To perform this step correctly, create a directory in `/etc/systemd/system` that is named after the unit file with `.d` appended to the end and a contained `*.conf` file that specifies the directives to override. This approach prevents the custom directives from being overwritten if a package update overwrites the system's copy of the unit file. It also appends the desired values to any values that are already in place for the `Requires` or `After` directives.

Following is an example override file:

```
[root@ictm1624h3 ~]# cat /etc/systemd/system/beegfs-meta.service.d/99-eseries-beegfs_override.conf
[Unit]
Requires=remote-fs.target
After=remote-fs.target
```

You can confirm that the override takes effect by running `systemctl status beegfs-<service>`.

```
[root@ictm1624h3 ~]# systemctl status beegfs-meta.service
● beegfs-meta.service - BeeGFS Metadata Server
   Loaded: loaded (/usr/lib/systemd/system/beegfs-meta.service; enabled; vendor preset: disabled)
  Drop-In: /etc/systemd/system/beegfs-meta.service.d
           └─99-eseries-beegfs_override.conf
   Active: active (running) since Tue 2020-07-28 15:26:40 CDT; 1h 47min ago
     Docs: http://www.beegfs.com/content/documentation/
 Main PID: 61536 (beegfs-meta/Mai)
    Tasks: 155
   CGroup: /system.slice/beegfs-meta.service
           └─61536 /opt/beegfs/sbin/beegfs-meta cfgFile=/etc/beegfs/beegfs-meta.conf
runDaemonized=false

Jul 28 15:26:40 ictm1624h3.ict.englab.netapp.com systemd[1]: Started BeeGFS Metadata Server.
```

And you can verify that the directives are as expected with `systemctl show beegfs-<service>`.

```
[root@ictm1624h3 ~]# systemctl show beegfs-meta | egrep -e Requires= -e After=
Requires=remote-fs.target basic.target network-online.target
After=rdma.service network-online.target systemd-journald.socket beegfs-storage.service
system.slice opensm.service remote-fs.target beegfs-mgmtd.service openibd.service zfs.target
openib.service opensmd.service basic.target
```

Appendix B: Optimize BeeGFS Client Startup with IB

When BeeGFS clients start, they perform a mount sanity check to ensure that the client helper daemon and other BeeGFS services (especially the management service) are reachable. In some instances, if network connections are not connected before the sanity check times out, the BeeGFS client service might be in a failed state after a reboot.

In testing, it was observed with IB that the links came online (satisfying `network-online.target`, a prerequisite for `beegfs-client.service`), but then took longer to actually connect. Because of how IB subnet managers periodically poll (or sweep) the subnet for newly attached devices, some amount of delay is expected.

This poll time is typically a user-configurable rate (for Mellanox MLNX-OS switches, see `show ib sm sminfo-poll-time` and set with `ib sm sminfo-poll-time <value>`). The implications of changing this value are outside the scope of this document, however.

Following is an example IB switch configuration:

```
ictm1624sw1 [standalone: master] > enable
ictm1624sw1 [standalone: master] # show ib sm sminfo-poll-time
10.000 seconds
```

To prevent BeeGFS client services from failing after a reboot, you might have to increase the `/etc/beegfs/beegfs-client.conf:sysMountSanityCheckMS` parameter from the default 11000ms (11 seconds) value.

Following is a truncated example BeeGFS client configuration:

```
[root@ictm1625h2 ~]# cat /etc/beegfs/beegfs-client.conf

sysMountSanityCheckMS         = 60000

# [sysMountSanityCheckMS]
# Perform some basic checks during mount (e.g. whether the client helper daemon
# and storage servers are reachable). Mounting will fail if a problem is
# detected.
# Values: Set the time (in ms) you want to spend waiting for the servers
# (especially the management daemon) to respond. Use 0 to disable all checks
# and allow mounting even if no servers are reachable.
# Default: 11000
```

Where to Find Additional Information

To learn more about the information that is described in this document, see the following documents and/or websites:

- E-Series and SANtricity 11 Documentation Center
  https://docs.netapp.com/ess-11/index.jsp
- SANtricity Software Express Configuration for Linux
  http://docs.netapp.com/ess-11/index.jsp?topic=%2Fcom.netapp.doc.ssm-exp-ic-lin%2Fhome.html
- Official BeeGFS Documentation
  https://www.beegfs.io/wiki/TableOfContents

Version History

Version	Date	Document Version History
Version 1.0	March 2019	Initial release.
Version 2.0	November 2019	Corrected a few command typos and nomenclature.
Version 3.0	August 2020	Updated examples to use the latest recommendations and nomenclature. Added appendixes A and B.

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

Copyright Information

Trademark Information

TR-4755-0820