Technical Report

# BeeGFS with NetApp E-Series

## Reference Architecture

Mahmoud Jibbe, Dean Lang, Jason Hennessy, Charles Binford, Keith Holt, Mark Regester, Waleed Ghaith, Abdel Sadek, NetApp

May 2019 | TR-4782


## Abstract

The objective of this document is to define a reference architecture for BeeGFS with NetApp® E-Series storage that offers reasonable and consistent performance while delivering common best practices. This document describes how E-Series can be used to deploy a parallel file system such as BeeGFS, the benefits of the configuration's ease-of-use, and the competitiveness of the solution with other parallel file systems offered in the industry.


NetApp®

## TABLE OF CONTENTS



## LIST OF TABLES



## LIST OF FIGURES



 

# 1 Solution Overview

Storage deployments in high-performance computing (HPC) often have high-bandwidth or high-IOPS workloads that also require low and consistent I/O response times. These deployments often distribute the workload across multiple storage systems using parallel file systems to provide scalable storage performance and capacity to handle these workloads.

The open-source parallel file system, BeeGFS, offers a cost-effective solution that avoids vendor lock-in and simplifies end-user configuration.

This solution overview presents configuration details of a representative deployment of BeeGFS with NetApp E-Series storage along with performance test results.

## 1.1 Solution Technology

Parallel file systems were created to solve bottleneck issues at the file system layer for HPC workloads. BeeGFS is optimized for highly concurrent access to shared files and designed to be easy to set up. You can configure a BeeGFS file system in less than a day by following the instructions in TR-4755: BeeGFS with NetApp E-Series Solution Deployment.

Most parallel file systems have the same basic structure that includes four main services:

- **Management service.** Registers and monitors all other services.
- **Storage service.** Stores the distributed user file contents.
- **Metadata service.** Stores access permissions and striping information.
- **Client service.** Mounts the file system to access the stored data.

BeeGFS implements these services as separate packages that can be installed on the same host or on discrete hosts.

In general, parallel files systems offer better scalability as workloads grow. With BeeGFS, as more throughput is required, more metadata and storage nodes can be added to independently scale the metadata and storage services as needed.

BeeGFS uses a distributed metadata architecture, allowing clients to access the metadata services in parallel. BeeGFS distributes metadata on a per-directory basis; thus you can manage each directory through a different metadata service to balance the load across all metadata services.

The BeeGFS packages also do not require any kernel patches to function properly. The metadata, storage, and management services run in user space daemons, and the client service runs in a patchless kernel module.

# 2 Reference Architecture

The following sections provide instructions for using the reference architecture shown in Figure 1.

**Figure 1) Reference architecture.**

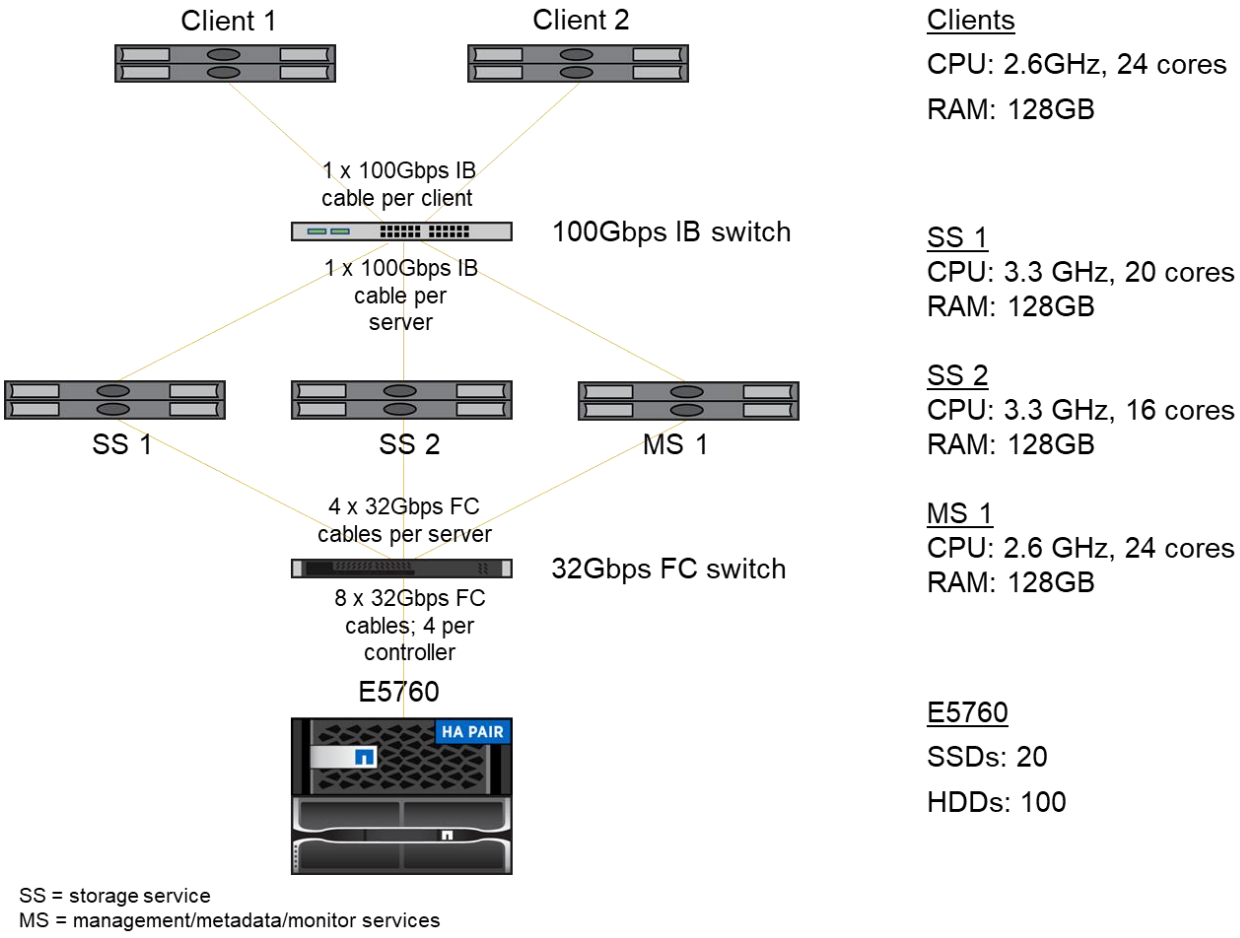


Table 1 lists the distribution of E-Series array drives.

**Table 1) E-Series array drives distribution.**

| Number of Volume Groups | Number of Volumes | RAID Type | Disk Type | Data Type |
|---|---|---|---|---|
| 1 | 11 | 2+2<br>RAID 1 | Solid-state drive (SSD) | Metadata |
| 1 | 4 | 16 disk<br>DDP | SSD | File |
| 10 | 10 | 8+2<br>RAID 6 | NL_SAS | File |

## 2.1 Test Configuration Details

This section provides detailed configuration steps and settings for the E-Series storage and all the BeeGFS components in Figure 1.

### E-Series Array Configuration

The E-Series array test configuration included the following components:

- E5760 array with dual controllers, 60 drives and one 60-drive expansion tray for a total of 120 drives. The drives were located and selected to maintain drawer loss protection. A drawer is one of the compartments of a shelf that you pull out to access the drives. Only the 60-drive high-capacity shelves have drawers. Drawer loss protection guarantees accessibility to the data on the volumes in a pool or a volume group if a total loss of communication occurs with a single drawer.
- NetApp SANtricity® OS 11.50
- Hosts type used for mapping: Linux DM-MP (Kernel 3.10 or later)
- 64GB cache per controller, 32K cache block size
- Write caching enabled, cache mirroring enabled, prefetch enabled
- One 2+2 R1 volume group (SSD); one volume for file system metadata (MD_VOL)
  - LUN mapping of metadata volume to metadata server only
  - Ten volumes to be used for external XFS logs for the bulk pool of NL_SAS HDD

    Five volumes owned by each controller, and five volumes mapped to each storage server
  - Toshiba 1.6TB (KPM51VUG1T60)
- One 16-drive SSD Dynamic Disk Pool; four volumes for data (SSD_Volume[1-4])
  - Two LUNs mapped to each storage server, two owned by each controller
  - Toshiba 1.6TB (KPM51VUG1T60)
- Ten 8+2 R6 NL_SAS HDD Volume Groups; one volume per volume group for bulk data (HDD_Volume[1-10])
  - Five LUNs mapped to each storage server, and five LUNS owned by each controller
  - Hitachi 6TB (HUS726060AL5211)

### BeeGFS Configuration

The BeeGFS test configuration included the following components:

- Operating System: RedHat Enterprise Linux (RHEL) 7.5
- Multipathing: Device Mapper Multipathing (DM-MP)
- BeeGFS version 7.1.1
- Storage for metadata service placed on MD_VOL
- SSD pool storage striped across SSD_Volume[1-4] targets
- Bulk pool storage striped across HDD_Volume[1-10] targets
- Random robin striping with 512K chunks on both pools
- Based on BeeGFS recommendations, the metadata server used an ext4 file system, and the storage servers used XFS file systems. For HDD volumes, external XFS logging feature was used to store logs on high-speed SSD.
  - XFS SSD fs creation:

```
# mkfs.xfs -f -d su=128k,sw=8 -isize=512  /dev/mapper/<devname>
```

  - XFS HDD fs creation:

```
# mkfs.xfs -f -d su=128k,sw=8 -isize=512  -l
```

```
logdev=/dev/mapper/<external_log_devname>,su=128k,size=520000b  /dev/mapper/<devname>
```

- XFS `/etc/fstab` options: rw,noatime,nodiratime,attr2,inode64,noquota,nobarrier
- ext4 fs creation:

```
# mkfs.ext4 -i 2048 -I 512 -J size=400 -Odir_index,filetype -E
lazy_itable_init=0,lazy_journal_init=0 /dev/mapper/<devname>
# tune2fs -o user_xattr /dev/mapper/<devname>
```

- ext4 `/etc/fstab` options: rw,noatime,nodiratime,data=ordered,nobarrier
- Set Linux max_sectors_kb = 1024 on storage servers with udev rule and reboot to activate it:

```
# cat /etc/udev/rules.d/S80-sys-block.rules
SUBSYSTEM!="block", GOTO="end_rule"
ENV{DEVTYPE}=="partition", GOTO="end_rule"
ACTION!="add|change", GOTO="end_rule"
KERNEL=="sd*", ATTR{queue/max_sectors_kb}="1024"
LABEL="end_rule"
# reboot
```

- After the host is up, verify the value on each SD and DM device, as shown in the following example:

```
# cat /sys/block/dm-7/queue/max_sectors_kb
1024
# cat /sys/block/sdd/queue/max_sectors_kb
1024
```

### BeeGFS .conf Changes from Defaults

In this investigation, NetApp applied the following settings to BeeGFS:

- beegfs-client.conf on client servers
  - connInterfacesFile = <specify the file>
  - connMaxInternodeNum = 24
- beegfs-storage.conf on storage servers
  - sysMgmtdHost = <host.mgmtd.ip.used>
  - storeStorageDirectory =,`/data/beegfs/ssdVol_1 ,/data/beegfs/ssdVol_2 ,/data/beegfs/hddVol_1 ,/data/beegfs/hddVol_2 ,/data/beegfs/hddVol_3 ,/data/beegfs/hddVol_4 ,/data/beegfs/hddVol_5`
  - storeAllowFirstRunInit = false
  - connInterfacesFile = <specify the file>
  - connMaxInternodeNum = 24
- beegfs-meta.conf on metadata server
  - sysMgmtdHost = <metadata.server.ip.used>
  - storeMetaDirectory = `/data/beegfs/meta`
  - tuneTargetChooser = randomrobin
  - connInterfacesFile = <specify the file>
- beegfs-mgmtd.conf on metadata server
  - storeMgmtdDirectory = `/data/beegfs/mgmt`
  - storeAllowFirstRunInit = false
  - connInterfacesFile = <specify the file>
  - tuneStorageSpaceLowLimit = 1T
  - tuneStorageSpaceEmergencyLimit = 20G
- beegfs-mon.conf on metadata server

 

- sysMgmtdHost = <metadata.server.ip.used>
- dbType = influxdb
- dbHostName = <host.influx.addr.used>
- logLevel = 3

### BIOS Changes

The BIOS changes were made on all the servers. The following procedure was used for the servers in the test configuration.

**Note:** Refer to the server manufacturer for the corresponding procedures.

1. Enable maximum performance (disable power saving modes, C1E, and C states).

```
System BIOS>>System Profile Settings>>System Profile>>Performance
```

2. Disable hyperthreading.

```
System BIOS>>Processor Settings>>Logical Processor>>Disabled
```

### Operating System Changes

Disable the CPU power saving states and set for maximum performance on all servers.

1. Add the following parameters to GRUB_CMDLINE_LINUX in `/etc/default/grub`:

```
processor.max_cstate=1 intel_idle.max_cstate=0
```

2. Generate a new config.

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
# reboot
```

3. Verify that the values took place after the reboot.

```
# dmesg | grep C-state
[    1.097864] ACPI: processor limited to max C-state 1
# dmesg | grep intel_idle:
[    1.097695] intel_idle: disabled
```

# 3 Test Results

Three common Linux-based test tools were used during the testing: IOR, MDTEST, and VDBENCH. The following subsections provide detailed results of the performance measurement.

## 3.1 IOR Tool

Commonly used in HPC environments, the Interleaved or Random (IOR) tool measures performance for sequential read/write operations with different file sizes, I/O sizes, and varying I/O concurrency. Tuning parameters enables use of single or multiple CPUs for file manipulations. IOR was used to measure sequential read and write performance for a small set of large files, with 1MB client IO sizes. Table 2 lists the IOR large file results.

**Table 2) IOR large file results.**

| Test | MiBps |
|---|---|
| Bulk pool reads | 11,500 |
| Bulk pool writes | 7,600 |

| Test | MiBps |
|---|---|
| SSD pool reads | 16,500 |
| SSD pool writes | 8,000 |

### IOR Test Parameters

The IOR large file test parameters include:

- Twelve files, 128GB each; six on each client
- 1MB I/O size
- Twelve processes; spread six on each client

### Analysis

The test analysis includes:

- The bulk pool reads are drive-limited, and the writes are approaching controller limits:
  - RAID 6 volume groups with a single volume are preferred over DDP volume groups for HPC workloads as they typically provide higher sequential read and write performance.
- The SSD pool write bandwidth number is at or near the limits of the E5760 array.
- The SSD pool read bandwidth number is limited by the number of SSDs in the volume group (16).

These test results match expectations for a single array, running either HDD or SSD separately.

## 3.2 MDTest Tool

MDTest is an application for evaluating the metadata performance of a parallel file system.

### MDTest Results

Table 3 lists the MDTest results.

**Table 3) MDTest results.**

| Operation | Maximum | Minimum | Mean | Std. Dev. |
|---|---|---|---|---|
| Directory creation | 18089.811 | 16393.441 | 17104.305 | 567.358 |
| Directory stat | 156916.758 | 149785.343 | 153373.768 | 2524.215 |
| Directory removal | 9191.082 | 8835.023 | 9027.765 | 123.785 |
| File creation | 38284.553 | 36847.249 | 37451.464 | 471.705 |
| File stat | 137700.455 | 136507.129 | 136991.339 | 449.419 |
| File read | 53236.907 | 51355.744 | 52174.543 | 666.082 |
| File removal | 41777.290 | 39654.192 | 40529.735 | 714.792 |
| Tree creation | 111.874 | 35.456 | 85.545 | 27.834 |
| Tree removal | 1.419 | 1.333 | 1.383 | 0.030 |

### MDTest Parameters

The MDTest parameters include:

- 24 processes over 480,000 files
  - Five iterations

### Analysis

The load on the E5760 controller was light during the MDTest. The array CPU use was between 5%–20%, running a single metadata service on a single array volume and disk array controller. Results with four SSDs shown Table 1 are not significantly higher than results with two SSDs. Four SSDs were used in this configuration, as opposed to two SSDs, to provide additional performance for the external XFS logs used by the HDD XFS file system data volumes. BeeGFS does allow for more than one metadata server/service. Using a second metadata service and volume on a second disk array controller can be explored for additional performance.

## 3.3 vdBench Tool

vdBench is an I/O tool commonly used to measure storage performance. vdBench was used for small-file testing because of its ease of use and the fact that vdBench provides detailed statistics associated with each run.

### Large Number of Small File Results

The next set of tests examine create, read, and write operations to a directory with 262,144 files. Each file is 4MB. vdBench was used to create the files, select files in sequential order, and sequentially read or write the files, with a client IO size of 1MB. A large number of smaller files, 4MB in this case, is a data set that might be characteristic of an AI/ML/DL application, as shown in Table 4.

**Table 4) vdBench small-file results.**

| | One BeeGFS Client | | | Two BeeGFS Clients | | |
|---|---|---|---|---|---|---|
| | Files/sec. | MBps | Avg. Resp. Time (ms.) | Files/sec. | MBps | Avg. Resp. Time (ms.) |
| **Create** | | | | | | |
| HDD | 1,200 | 4900 | 0.18 | 1,600 | 6900 | 0.27 |
| SSD | 1,400 | 5600 | 0.15 | 2,000 | 8100 | 0.22 |
| **Read** | | | | | | |
| HDD | 2,100 | 8700 | 1.66 | 2,300 | 9500 | 3.10 |
| SSD | 2,200 | 8900 | 1.63 | 2,500 | 10000 | 2.80 |
| **Write** | | | | | | |
| HDD | 1,300 | 5500 | 2.70 | 1,700 | 7000 | 4.30 |
| SSD | 1,500 | 6200 | 2.43 | 1,900 | 7800 | 3.70 |

### vdBench Parameters

vdBench small-file parameters for each active storage client include:

- **Create files.** One directory with 262,144 files, file size of 4MB each

- **Read files.** Select files sequentially, read files sequentially, 1MB IO size, 16 threads (files) active
- **Write files.** Select files sequentially, write files sequentially, 1MB IO size, 16 threads (files) active, open files with o_sync

### Analysis

A minimum of three runs were made for each test point, with the lowest result being reported. Rates are as reported by vdBench, rounded down to the nearest 100s for Files/sec. and MBps. Tests were run with one and then two storage clients. The results represent the upper end of expectations with a relatively new file system, with the two client tests. The average I/O response time difference between the create test and the write test is largely due to the o_sync flag used for the write test. Create tests without o_sync allow for more of an asynchronous write path by the client, allowing for a large dirty cache backlog on the storage servers, and a deep queue of writes to the array from the storage servers. The write test with o_sync follows more of a synchronous write path from the client to the array.

# 4 Conclusion

NetApp measured performance numbers for a specific configuration and specific workloads. This reference architecture provides high throughput levels with the IOR tests using a small set of large files that begin to approach limits of the array controllers and drives. The vdBench tests give insight into performance with high file counts and a smaller file size. Both sets of results are intended to show the upper range of performance expectations with data sets that are not highly fragmented.

# Where to Find Additional Information

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

- TR-4755: BeeGFS with NetApp E-Series Solution Deployment
  http://www.netapp.com/us/media/tr-4755.pdf
- NetApp product documentation
  https://docs.netapp.com/
- BeeGFS documentation
  http://www.beegfs.io

# Version History

| Version | Date | Document Version History |
|---|---|---|
| Version 1.0 | May 2019 | Initial release. |

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-4782-0519