HPC High-Capacity Storage Solution for BeeGFS

Dell EMC PowerEdge Servers, PowerVault Storage and PowerConnect Switches

December 2021

H19033

White Paper

Abstract

This white paper describes a solution for HPC high-capacity, high-throughput scalable storage, a **Dell Technologies Validated Design for BeeGFS**. The solution's architecture, tuning guidelines and performance are explained.

Dell Technologies Validated Design

Copyright

Contents

Introduction

Executive summary

In high-performance computing (HPC), designing a well-balanced storage system to achieve optimal performance presents significant challenges. A typical storage system consists of a variety of considerations, including file system choices, file system tuning, disk drive, storage controller, IO cards, network cards, and switch strategies. Configuring these components for best performance, manageability, and future scalability requires a great deal of planning and organization.

The Dell EMC Validated Design for HPC BeeGFS High-Capacity Storage is a fully supported, easy-to-use, high-throughput, scale-out, parallel file system storage solution with well-described performance characteristics. This solution is offered with deployment services and full hardware and software support from Dell Technologies.

The solution in large configuration with four ME4084 arrays scales to ~6 PB of raw storage and uses Dell EMC PowerEdge servers and the Dell EMC PowerVault storage arrays.

Document purpose

The purpose of this document is to describe the architecture, tuning best practices and performance of the Dell EMC Validated Design for High-Capacity Storage with BeeGFS for sequential and random workloads.

Audience

This document is intended for solution architects and administrators who are already utilizing HPC clusters for their research and scientific computing requirements. This paper is not a detailed configuration guide but does advise best practices for configuration. This paper assumes that the readers will understand the basic concepts of storage performance benchmarking.

Business challenges

Market environment

In recent years, the requirement for I/O performance has increased dramatically with increasing demand for both traditional HPC I/O bandwidth but also for increased IOPS and metadata performance. The BeeGFS filesystem has steadily gained popularity and is increasingly being used as the file system of choice in many top HPC centers today. This whitepaper demonstrates how the BeeGFS filesystem deployed on the latest generation of PowerEdge servers and PowerVault storage arrays offers high capacity, high availability and high performance that can be tailored to meet the needs of the customer.

Building-Block Approach

A highly flexible solution that provides superior performance and reliable scalability to run todays and tomorrow’s most demanding HPC applications.

Ease of Use

Optimized for use in today’s large-scale data processing systems that consist of thousands of compute nodes (BeeGFS clients) running tens of thousands of concurrent processes.

High Availability

The solution is designed to enhance the availability of storage services to the HPC cluster by using a pair of Dell EMC PowerEdge servers and PowerVault storage arrays along with Pacemaker and Corosync software. The goal of the solution is to improve storage services availability and maintain data integrity in the event of possible failures or faults, and to optimize performance in a failure-free scenario.

Solution overview

Configurations

The Dell EMC Validated Design for HPC BeeGFS Storage is available in three base configurations: Small, Medium, and Large. These base configurations can be used as building blocks to create additional flexible configurations to meet different capacity and performance goals as illustrated in Figure 1.

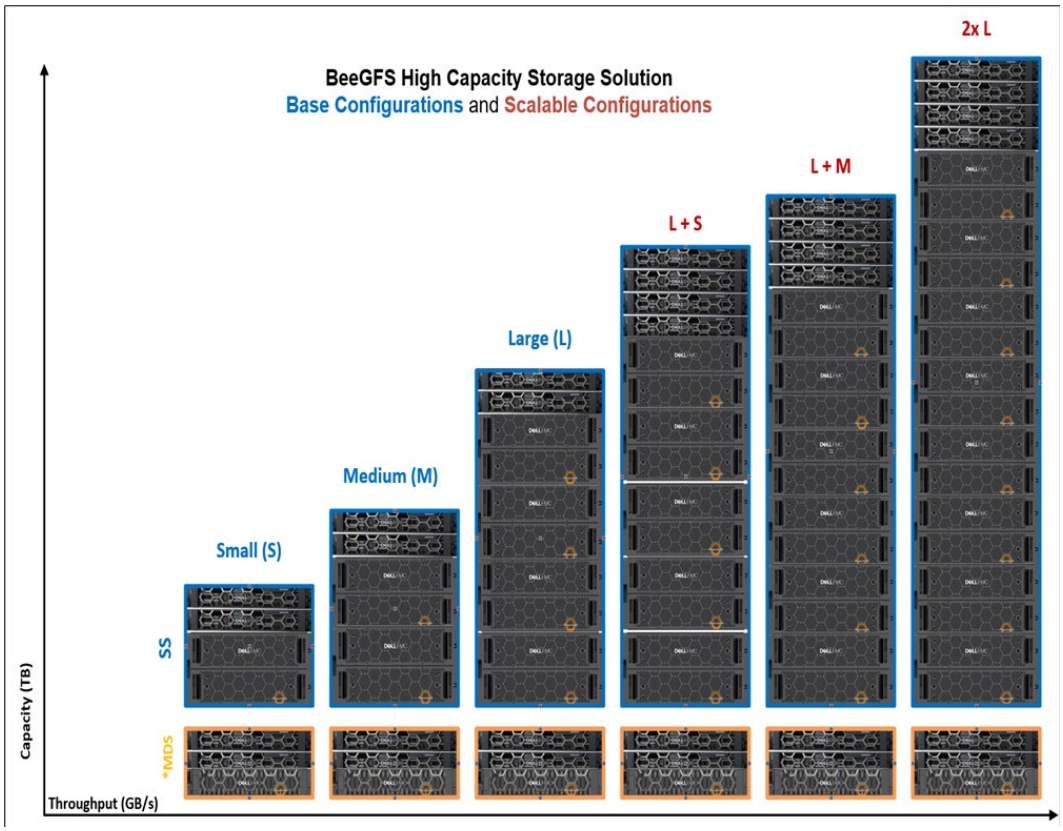


Figure 1. BeeGFS high-capacity storage solution—base and scalable configurations

The metadata component of the solution that includes a pair of metadata servers (MDS) and a metadata target storage array remains the same across all the configurations as shown in Figure 1. The storage component of the solution includes a pair of storage servers (SS) and a single storage array for the small configuration, while a medium configuration uses two storage arrays, and a large configuration uses four storage arrays. The PowerVault ME4024 storage array is used as the metadata storage, and PowerVault ME4084 arrays are used as the data storage.

To scale beyond a large configuration, additional pair of storage servers are needed. The additional storage server pair can have either one, two, or four storage arrays as indicated

in the scalable configurations. As the storage capacity requirements for BeeGFS metadata are very small, typically 0.3% to 0.5% of total storage capacity, the metadata component may either remain the same, or additional metadata modules may be added depending upon the number of directories and file entries in the local environment, you can have up to four ME4024 storage arrays including the initial metadata pair. The solution supports using additional metadata modules (servers and ME4024 arrays). Contact a Dell EMC sales representative to discuss which offering works best in your environment, and how to order.

BeeGFS File System

This storage solution is based on BeeGFS, an available source parallel file system, which offers flexibility and easy scalability. The general architecture of BeeGFS consists of four main services: management, metadata, storage, and client. The server components are implemented as user space daemons. The client is a patchless kernel module. An additional monitoring service is also available.

The key elements of the BeeGFS file system are as follows:

MetaData Targets (MDTs)

Stores all the metadata for the file system including filenames, permissions, time stamps, and the location of stripes of data.

Management service (MGMTD)

Stores management data such as configuration and all the file system components.

MetaData Server (MDS)

Server that runs the metadata services.

Storage Targets (STs)

Stores the data stripes or extents of the files on a file system in the ME4084 arrays. There can be multiple storage targets in a single storage service.

Storage Server (SS)

Server that runs the storage services.

Client Module

The BeeGFS client kernel module is installed on the clients to allow access to data on the BeeGFS file system. To the clients, the file system appears as a single namespace that can be mounted for access.

For more information on BeeGFS file system architecture, see Introduction to BeeGFS.

Solution architecture

Overview

Figure 2 shows the Large configuration architecture with four PowerVault ME4084 storage arrays.

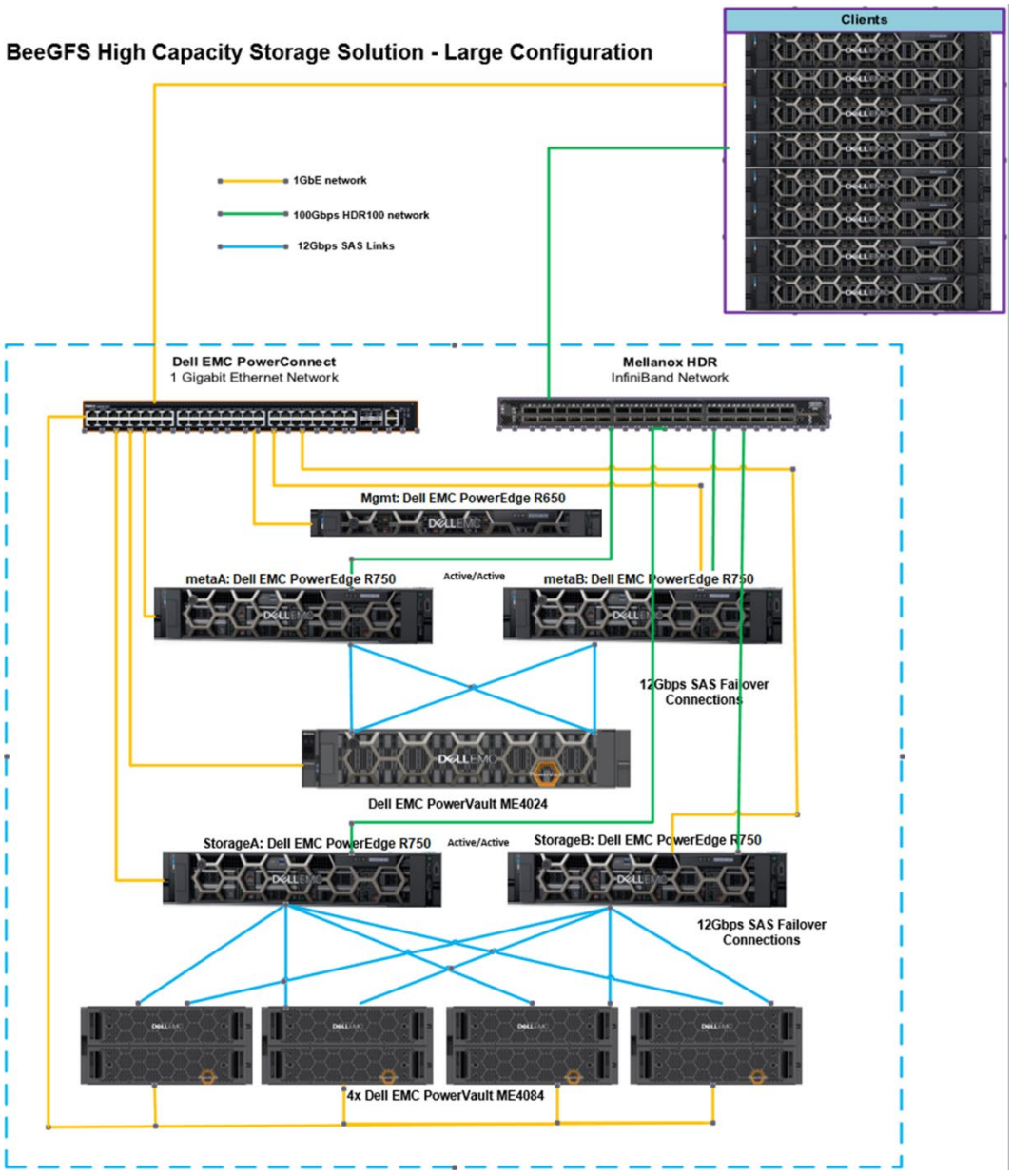


Figure 2. Solution reference architecture—Large configuration

In Figure 2, the management server (the topmost server) is a PowerEdge R650. The MDS function is provided by two PowerEdge R750 servers. The MDS pair is attached to a PowerVault ME4024 through 12 Gb/s SAS links. The PowerVault ME4024 hosts the metadata targets.

The SSs are a pair of PowerEdge R750 servers. The SS pair is attached to four fully populated PowerVault ME4084 storage arrays through 12 Gb/s SAS links. The four PowerVault ME4084 arrays host the storage targets.

The solution uses InfiniBand HDR100 as the data network connecting the storage solution to the compute clients. Gigabit Ethernet is used for management operations.

Management Server

The single management server is connected to the MDS pair and SS pair through an internal 1 GbE network.

BeeGFS provides a tool called `beegfs-mon` that collects use and performance data from the BeeGFS services and stores the data in a timeseries database called InfluxDB. `beegfs-mon-grafana` provides predefined Grafana panels that can be used out of the box to extract and visualize this data. Both Grafana and InfluxDB are installed on the management server. After starting and enabling InfluxDB and Grafana server services, Grafana dashboard can be accessed using the url http://<IP of mgmt.>:3000.

Metadata Server

The `beegfs-mgmtd` service in the BeeGFS high-capacity solution is not actually running on the management server. It is managed by Pacemaker to run on either metaA or metaB. Its storage is on the first meta partition on metaA server and will move together with `beegfs-mgmtd` service in case of a service failure. The `beegfs-mgmtd` service is set up on both metadata servers and is started on the metaA server. The `beegfs mgmtd` store is initialized on the directory mgmtd on the metadata target 1 as follows:

```
/opt/beegfs/sbin/beegfs-setup-mgmtd -p /beegfs/metaA-numa0-1/mgmtd
-S beegfs-mgmt
```

The two PowerEdge R750 servers that are used as the MDS are directly attached to the PowerVault ME4024 storage array housing the BeeGFS MDTs and MGMTD. Figure 3 shows the SAS ports on ME4024 array.

Figure 3. ME4024 SAS ports

Metadata Targets

The ME4024 array is fully populated with 24× 960 GB SAS SSDs. An optimal way to configure the 24 drives for metadata is to configure twelve MDTs. Each MDT is a RAID 1 disk group of two drives each. Figure 4 shows how the MDTs are configured.

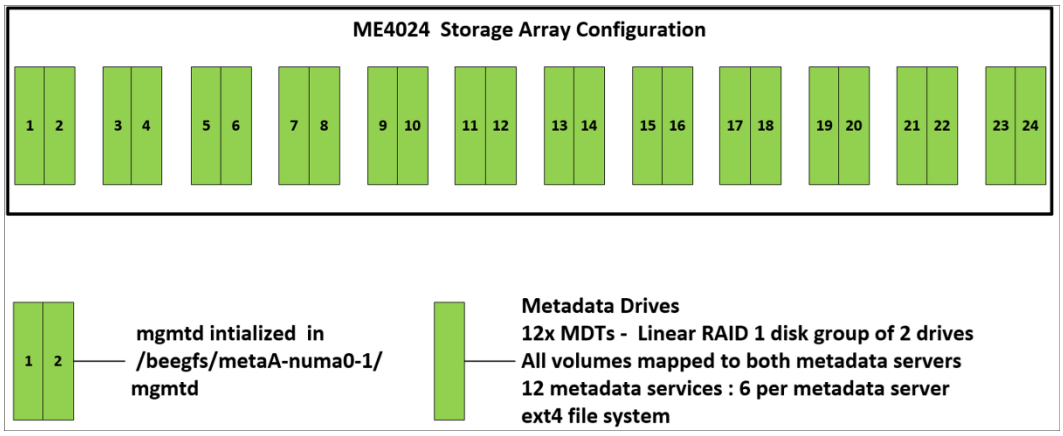


Figure 4. Configuration of metadata targets in the ME4024 storage array

The management service runs on the metadata servers and is initially setup on the metaA server. The metadata target is formatted with the ext4 file system because it outperforms other file systems with small file I/O and handles storing metadata as extended attributes inside the inode more efficiently.

Storage Servers

Each PowerEdge R750 server in the SS pair is equipped with two quad-port 12 Gb/s SAS host bus adapters and two NVIDIA Mellanox InfiniBand HDR100 adapters to handle storage requests. This allows the SAS HBAs to be evenly distributed across the two CPUs for load balancing. With two quad-port 12 Gb/s SAS controllers in each PowerEdge R750, the two servers are redundantly connected to each of the four PowerVault ME4084 high-density storage arrays, with a choice of 4 TB, 8 TB, 12 TB or 16 TB of NL SAS 7.2 K CPURPM hard disk drives (HDDs) or any supported NL SAS drive as defined in the support matrix https://dl.dell.com/topicspdf/me4-series-sm_en-us.pdf. Figure 5 shows the SAS ports on the ME4084 array.

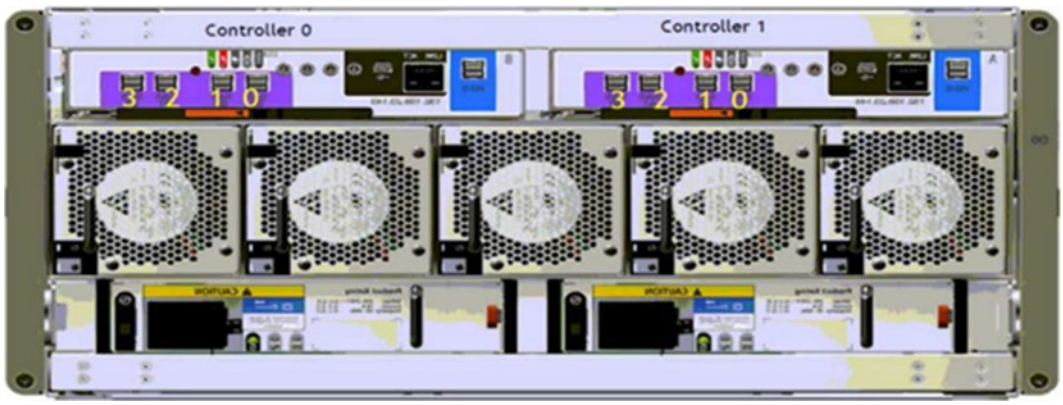


Figure 5. ME4084 SAS ports

Storage Targets

Figure 6 illustrates how each storage array is divided into eight linear RAID 6 disk groups of ten drives each with eight data and two parity disks per virtual disk.

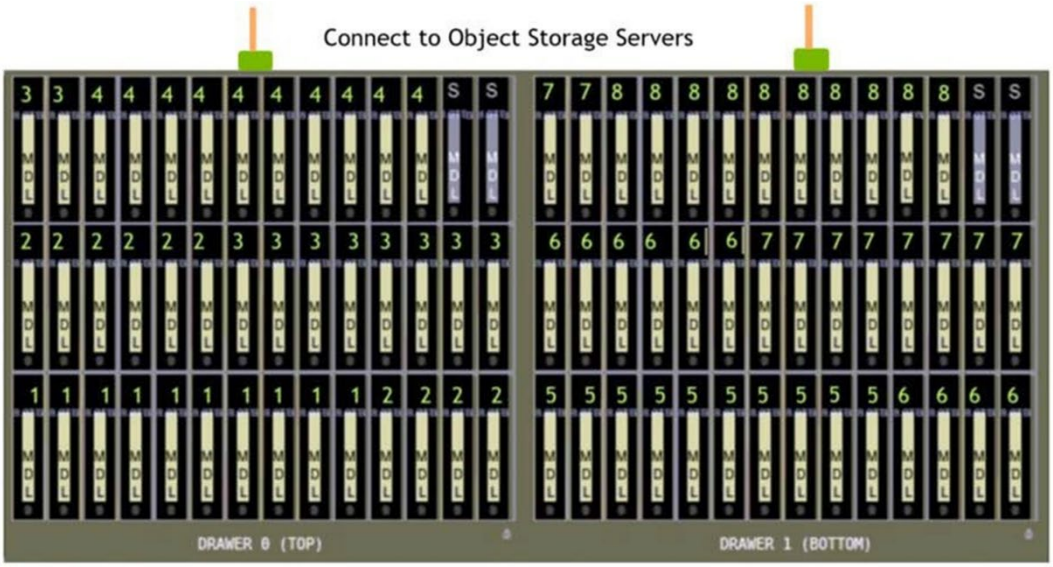


Figure 6. RAID 6 (8+2) LUNs layout on one ME4084

Each ST provides about 64 TB of formatted object storage space when populated with 8 TB HDDs. Since each array has 84 drives, after creating eight RAID-6 disk groups, we have 4 spare drives per array, 2 per tray, which can be configured as global hot spares across the 8 disk groups in the array. For every disk group, a single volume using all the RAID's capacity is created. As a result, a large base configuration as shown in Figure 2 has a total of 32 linear RAID 6 volumes across four ME4084 storage arrays. Each of these RAID 6 volumes are configured as an ST for the BeeGFS file system, resulting in a total of 32 STs across the file system in the base configuration.

The STs are presented to clients via InfiniBand HDR100 connection. From any compute node that is equipped with the BeeGFS client, the entire namespace can be accessed and managed like any other file system.

Hardware and Software Configuration

Table 1 describes the hardware and software details of the solution.

Table 1. Solution hardware and software specifications

Component	Specification
Management server	1× Dell EMC PowerEdge R650
Metadata Servers	2× Dell EMC PowerEdge R750
Storage servers	2× Dell EMC PowerEdge R750
Processors	Management Server, MDS and SS: 2× Intel Xeon Gold 6326 2.9G, 16 cores
Memory	Management Server, MDS and SS: 16x 16GB RDIMM, 3200MT/s, Dual Rank
Local disks and RAID controller	Management Server, MDS and SS: PERC H345 Integrated RAID Controller, 2× 960GB 12Gbps SAS SSDs configured in RAID1 for OS

Component	Specification
InfiniBand HCA	2× NVIDIA Mellanox ConnectX-6 Single Port HDR100 Adapter per MDS and SS with firmware version 20.28.4512
External storage controllers	2× Dell HBA355e Adapter per MDS and SS
Object storage enclosures	4× Dell EMC PowerVault ME4084 fully populated with a total of 336 drives
Metadata storage enclosure	1× Dell EMC PowerVault ME4024 with 24 SSDs
RAID controllers	Duplex RAID controllers in the ME4084 and ME4024 enclosures
Hard Disk Drives	On each ME4084 Enclosure: 84× 8 TB 3.5 in. 7.2 K RPM NL SAS3 ME4024 Enclosure: 24× 960 GB SAS3 SSDs
Operating system	Red Hat Enterprise Linux release 8.3
Kernel version	4.18.0-147.5.1.el8_1.x86_64
NVIDIA Mellanox OFED	5.4-1.0.3
BeeGFS file system version	7.2.4
Grafana	8.1.5-1
InfluxDB	1.8.9-1
Pacemaker	2.0.4-6
Corosync	3.0.3-4
Management Switch	Dell Networking N3224T-ON
InfiniBand Switch	NVIDIA QM8790 Quantum HDR 200 Gb/s Edge Switch—1U with firmware version 27.2008.2500

High-availability testing

Mechanisms to Handle Failure

There are many distinct types of failures and faults that can impact the functionality of a highly available BeeGFS storage solution. Table 2 lists the potential failures that are tolerated in the solution.

Table 2. High availability—mechanisms to handle failure

Failure type	Mechanism to handle failure
Single local Operating System disk failure on a server	Operating system installed on a RAID1 virtual device (two disks).
Power supply or power bus failure	Each server has dual redundant PSUs, and each PSU must be connected to a separate power bus. The server can continue to be operational with a single PSU.
InfiniBand link/Adapter failure	Two physical interfaces configured in an active-backup logical bonded interface. When the active link fails, the backup link takes over. When the active adapter fails, the passive takes its place

Failure type	Mechanism to handle failure
SAS cable / SAS port failure	Two SAS HBA cards installed on each MDS/SS. Each card has a SAS cable to a different SAS controller in the shared ME4084 and the Operating System multipath service manages all SAS paths available, keeping only one active per LUN. A single SAS card or cable failure will not impact data availability, but performance may be reduced depending on I/O load.
Single server failure	Event monitored by pacemaker and corrosion services. In case of a failure, the services failover to its peer server. (Reason for hardware redundancy)
HDD failures in the ME arrays	The storage targets are in RAID 6 disk group of 10 drives (8+2). Four global spares have been configured. So automatic RAID rebuild takes place in case of a disk failure.
Single ME4084 SAS controller failure	If a single ME4084 SAS controller fails, the remaining ME4 controller takes over the I/O transactions (leveraging shared cache among controllers), ownership of disk groups, volumes, SAS connections, etc. Multipath services detect the link failures (using ALUA) and instructs I/O to utilize only the connections to the remaining controller. Performance may be degraded depending on the I/O load.

Performance evaluation

Factors

There are many factors involved while determining file system performance. Some of these factors include:

Network bottlenecks: Careful network design is required to maximize overall throughput.

Number of client systems: Number of clients and storage servers need to be balanced to achieve best performance.

Type of data access: Most parallel filesystems treat file metadata (e.g., file names, directory structure, permissions, etc.) differently to the file contents. Having dedicated metadata servers and storage servers can lead to better throughput when performing different file operations (e.g., read, write, copy, mkdir, etc.).

Benchmarks Built-in with BeeGFS

Before evaluating performance using benchmark tools, one should determine the theoretical maximum performance that can be achieved from the given storage solution. For this purpose, benchmarks built-in with the BeeGFS file system, storage targets benchmark (StorageBench) and network benchmark (NetBench) are used.

Storage Benchmark

The StorageBench benchmark measures the streaming throughput of the underlying file system and devices independent of the network performance. This benchmark is started and monitored with the `beegfs-ctl` tool which is provided by the `beegfs-utils` package. To simulate the client IO, this benchmark generates read/write operations locally on the servers without any client communication.

The following example starts a write benchmark on all targets of all BeeGFS storage servers with an IO block size of 1m, using 32 threads per target, each of which will write 20GB of data to its own file.

```
# beegfs-ctl --storagebench --alltargets --write --blocksize=1m --
size=20G --threads=32

Write storage benchmark was started.
You can query the status with the --status argument of beegfs-ctl.

Server benchmark status:
Running:      2
```

To query the benchmark status/result of all the targets, the following command is executed:

```
# beegfs-ctl --storagebench --alltargets --status --verbose

Server benchmark status:
Finished:     2

Write benchmark results:
Min throughput:             702107  KiB/s   nodeID: storageB [ID:
2], targetID: 23
Max throughput:             710685  KiB/s   nodeID: storageB [ID:
2], targetID: 17
Avg throughput:             706768  KiB/s
Aggregate throughput:     22616601  KiB/s

List of all targets:
1                         707421  KiB/s   nodeID: storageA [ID: 1]
2                         705866  KiB/s   nodeID: storageA [ID: 1]
3                         705662  KiB/s   nodeID: storageA [ID: 1]
4                         705947  KiB/s   nodeID: storageA [ID: 1]
5                         707342  KiB/s   nodeID: storageA [ID: 1]
6                         707568  KiB/s   nodeID: storageA [ID: 1]
7                         705928  KiB/s   nodeID: storageA [ID: 1]
8                         707023  KiB/s   nodeID: storageA [ID: 1]
9                         705649  KiB/s   nodeID: storageA [ID: 1]
10                        709182  KiB/s   nodeID: storageA [ID: 1]
11                        707138  KiB/s   nodeID: storageA [ID: 1]
12                        705762  KiB/s   nodeID: storageA [ID: 1]
13                        706228  KiB/s   nodeID: storageA [ID: 1]
14                        706330  KiB/s   nodeID: storageA [ID: 1]
15                        708324  KiB/s   nodeID: storageA [ID: 1]
16                        705950  KiB/s   nodeID: storageA [ID: 1]
17                        710685  KiB/s   nodeID: storageB [ID: 2]
18                        710042  KiB/s   nodeID: storageB [ID: 2]
19                        709987  KiB/s   nodeID: storageB [ID: 2]
20                        710467  KiB/s   nodeID: storageB [ID: 2]
21                        702424  KiB/s   nodeID: storageB [ID: 2]
22                        702673  KiB/s   nodeID: storageB [ID: 2]
23                        702107  KiB/s   nodeID: storageB [ID: 2]
24                        702184  KiB/s   nodeID: storageB [ID: 2]
25                        706246  KiB/s   nodeID: storageB [ID: 2]
26                        707289  KiB/s   nodeID: storageB [ID: 2]
27                        706776  KiB/s   nodeID: storageB [ID: 2]
28                        705810  KiB/s   nodeID: storageB [ID: 2]
```

```
29                          709858  KiB/s   nodeID: storageB [ID: 2]
30                          708166  KiB/s   nodeID: storageB [ID: 2]
31                          706664  KiB/s   nodeID: storageB [ID: 2]
32                          707903  KiB/s   nodeID: storageB [ID: 2]
```

From the output we can infer that the theoretical maximum write performance that can be achieved is **23.2 GB/s** and that the storage targets and connections are properly configured.

The following example starts a read benchmark on all targets of all BeeGFS storage servers with an IO block size of 1m

```
# beegfs-ctl --storagebench --alltargets --read --blocksize=1m --
size=20G --threads=32

Read storage benchmark was started.
You can query the status with the --status argument of beegfs-ctl.

Server benchmark status:
Running:      2

# beegfs-ctl --storagebench --alltargets --status --verbose
Server benchmark status:
Finished:     2

Read benchmark results:
Min throughput:             669365  KiB/s   nodeID: storageA [ID:
1], targetI
D: 7
Max throughput:             811450  KiB/s   nodeID: storageA [ID:
1], targetI
D: 8
Avg throughput:             763821  KiB/s
Aggregate throughput:     24442299  KiB/s

List of all targets:
1                           770210  KiB/s   nodeID: storageA [ID: 1]
2                           808646  KiB/s   nodeID: storageA [ID: 1]
3                           784148  KiB/s   nodeID: storageA [ID: 1]
4                           808984  KiB/s   nodeID: storageA [ID: 1]
5                           756048  KiB/s   nodeID: storageA [ID: 1]
6                           784366  KiB/s   nodeID: storageA [ID: 1]
7                           669365  KiB/s   nodeID: storageA [ID: 1]
8                           811450  KiB/s   nodeID: storageA [ID: 1]
9                           752802  KiB/s   nodeID: storageA [ID: 1]
10                          678202  KiB/s   nodeID: storageA [ID: 1]
11                          774487  KiB/s   nodeID: storageA [ID: 1]
12                          755134  KiB/s   nodeID: storageA [ID: 1]
13                          757666  KiB/s   nodeID: storageA [ID: 1]
14                          753994  KiB/s   nodeID: storageA [ID: 1]
15                          736778  KiB/s   nodeID: storageA [ID: 1]
```

```
16                          751701  KiB/s   nodeID: storageA [ID: 1]
17                          738865  KiB/s   nodeID: storageB [ID: 2]
18                          745544  KiB/s   nodeID: storageB [ID: 2]
19                          753347  KiB/s   nodeID: storageB [ID: 2]
20                          748829  KiB/s   nodeID: storageB [ID: 2]
21                          766946  KiB/s   nodeID: storageB [ID: 2]
22                          765260  KiB/s   nodeID: storageB [ID: 2]
23                          768562  KiB/s   nodeID: storageB [ID: 2]
24                          769207  KiB/s   nodeID: storageB [ID: 2]
25                          790460  KiB/s   nodeID: storageB [ID: 2]
26                          763252  KiB/s   nodeID: storageB [ID: 2]
27                          783895  KiB/s   nodeID: storageB [ID: 2]
28                          772880  KiB/s   nodeID: storageB [ID: 2]
29                          787025  KiB/s   nodeID: storageB [ID: 2]
30                          773996  KiB/s   nodeID: storageB [ID: 2]
31                          784955  KiB/s   nodeID: storageB [ID: 2]
32                          775295  KiB/s   nodeID: storageB [ID: 2]
```

From the output we can infer that the theoretical maximum write performance that can be achieved is **25.0 GB/s.**

The generated files will not be automatically deleted when the benchmark is complete. The files can be deleted using the following command:

```
# beegfs-ctl --storagebench --alltargets –cleanup
```

NetBench Benchmark

The NetBench mode is intended for network streaming throughput benchmarking. When the NetBench mode is enabled, data is not actually written to an actual storage device. This is because, in this mode, the write requests that are transmitted over the network from the client to the storage servers, are not submitted to the underlying file system. Instead, the write requests get discarded. Similarly, in case of a read request, instead of reading from the underlying file system on the servers, only memory buffers are sent to the clients. Consequently, the NetBench mode is independent of the underlying disks and can be used to test the maximum network throughput between the clients and the storage servers.

Before starting the benchmarking using IOzone, the NetBench tool was used to benchmark the solution's overall network performance. Enable NetBench mode on clients as shown below:

```
"echo 1 > /proc/fs/beegfs/<client ID>/NetBench_mode"
```

Note: When you have multiple BeeGFS file systems mounted on a client, make sure that the NetBench mode is enabled only on the appropriate file system of relevance. The following command can be used to identify the client node ID for a given management node.

```
beegfs-ctl --listnodes --nodetype=client --sysMgmtdHost=10.10.218.200
```

Provided below is the partial output of the NetBench benchmark results for the Large configuration of the BeeGFS High-Capacity Storage Solution with 4x PowerVault Storage Arrays.

```
# cat NetBench-iozone_results_write_int1_th256_st1.101421114219
```

```
START: 1634226139
0 1m 16g 256 /home/testing/benchmarking/hosts.16334
        Iozone: Performance Test of File I/O
                Version $Revision: 3.492 $
                Compiled for 64 bit mode.
                Build: linux

        Run began: Thu Oct 14 11:42:19 2021

        Include close in write timing
        Include fsync in write timing
        Setting no_unlink
        Record Size 1024 kB
        O_DIRECT feature enabled
        File size set to 16777216 kB
        No retest option selected
        Network distribution mode enabled.
        Command line used: /home/testing/bin/iozone -i 0 -c -e -w
-r 1m -I -s 16g -t 256 -+n -+m
/home/testing/benchmarking/hosts.16334
        Output is in kBytes/sec
        Time Resolution = 0.000001 seconds.
        Processor cache size set to 1024 kBytes.
        Processor cache line size set to 32 bytes.
        File stride size set to 17 * record size.
        Throughput test with 256 processes
        Each process writes a 16777216 kByte file in 1024 kByte
records

Test running:
Children see throughput for 256 initial writers=24105020.27 kB/sec
Min throughput per process                    = 76254.54 kB/sec
Max throughput per process                    = 103449.27 kB/sec
Avg throughput per process                    = 94160.24 kB/sec
Min xfer                                      = 12367872.00 kB
iozone test complete.
END: 1634226409
```

The actual theoretical network performance from this solution which has two HDR100 InfiniBand adapters is 25 GB/s and the maximum achievable performance from the existing network infrastructure is **24.1 GB/s.**

IOzone Benchmark

After ascertaining the theoretical maximum performance that can be achieved from the storage solution by running the StorageBench and NetBench benchmarks, the next step is to run the IOzone Benchmarks for performance characterization of the I/O between the clients and the storage arrays. Our performance studies of the solution use InfiniBand HDR100 data networks. Performance testing objectives were to quantify the capabilities of the solution, identify performance peaks, and determine the most appropriate methods for scaling.

We generally try to maintain a standard and consistent testing environment and methodology. In some areas we purposely optimized server or storage configurations and took measures to limit caching effects.

Large Base Configuration

We performed the tests on the solution configuration described in Table 1. The following table details the client test bed that we used to provide the I/O workload.

Table 3. Client configuration

Component	Specification
Operating system	Red Hat Enterprise Linux release 8.3 (Ootpa)
Kernel version	4.18.0-240.el8.x86_64
Servers	5× Dell EMC PowerEdge R840 11× Dell EMC PowerEdge C6420
InfiniBand Adapter	1× ConnectX-6 100Gb/s HDR100 adapter card with firmware version 20.31.1014
NVIDIA Mellanox OFED version	5.2-2.2.0.0
BeeGFS	7.2.4

Our performance analysis focused on the following two key performance characteristics:

- Throughput, data sequentially transferred in GB/s
- I/O operations per second (IOPS)

The goal was a broad but accurate overview of the capabilities of the solution using the InfiniBand HDR100 interconnect. We used IOzone benchmarks for performance characterization. IOzone uses an N-to-N file-access method. N-to-N load was tested, where every thread of the benchmark (N clients) writes to a different file (N files) on the storage system. For examples of the commands that we used to run these benchmarks, see Appendix A Benchmark command reference.

We ran each set of tests on a range of clients to test the scalability of the solution. The number of simultaneous physical clients involved in each test ranged from a single client to sixteen clients. The number of threads per node corresponds to the number of physical compute nodes, up to sixteen. The total number of threads above sixteen were simulated by increasing the number of threads per client across all clients. For instance, for 128 threads, each of the 16 clients ran eight threads.

To prevent inflated results due to caching effects, we ran the tests with a cold cache. Before each test started, the BeeGFS file system under test was remounted. A sync was performed, and the kernel was instructed to drop caches on all the clients and BeeGFS servers (MDS and SS) with the following commands:

```
sync && echo 3 > /proc/sys/vm/drop_caches
```

In measuring the solution performance, we performed all tests with similar initial conditions. The file system was configured to be fully functional, and the targets tested were emptied of files and directories before each test.

IOzone Sequential N-N Reads and Writes

To evaluate sequential reads and writes, we used IOzone benchmark version 3.492 in the sequential read and write mode. We conducted the tests on multiple thread counts, starting at two threads and increasing in powers of two to 256 threads. Because this test works on one file per thread, at each thread count, the number of files equal to the thread count were generated. The threads were distributed across 16 physical client nodes in a round-robin fashion.

We converted throughput results to GB/s from the KB/s metrics that were provided by the tool. Except for the single thread count, for which 1 TB was used as aggregate file size, for all other thread counts, an aggregate file size of 8 TB was chosen to minimize the effects of caching from the servers. Within any given test, the aggregate file size used was equally divided among the number of threads. A record size of 1 MB was used for all runs. Operating system caches were also dropped or cleaned on the client nodes between tests and iterations and between writes and reads.

The commands used for Sequential N-N tests are given below:

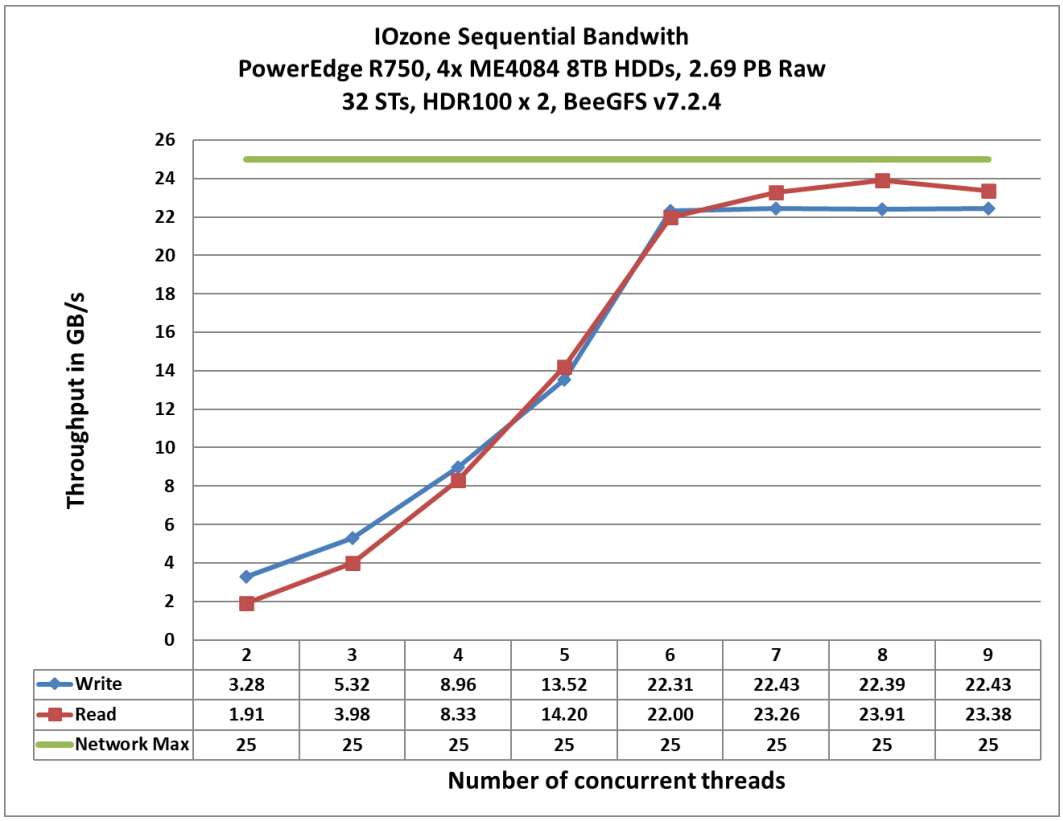
Sequential Writes and Reads: `iozone -i 0 -i 1 -c -e -w -r 1m -s $Size -t $Thread -+n -+m /path/to/threadlist`

For BeeGFS, the default chunksize is 512K and stripe count is four. However, the chunk size and the number of targets per file (stripe count) can be configured on a per-directory or per-file basis. For all these tests, BeeGFS the chunksize was set to 1 MB and stripe count was set to 1 as shown below:

```
$beegfs-ctl --setpattern --numtargets=1 --chunksize=1m
/mnt/beegfs/benchmark

$beegfs-ctl --getentryinfo --mount=/mnt/beegfs/
/mnt/beegfs/benchmark/ --verbose

Entry type: directory

EntryID: 1-5E72FAD3-1

ParentID: root

Metadata node: metaA-numa0-1 [ID: 1]

Stripe pattern details:

+ Type: RAID0

+ Chunksize: 1M

+ Number of storage targets: desired: 1

+ Storage Pool: 1 (Default)
```

```
Inode hash path: 61/4C/1-5E72FAD3-1
```

Figure 7 shows the sequential N-N performance of the solution:



	2	3	4	5	6	7	8	9
Write	3.28	5.32	8.96	13.52	22.31	22.43	22.39	22.43
Read	1.91	3.98	8.33	14.20	22.00	23.26	23.91	23.38
Network Max	25	25	25	25	25	25	25	25

Figure 7. Sequential N-N reads and writes

As the figure shows, the peak read throughput of 23.91 GB/s was attained at 128 threads. The peak write was 22.43 GB/s at 256 threads. The read and write performance scale linearly with the increase in the number of threads until the system attained its peak. After this, we see that reads and writes saturate as we scale. This brings us to understand that the overall sustained performance of this configuration for reads is ≈ 23GB/s and that for the writes is ≈ 22 GB/s with the peaks as mentioned above. The reads are very close to or slightly higher than the writes, independent of the number of the threads used.

IOzone Random Reads and Writes

As described in the IOzone sequential N-N reads and writes, stripe count of 1 and chunk size of 1 MB was used. The files written were distributed evenly across the STs (round-robin) to prevent uneven I/O loads on any single SAS connection or ST in the same way that a user would expect to balance a workload.

The request size was set to 4KiB. Performance was measured in I/O operations per second (IOPS). The operating system caches were dropped between the runs on the BeeGFS servers. The file system was unmounted and remounted on clients between iterations of the test.

The command used for random read and write tests is as follows:

```
iozone -i 2 -w -c -O -I -r 4K -s $Size -t $Thread -+n -+m
/path/to/threadlist
```

The following figure shows the random read and write performance:

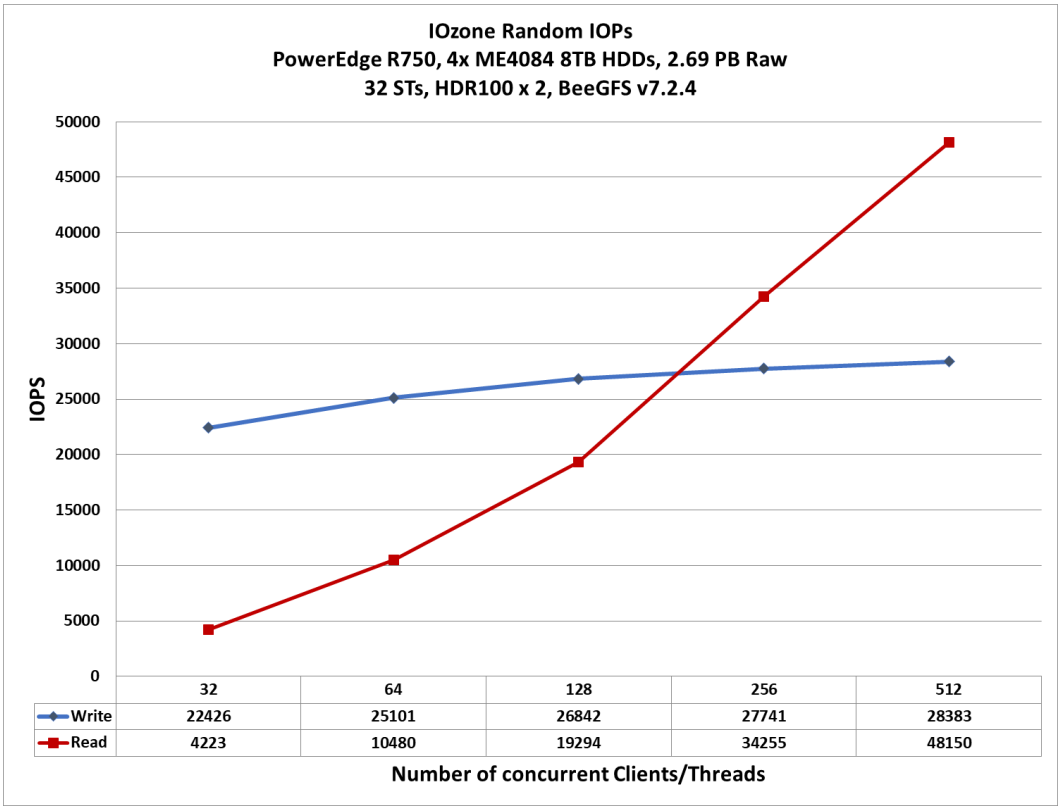


Figure 8. Random N-N reads and writes

As the figure shows, the write performance reaches at 28K IOPS and the read performance of around 48K IOPS at 512 threads, which is the maximum number of threads tested for the solution. ME4 requires a higher queue depth to reach the maximum read performance and the graph indicates that the performance may continue to increase with more than 512 threads.

Performance tuning

Parameters

This section shows the tuning parameters that we configured on the BeeGFS testbed system in the Dell HPC and AI Innovation Lab.

- Set the number of processes for root to 50000 to improve performance.

```
echo ulimit -n 50000 >> /etc/security/limits.conf
```

- Tuned the IO scheduler settings for the storage block devices on the storage servers by adding the following lines to `/etc/rc.local and` make `/etc/rc.local` executable afterwards:

```
for mdev in /dev/mapper/storage*; do
dev=$(basename $(readlink -f "$mdev"))
echo "$dev"
echo deadline > /sys/block/${dev}/queue/scheduler
```

```
echo 2048 > /sys/block/${dev}/queue/nr_requests
echo 4096 > /sys/block/${dev}/queue/read_ahead_kb
echo 256 > /sys/block/${dev}/queue/max_sectors_kb
done
$ chmod +x /etc/rc.local
```

• Tuned the IO scheduler settings for the metadata block devices on the metadata servers by adding the following lines to /etc/rc.local and make /etc/rc.local executable afterwards:

```
for mdev in /dev/mapper/storage*; do
dev=$(basename $(readlink -f "$mdev"))
echo "$dev"
echo deadline > /sys/block/${dev}/queue/scheduler
echo 128 > /sys/block/${dev}/queue/nr_requests
echo 128 > /sys/block/${dev}/queue/read_ahead_kb
echo 256 > /sys/block/${dev}/queue/max_sectors_kb
done
$ chmod +x /etc/rc.local
```

- Disabled transparent huge pages by creating /`etc/tmpfiles.d/90-beegfs-hugepages.conf` file with the following content:

```
# cat /etc/tmpfiles.d/90-beegfs-hugepages.conf
# Recommended configuration for BeeGFS servers

# Disable transparent hugepages
# Type  Path                          Mode    UID     GID     Age Argument
    w   /sys/kernel/mm/transparent_hugepage/khugepaged/defrag  - - - -    0
    w   /sys/kernel/mm/transparent_hugepage/defrag - - - -       never
    w   /sys/kernel/mm/transparent_hugepage/enabled  -  - -   -  never
```

- Tuned virtual memory settings by adding the following lines to /etc/sysctl.d/90-beegfs.conf:

```
# VM ratios recommended for BeeGFS
vm.dirty_background_ratio = 5
vm.dirty_ratio = 20
vm.min_free_kbytes = 262144
vm.vfs_cache_pressure = 50
```

- The following BeeGFS specific tuning parameters were used in the metadata, storage, and client configuration files:

beegfs-meta.conf

```
connMaxInternodeNum = 64
tuneNumWorkers = 12
tuneUsePerUserMsgQueues = true # Optional
```

```
tuneTargetChooser = roundrobin (benchmarking)
```

beegfs-storage.conf

```
connMaxInternodeNum = 64
tuneNumWorkers = 12
tuneUsePerTargetWorkers = true
tuneUsePerUserMsgQueues = true # Optional
tuneBindToNumaZone = 0
tuneFileReadAheadSize = 2m
```

beegfs-client.conf

```
connMaxInternodeNum = 24
connRDMABufNumber = 22
connRDMABufSize = 32768
```

Note: The `tuneTargetChooser` parameter was set to `roundrobin` for the purpose of benchmarking so that the targets are chosen in a deterministic, round-robin fashion. However, in a production system, it is recommended to use the “randomized” algorithm which chooses the targets in a random fashion.

Conclusion

Summary

The Dell EMC Validated Design for HPC BeeGFS High-Capacity Storage is a high-performance clustered file system solution that is easy to manage, fully supported, and capable of scaling both throughput and capacity. The solution includes the PowerEdge server platform, PowerVault ME4 storage products, and BeeGFS technology. The Large configuration of the solution with four ME4084 arrays and 2.69 PB of raw storage space, has shown to sustain a sequential throughput of approximately 22 GB/s, which is consistent with the needs of HPC environments.

We value your feedback

Dell Technologies and the author of this document welcome your feedback on the solution and the solution documentation. Contact the Dell Technologies Solutions team by email.

Author: Nirmala Sundararajan

Note: For links to additional documentation for this solution, see the Dell Technologies Solutions Info Hub for High-Performance Computing.

References

Dell Technologies documentation

The following Dell Technologies documentation provides additional and relevant information. Access to these documents depends on your login credentials. If you do not have access to a document, contact your Dell Technologies representative.

- *Dell EMC Ready Solutions for HPC BeeGFS High-Capacity Storage*
- *Dell EMC ME4 Series Storage System Administrator's Guide*

ThinkParQ documentation

The following ThinkParQ documentation provides additional and relevant information:

- *BeeGFS Documentation*
- *General Architecture of BeeGFS File System*
- *Evaluating the MetaData Performance of BeeGFS*

Appendix

IOzone Benchmark Tool

This appendix provides information about the IOzone benchmark tool that was used to measure sequential N to N read- and write throughput (GB/s) and random read- and write I/O operations per second (IOPS).

The following sub sections provide the command line reference and describe the various options used in the commands to run the respective benchmarks.

IOzone Reference and Options

The IOzone tests were N-to-N. Meaning, N client threads would read or write N independent files. The command used to run the IOzone benchmarks are given below:

IOzone sequential tests

Sequential Writes

```
iozone -i 0 -c -e -w -r 1m -I -s $Size -t $Thread -+n -+m
/path/to/threadlist
```

Sequential Reads

```
iozone -i 1 -c -e -w -r 1m -I -s $Size -t $Thread -+n -+m
/path/to/threadlist
```

By using -c and -e in the test, IOzone provides a more realistic view of what a typical application is doing.

IOzone Random Writes/ Reads

```
iozone -i 2 -w -c -O -I -r 4K -s $Size -t $Thread -+n -+m
/path/to/threadlist
```

The `O_Direct` command line parameter allows us to bypass the cache on the compute node on which we are running the IOzone thread. The following table describes IOzone command line arguments.

Table 4. IOzone command-line arguments

IOzone argument	Description
-i 0	Write test
-i 1	Read test
-i 2	Random Access test
-+n	No retest
-c	Includes close in the timing calculations
-t	Number of threads
-e	Includes flush in the timing calculations
-r	Records size
-s	File size
-t	Number of threads

IOzone argument	Description
-+m	Location of clients to run IOzone when in clustered mode
-w	Does not unlink (delete) temporary file
-I	Use O_DIRECT, bypass client cache
-O	Give results in ops/sec