HPC Scratch Storage with BeeGFS

BeeGFS in a high-performance configuration on Dell EMC PowerEdge servers with PowerSwitch and InfiniBand networking

October 2021

White Paper

Abstract

This document describes the **Dell Technologies Validated Design for BeeGFS, storage in the high-performance configuration.** The document describes the design components, performance characterization and scalability.

Dell Technologies Validated Design

DELL Technologies

Copyright

The information in this publication is provided as is. Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any software described in this publication requires an applicable software license.

Copyright ©2021 Dell Inc. or its subsidiaries. All Rights Reserved. Dell Technologies, Dell, EMC, Dell EMC and other trademarks are trademarks of Dell Inc. or its subsidiaries. Intel, the Intel logo, the Intel Inside logo and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries. Other trademarks may be trademarks of their respective owners. Published in the USA 10/21 White Paper.

Dell Inc. believes the information in this document is accurate as of its publication date. The information is subject to change without notice.

Contents

Introduction

Executive summary

The Dell Technologies Validated Design for HPC BeeGFS Storage is available as a scratch solution — a staging ground for transient data that is usually not held beyond the lifetime of the job. Many types of jobs generate large amount of data while the job is running, but that data is discarded when the job completes. These jobs do not perform all their computations purely in memory. Each of the worker threads loads an initial configuration and reads/writes many intermediate points to disk. In such scenarios, the scratch space is often considered as a limiting factor. It is often too small or too slow. The BeeGFS high-performance storage solution has been designed to tackle this problem where the multimode job will generate many large intermediate files, on the order of terabytes. It is particularly useful in the following storage cases where the Intermediate data thus generated:

- Can be replaced easily in case of data loss
- Won't fit in the home directory quota
- Requires high speed or intensive access for heavy computation
- Would generate unreasonable network bandwidth if kept on a file server
- Job does not delete any files created in scratch before exiting

The Validated Design for HPC BeeGFS Storage is delivered as an all-inclusive storage solution and is available with deployment services and full hardware and software support from Dell Technologies. The solution is designed using standards based HPC components and focuses on ease of deployment and ease of use.

Document purpose

The purpose of this document is to describe the design of the Dell Technologies Validated Design for BeeGFS including the solution configuration, software, and application versions. The capabilities of the system are quantified by presenting the benchmark performance for IOzone sequential N-N read and write throughput (GB/s) and random read and write I/O operations per second (IOPS), IOR N-1 performance and metadata tests. An extensive appendix with details on the benchmarks used and the testing methodology adopted is included at the end of the document.

Audience

This document is intended for HPC system architects and IT managers who understand the basic concepts of storage performance benchmarking.

What's new?

Updates to hardware and software

The BeeGFS storage solution was introduced to the Dell Technologies HPC storage portfolio in November 2019. Along with the current version, three versions of BeeGFS high-performance storage solution have been released since 2019. The table below lists the high-performance storage solutions released to-date and the salient features.

Table 1. BeeGFS High-Performance Storage Solutions from Dell Technologies

Component	Initial Release (Nov 2019)	HDR100 Refresh (Nov 2020)	HDR Refresh (Nov 2021)
Server	PowerEdge R740xd	PowerEdge R740xd	PowerEdge R750
NVMe drives	24x 1.6 TB P4600 Mixed Use NVMe U.2 PCIe G en3	24x 3.2 TB P4610 Mixed Use NVMe U.2 PCIe Gen3	16x 1.6 TB P5600 Dell Express Flash NVMe U.2 PCIe Gen4
Connection from drive to processor	NVMe Switch adapter with 16 PCIe Lanes	NVMe Switch adapter with 16 PCIe Lanes	CPU direct attach (direct cable connection from NVMe ports on system board of the server)
InfiniBand Adapters	2x ConnectX-5 Single Port EDR	2x ConnectX-6 Single Port HDR100	2x ConnectX-6 Single Port HDR
InfiniBand Switch	SB7890 InfiniBand EDR 100 Gb/s Switch -1U (36x EDR 100 Gb/s ports)	QM8790 Quantum HDR Edge Switch – 1U (80x HDR100 100 Gb/s ports using splitter cables)	QM8790 Quantum HDR Edge Switch – 1U (40x HDR 200 Gb/s ports)
Operating System	CentOS 7.6	CentOS 7.6	RHEL 8.3
Kernel Version	3.10.0-957.27.2.el7.x86_64	4.18.0-193.14.2.el8_2.x86_64	4.18.0-240.el8.x86_64
BeeGFS file system	7.1.3	7.2	7.2.3
NVIDIA OFED version	4.5-1.0.1.0	5.0-2.1.8.0	5.4-1.0.3.0

For the CPU direct-attach as well as the x16 adapter configuration, each NVMe SSD in the system has the bandwidth of four PCIe lanes.

Design approach

As the design goal is performance, the solution uses the latest PowerEdge R750 server equipped with 16x 1.6TB P5600 mixed use U.2 Gen4 NVMe drives and 2x NVIDIA® ConnectX-6 HDR adapters.

The Validated Design for BeeGFS high-performance storage solution uses a flexible building block approach, where individual building blocks can be combined to offer various configurations to meet customer-specific workloads and use cases and to provide for future expansion.

The solution focuses on performance, flexibility and scalability and can deliver more than 161 GB/s with a sequential reads workload and 83.8 GB/s with a sequential write workload for a Small+1 configuration.

Design concepts

Overview

The Validated Design for BeeGFS is composed of four main services:

- Management Service: Maintains a list of all other BeeGFS services and their state
- Storage Service: Stores user file contents
- Metadata Service: Stores file metadata like access permissions and striping information
- Client Service: Mounts the file system to access the stored data

BeeGFS separates the metadata service from user file chunks on the servers. The file chunks are provided by the storage service and contain the data that users want to store. The moment a client has got the metadata for a specific file or directory, it can talk directly to the storage service to store or retrieve file chunks. So, there is no further involvement of metadata service in read or write operations.

BeeGFS supports multi-mode which means it is possible for us to run multiple instances of each of its services on a single machine.

No dedicated Metadata Server

The unique feature of BeeGFS is its distributed metadata architecture which allows for scaling of performance for metadata operations. Each metadata service instance has exactly one metadata target to store its data. BeeGFS metadata is very small and grows linearly with the number of user-created files. 512GB of metadata capacity is usually good for about 150 million user files. So, *there is no dedicated metadata server in the configuration.* Instead, the eight drives in the NUMA zone 0 of one of the servers are used as the metadata targets, with metadata services hosted in that NUMA zone. The remaining eight drives in NUMA zone 1 of the server are used as storage targets with storage services hosted in that NUMA zone.

Bind Services to respective NUMA zones

We are binding the metadata service to NUMA zone 0 and storage service to NUMA zone 1 on the server that cohosts the metadata and storage targets. The storage services are also pinned to respective the NUMA zone where their respective NVMe drives and IB interfaces are connected This eliminates UPI inter-CPUs communications (effectively partitioning the hardware by NUMA zone) and minimizes the problem of system bus contention by increasing the number of paths between CPU and RAM. CPUs typically contend only with CPUs within their NUMA node for access to RAM rather than with all the CPUs on a system.

Balanced Configuration

Each server is equipped with 2x ConnectX-6 HDR adapters. Each CPU is connected to one InfiniBand interface and handles eight NVMe devices each. Such a balanced configuration provides maximum performance by ensuring that both the processors are equally occupied in handling the I/O requests to and from the NVMe drives, and their work is completely independent from the components point of view.

Solution requirements

Physical environment

The physical environment consists of the following components:

- One PowerEdge R650 (runs BeeGFS Management Service and Monitoring Service)
- Six PowerEdge R750 each with 16x NVMe P5600 U.2 1.6TB and two InfiniBand ConnectX-6 HDR Adapters

  The ConnectX-6 InfiniBand adapters are installed on Slots 3 &6 as shown below:

InfiniBand Devices

Status	Name	Product Name
✓	InfiniBand Slot 3	Mellanox ConnectX-6 Single Port VPI HDR QSFP Adapter - 1C:34:DA:54:BF:88
✓	InfiniBand Slot 6	Mellanox ConnectX-6 Single Port VPI HDR QSFP Adapter - 1C:34:DA:54:BB:98

- One Dell EMC PowerSwitch N3248TE-ON (Management network switch)
- One NVIDIA Quantum QM8790 HDR switch

The servers have the BIOS settings described in the blog Intel Ice Lake – BIOS Characterization for HPC. However, the servers have Sub-NUMA cluster disabled on them.

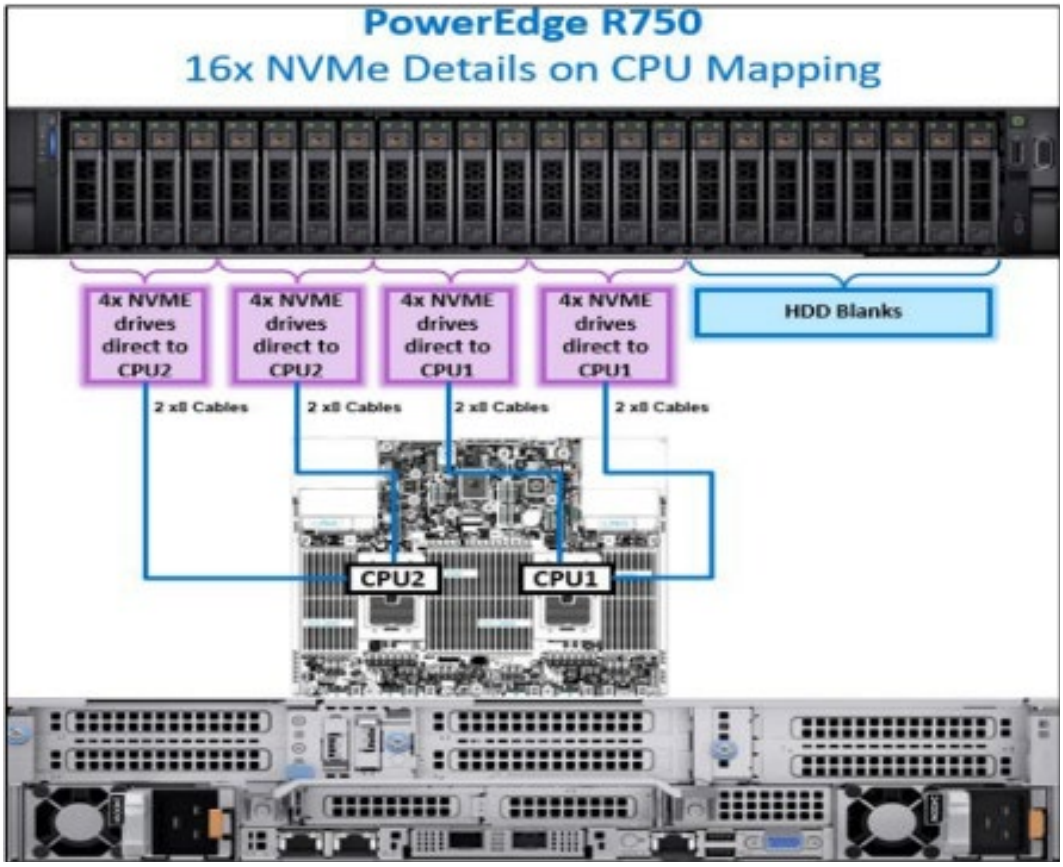


Figure 1. NVMe CPU mapping on PowerEdge R750 server

Figure 1 shows the NVMe CPU mapping on the PowerEdge R750 servers in which eight NVMe drives connect directly to CPU1 and the remaining eight NVMe drives connect directly to CPU2. This configuration is suitable when moderate amount of NVMe storage is enough to meet the requirements.

The `lstopo` command can be used to show PCIe connectivity and to visualize the various NUMA nodes installed on the system. The `hwloc` package contains the `lstopo` binary.

As the performance impact of NUMA misses is significant, the services are configured to use specific NUMA zones to avoid unnecessary use of UPI cross-socket links, thereby reducing latency. This NUMA separation is achieved manually by configuring NUMA balancing by creating custom systemd unit files and by configuring multihoming. Hence the automatic NUMA balancing is disabled, as shown below:

```
#  echo 0 > /proc/sys/kernel/numa_balancing
```

The same can be verified by running the following command:

```
# cat /proc/sys/kernel/numa_balancing

0
```

Vendor requirements

BeeGFS is an open-source parallel file system based on POSIX file system interface. It includes enterprise features such as

- High Availability
- Quota Enforcement
- Access Control Lists (ACLs)
- Storage Pools
- Burst buffer function with BeeOND

The BeeGFS client module is licensed under the General Public License (GPL) version 2.0. All other BeeGFS components are licensed under the terms of the BeeGFS End User License Agreement.

Solution design

Physical architecture

Figure 2 shows the design of the solution. The management server is only connected via Ethernet to the metadata and storage servers. Each metadata and storage server have two InfiniBand links and is connected to the private network via Ethernet. The clients have one InfiniBand link and are connected to the private interface via Ethernet.

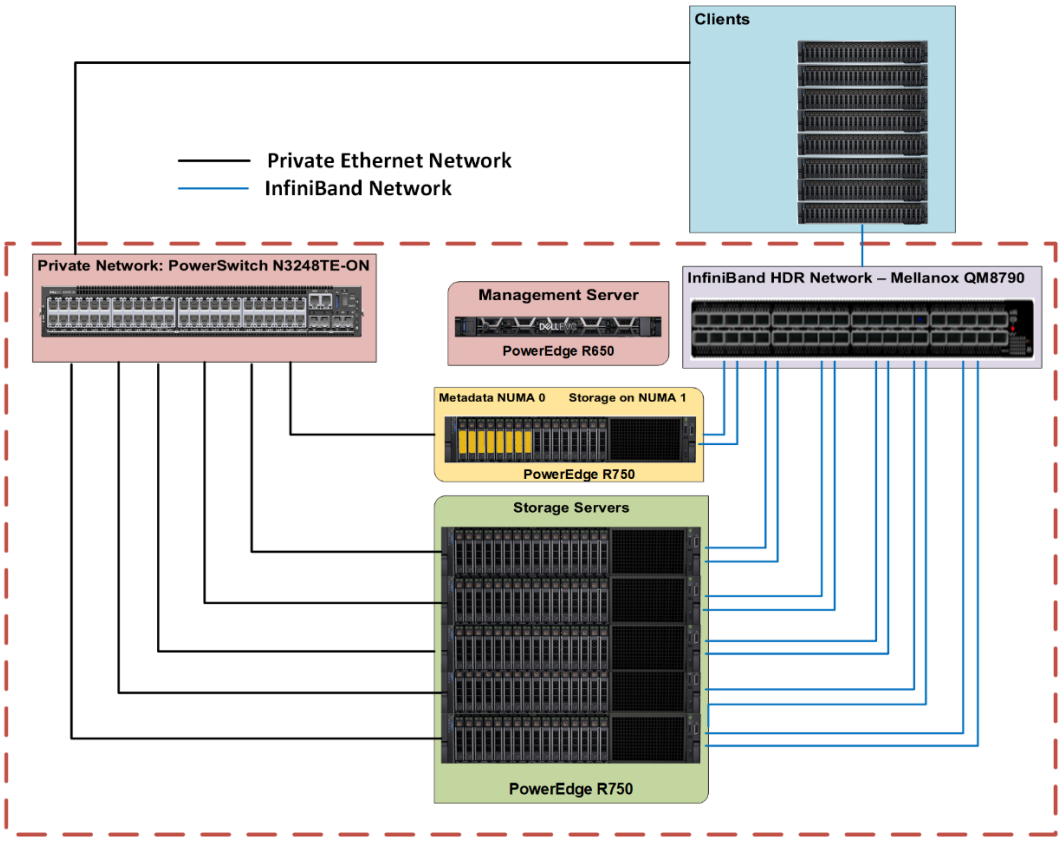


Figure 2. Dell Technologies Validated Design for BeeGFS High-Performance

This is the “Medium” configuration which uses five dedicated storage servers. Other configurations with varying number of storage servers are also available as described in the *Sizing Guidelines Section* of this document.

Figure 3 shows the network connection in which the InfiniBand connections to the NUMA zones are highlighted. Each server has two IB links and the traffic through NUMA 0 zone is handled by interface `ib0` while the traffic through NUMA zone 1 is handled by `ib1`. The BeeGFS services are also configured to use only the corresponding IB interface to bind the services to the respective NUMA zone.

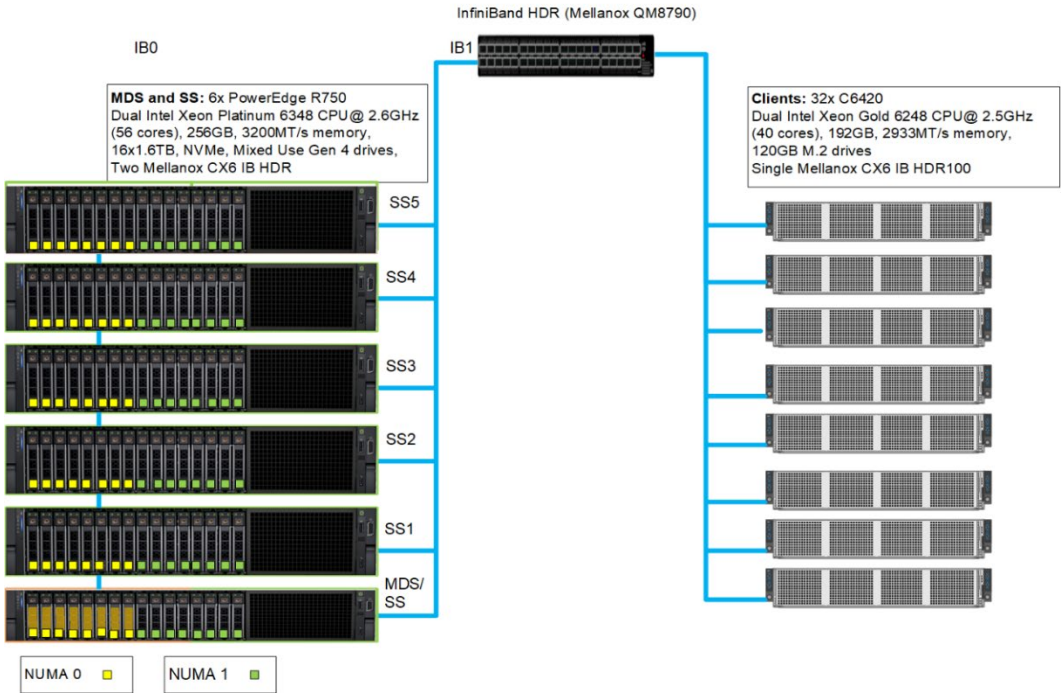


Figure 3. Network connections for the BeeGFS High-Performance solution

Management server

A PowerEdge R650 is used as the management server. In addition to hosting the management service (*beegfs-mgmtd.service*), it also hosts the monitoring service (*beegfs-mon.service*) which collects statistics from the system and provides them to the user, using the time series database InfluxDB. For visualization of data, *beegfs-mon-grafana* provides predefined Grafana panels that can be used out-of-the-box. The management server has 2x 480 GB Mixed Use SATA SSDs configured in RAID 1 for the operating system and InfluxDB.

Metadata server

A PowerEdge R750 with 16x P5600 1.6 TB NVMe drives is used for metadata storage. As the storage capacity requirements for BeeGFS metadata are small, instead of using a dedicated metadata server, only the eight drives on NUMA zone 0 were used to host the Metadata Targets (MDTs), while the remaining eight drives on NUMA zone 1 host Storage Targets (STs). Figure 4 shows the metadata server. The eight drives enclosed in the yellow rectangle are the MDTs in the NUMA zone 0 whereas the eight drives in the NUMA zone 1 are STs. This configuration not only avoids NUMA issues but also provides enough metadata storage to facilitate scaling the capacity and performance as needed.

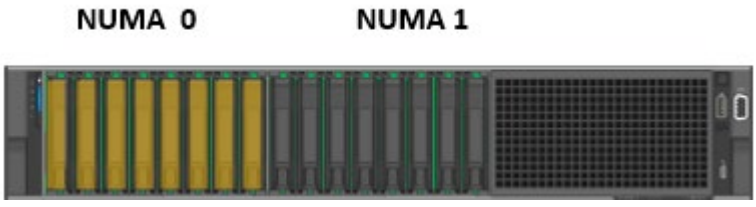


Figure 4. Metadata server

The RAID 1 configurations of the metadata server are shown in Figure 5.

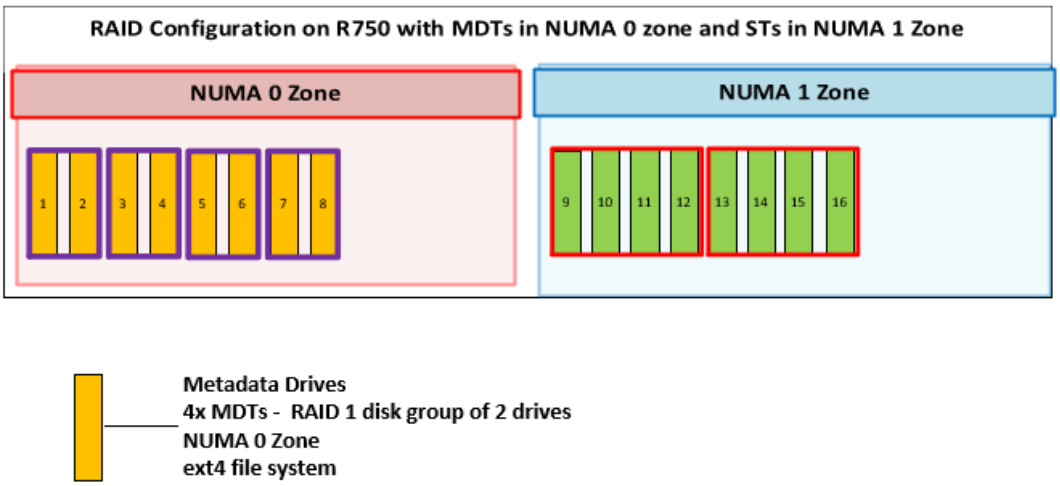


Figure 5. Configuration of drives on the metadata servers

The eight drives used for metadata are configured as 4x RAID 1 disk group of two drives, each serving as an MDT. There are four metadata services running each of which handles one MDT. The remaining eight storage drives are configured in 2x RAID 0 disk groups of four drives each. There are two storage services running on the NUMA 1 zone, one service for each ST. So, the server which co-hosts the metadata and Storage Targets has four MDTs and two STs. It runs four metadata services and two storage services. Each MDT is an ext4 file system on top of the RAID 1 configuration. The STs are based on XFS file system configured on the RAID 0.

Storage server

Figure 6 shows the 5x PowerEdge R750 servers used as dedicated storage servers.

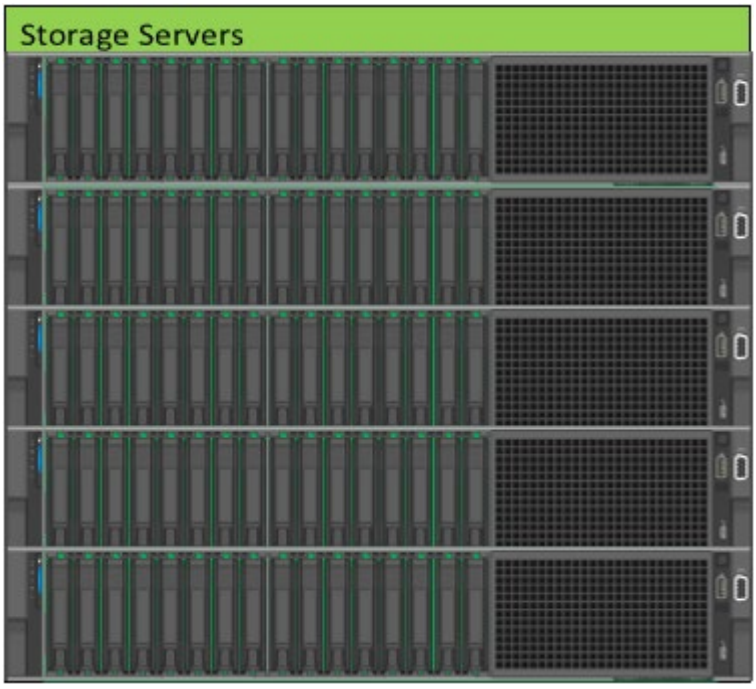


Figure 6. Dedicated storage servers

Each storage server has four storage targets, two per NUMA zone. In total, there are 22 storage targets in the Medium configuration with six servers. The targets are configured

like the STs on the Metadata Server. The NVMe drives are configured in RAID 0 disk groups of four drives each and XFS is used as the underlying file system for the beegfs-storage services. Figure 7 shows the RAID configuration on the storage servers.

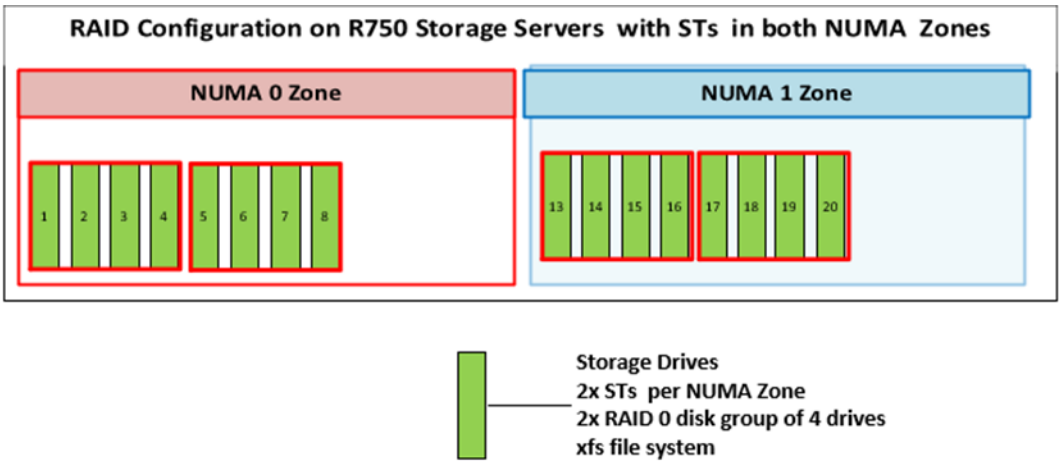


Figure 7. Configuration of drives on the storage servers

Clients

Sixteen C6420 servers were used as clients. The BeeGFS client module must be loaded on to all the hosts that need to access the BeeGFS file system. When the beegfs-client service is started, it mounts the file systems defined in the /`etc/beegfs/beegfs-mounts.conf` file instead of the usual approach based on `/etc/fstab`. Adopting this approach starts the beegfs-client like any other Linux service through the service startup script. It also enables the automatic recompilation of the BeeGFS client module after system updates. It is possible to mount multiple beegfs instances on the same client as shown below:

```
$ cat /etc/beegfs/beegfs-mounts.conf
/mnt/beegfs-medium /etc/beegfs/beegfs-client-medium.conf
/mnt/beegfs-small /etc/beegfs/beegfs-client-small.conf
```

Base configurations

Figure 8 shows the two base configurations that have been tested and validated extensively at the Dell Technologies HPC & AI Innovation Lab.

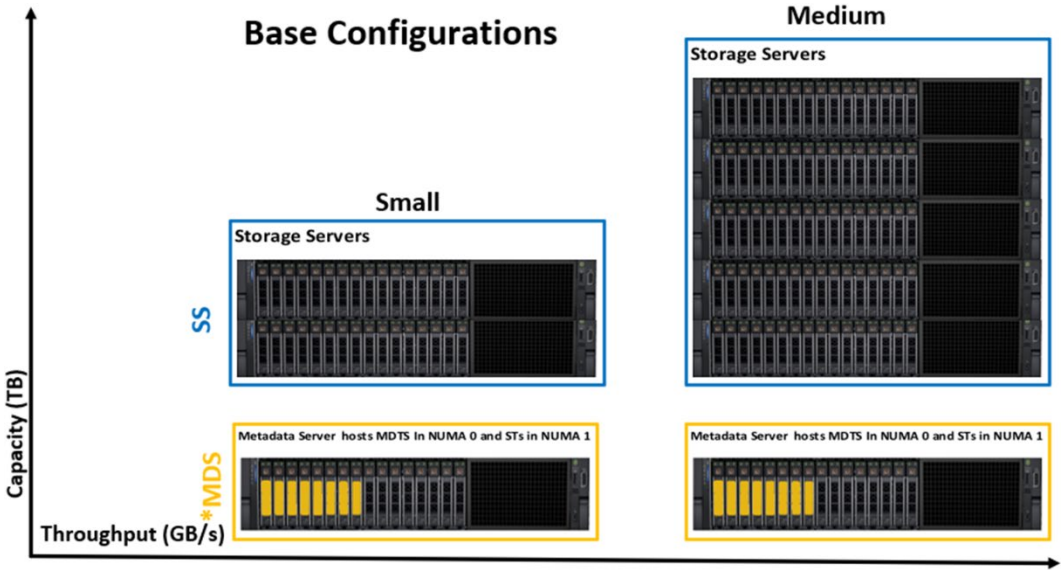


Figure 8. Base configurations

The small configuration consists of three R750 servers with a total of 10 storage targets. The medium configuration has six R750 servers and has a total of 22 storage targets. The user can start with a “Small” configuration or with the “Medium” configuration and can add storage or metadata servers as needed to increase storage space and overall performance, or number of files and metadata performance, respectively.

Hardware

Table 2 and Table 3 describe the hardware specifications of the management server and metadata/storage server respectively. Table 4 describes the other components used in the solution.

Table 2. PowerEdge R650 configuration (management server)

Component	Details
Server Model	PowerEdge R650
Processor	2 x Intel Xeon Gold 6330 CPU @ 2.00GHz, 28 cores each
Memory	16 x 8GB DDR4 3200MT/s RDIMMs – 128 GB
Local Disks	2 x 480GB 2.5in SAS SSDs in RAID 1
RAID Controller	PERC H745 Integrated RAID Controller
Out of Band Management	iDRAC9 Enterprise with Lifecycle Controller
Power Supplies	Dual Power Supply Units

Table 3. PowerEdge R750 configuration (metadata and storage servers)

Component	Details
Server Model	PowerEdge R750
Processor	2x Intel Xeon Gold 6348 CPU @2.6G, 28C
Memory	16 x 16GB DDR4 3200MT/s RDIMMs - 256GB

Component	Details
BOSS Card	2x 240GB M.2 SATA SSDs in RAID 1 for OS
Local Drives	16x 1.6TB Enterprise NVMe Mixed Use AG Drive U.2 Gen4.0 x4
NVIDIA EDR card (slots 1 & 8)	2x NVIDIA ConnectX-6 HDR 200Gb/s InfiniBand HCA
Out of Band Management	iDRAC9 Enterprise with Lifecycle Controller
Power Supplies	Dual Power Supply Units

Table 4. Other components

Component	Details
Private Gigabit Ethernet Switch	Dell PowerSwitch N3248TE-ON, 48x1G, 4x10G SFP+, 32GB, 1xAC PSU, IO/PS, OS10
InfiniBand Switch	NVIDIA QM8790 HDR switch with 40x HDR 200Gb/s ports

Note: Instead of the unmanaged NVIDIA QM8790 switch system used in the configuration, the managed QM8700 switch system may also be used. QM8700/8790 switches systems provide the highest performing fabric solution in a 1U form factor. The difference between the two switch systems is that QM8790 comes without the CPU management ports and are managed using firmware tools whereas QM8790 includes a CPU that runs the management software and has management ports, which are used to transfer management traffic into the system.

Software

Table 5 describes the software and firmware versions used for the solution.

Table 5. Software and firmware versions

Component	Details
Operating System	Red Hat Enterprise Linux 8.3
Kernel Version	4.18.0-240.el8.x86_64
BIOS	1.1.3
CPLD	1.02
1.6TB P5600 NVMe SSDs FW version	1.0.0
iDRAC	4.40.23.00
BeeGFS	7.2.3
NVIDIA MLNX_OFED	5.4-1.0.3.0
NVMe SSDs FW version	1.0.0
iDRAC	4.40.23.00
InfluxDB	1.8.6-1
NVIDIA EDR card (slots 1 & 8)	2x NVIDIA ConnectX-6 HDR 200Gb/s InfiniBand HCA

Component	Details
System Management	OMSA 9.5.0-4063
IOzone Benchmark	3.492
IOR and metadata Benchmark	3.3.0

Validation

Overview

The performance characterization was carried out on the Small configuration described in the Base Configurations section. This “Small” configuration uses 3x PowerEdge R750 servers with has 4 metadata targets and 10 storage targets. Performance testing objectives were to quantify the capabilities of the solution, identify performance peaks, and determine the most appropriate methods for scaling. We ran multiple performance studies, stressed the configuration with different types of workloads to determine the limitations of performance, and verified the sustainability of that performance.

We generally try to maintain a standard and consistent testing environment and methodology and take measures to limit caching effects. 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.

Our performance analysis focused on these key performance characteristics:

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

To characterize the performance of the solution, the following benchmarks were used:

- IOzone N to N sequential
- IOR N to 1 sequential
- IOzone random
- MDtest

For all the benchmarks listed above the number of compute nodes used for testing was 16. When a higher number of threads was required, those threads were equally distributed on the compute nodes (i.e., 64 threads = 4 threads per node, 128 threads = 8 threads per node, 256 threads = 16 threads per node, 512 threads = 32 threads per node, and 1024 threads = 64 threads per node). The intention was to simulate a higher number of concurrent clients with the limited number of compute nodes.

Client configuration

Table 6 shows the configuration of the compute clients used for performance evaluation of the solution.

Table 6. Client configuration

Component	Details
Clients	16x Dell EMC PowerEdge C6420 Compute Nodes
BIOS	2.9.3
Processor	2x Intel Xeon Gold 6248 CPU @ 2.50GHz, 20 cores each
Memory	12x 16GB DDR4 2933 MT/s DIMMs - 192GB
BOSS Card	2x 120GB M.2 boot drives in RAID 1 for OS
Operating System	Red Hat Enterprise Linux Server release 8.3
Kernel Version	4.18.0-240.el8.x86_64
Interconnect	1x NVIDIA ConnectX-6 HDR100 HCA

Client configuration

The following tuning options are in place on the metadata and storage servers:

```
# cat /etc/sysctl.d/90-beegfs.conf

vm.dirty_background_ratio = 5
vm.dirty_ratio = 20
vm.min_free_kbytes = 262144
vm.vfs_cache_pressure = 50
vm.zone_reclaim_mode = 0
kernel.numa_balancing = 0
```

In addition to the above, the following BeeGFS tuning options were used:

- `tuneTargetChooser` parameter was set to "`roundrobin`" in the metadata configuration file
- `tuneNumWorkers` parameter was set to 128 for metadata and for storage
- `connMaxInternodeNum` parameter was set to 64 for metadata, storage and for clients

Findings

Sequential writes and reads IOzone N-N

To evaluate sequential reads and writes, the IOzone benchmark v3.492 was used in the sequential read and write mode. These tests were conducted on multiple thread counts starting at 1 thread and increasing in powers of 2, up to 1024 threads. At each thread count, an equal number of files were generated since this test works on one file per thread or the N clients to N file (N-N) case. The processes were distributed across 16 physical client nodes in a round robin or cyclical fashion so that the requests are equally distributed and there is load balancing. An aggregate file size of 8TB was selected which was equally divided among the number of threads within any given test. The aggregate file size was chosen large enough to minimize the effects of caching from the servers as well as from BeeGFS clients. IOzone was run in a combined mode of write then read (-i 0, -i 1) to allow it to coordinate the boundaries between the operations.

For this testing and results, we used a 1MiB record size for every run. The commands used for Sequential N-N tests are as follows:

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

The default stripe count for BeeGFS is 4. However, the chunk size and the number of targets per file can be configured on a per-directory basis. For all these tests, BeeGFS stripe size was chosen to be 2MB and stripe count was chosen to be 2 since we have two targets per NUMA zone as shown below:

```
$ beegfs-ctl --getentryinfo /mnt/15g-perf/Benchmarks/ --
cfgFile=/etc/beegfs/beegfs-client-15Gperf.conf

Entry type: directory
EntryID: 1-61147BE4-1
Metadata node: storage1-numa0-3 [ID: 3]
Stripe pattern details:
+ Type: RAID0
+ Chunksize: 2M
+ Number of storage targets: desired: 2
+ Storage Pool: 1 (Default)
```

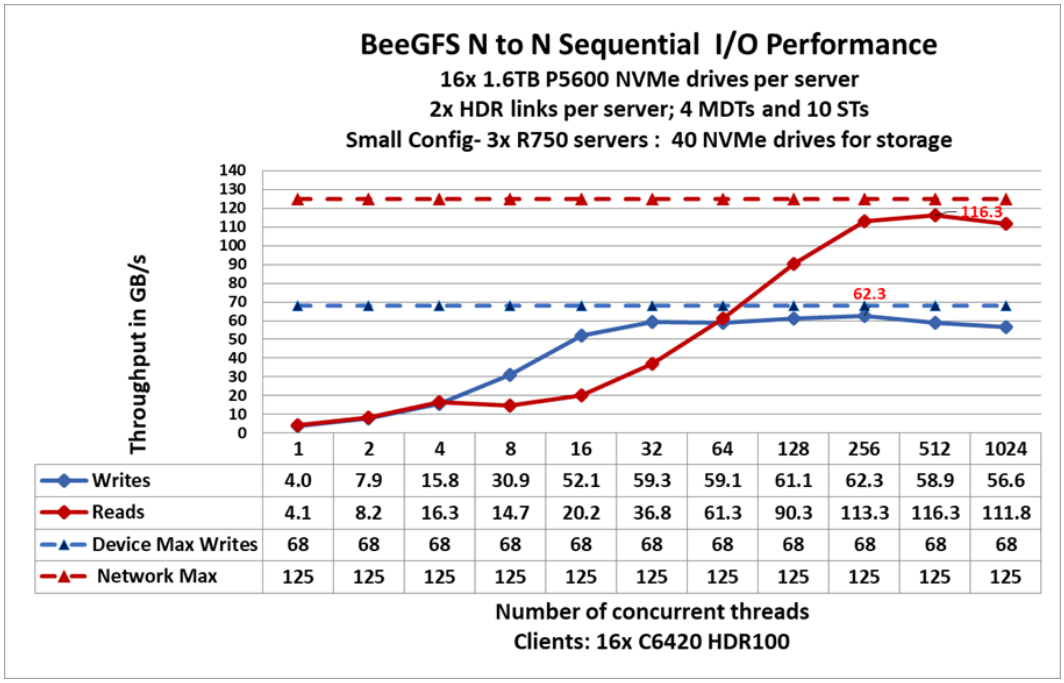


Figure 9. Sequential IOzone Performance with aggregate file size 8 TB

The IOzone sequential write and read tests were performed three times, and the mean value is plotted in Figure 9 above, with the peak performance highlighted in the chart. We observe that peak read performance is 116 GB/s at 512 threads and peak write is 62 GB/s at 256 threads. As per the technical specifications of the Intel P5600 1.6 TB NVMe SSDs, each drive can provide 3.5 GB/s peak read performance and 1.7 GB/s peak write performance, which allows a theoretical peak of 140 GB/s for reads and 68 GB/s for writes. However, here the network is the limiting factor. We have a total of 5 InfiniBand HDR links for the storage servers in the test bed.

Each link can provide a theoretical peak performance of 25 GB/s which provides an aggregate theoretical peak performance of 125 GB/s. The achieved peak read and write performance are 93% and 92% respectively of the theoretical peak performance.

The single thread performance is observed to be ~4 GB/s for writes and reads. We observe that the write performance increases monotonically up to 16 threads, reaches a plateau, peaks at 256 threads and then starts decreasing. At lower thread counts read and write performance are the same. The read performance registers an almost steady linear increase as we scale up the number of concurrent threads, and we observe the peak performance at 512 threads.

We also observe that the read performance is lower than writes for thread counts from 8 to 128. This is because while a PCIe read operation is a Non-Posted Operation, requiring both a request and a completion, a PCIe write operation is a fire and forget operation. Once the Transaction Layer Packet is handed over to the Data Link Layer, the operation completes. A write operation is a "Posted" operation that consists of a request only. Read throughput is typically lower than the write throughput because reads require two transactions instead of a single write for the same amount of data. The PCI Express uses a split transaction model for reads. The read transaction includes the following steps:

- The requester sends a Memory Read Request (MRR).
- The completer sends out the acknowledgement to MRR.
- The completer returns a Completion with Data.

The read throughput depends on the delay between the time the read request is issued and the time the completer takes to return the data. However, when the application issues enough number of read requests to cover this delay, then throughput is maximized. That is the reason why while the read performance is less than that of the writes from 16 threads to 128 threads, we measure an increased throughput when the number of requests increases. A lower throughput is measured when the requester waits for completion before issuing subsequent requests. A higher throughput is registered when multiple requests are issued to amortize the delay after the first data returns.

More detail about the PCI Express Direct Memory Access is available at https://www.intel.com/content/www/us/en/programmable/documentation/nik1412547570040.html6

Sequential writes and reads IOR N-1

The performance of sequential reads and writes with N threads to a single shared file was measured with IOR version 3.3.0, assisted by OpenMPI 4.1.0rc5 to run the benchmark over the 16 compute nodes. Tests executed varied from single thread up to 256 threads.

The following command was used to execute the benchmark for writes and reads, where threads was the variable. The number of threads used was incremented in powers of two. The transfer size is 2M and each thread wrote to or read 128G from a single file striped over all the 10 storage targets. Three iterations of each test have been run and the mean value has been recorded. Figure 10 shows the N to 1 sequential I/O performance. The command used to run the test is given below:

```
mpirun -machinefile $hostlist --map-by node -np $threads ~/bin/ior
-C -w -r -i 3 -d 3 -e -o $working_dir/ior.out -s 1 -g -t 2M -b
$file_size
```

The -C option (reorder/Tasks) forces each MPI process to read data written by its neighboring node thereby ensuring the reliability of the read performance.

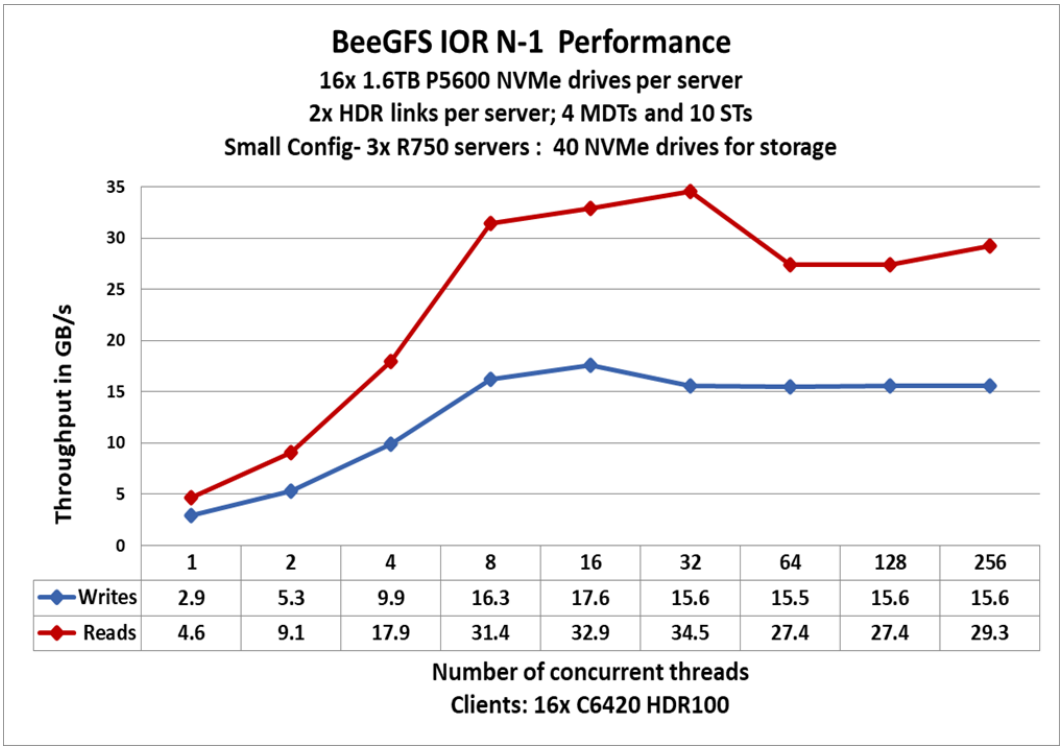


Figure 10. IOR N-1 (single shared file) performance

From the results in Figure 10, we observe that there is a steady increase in both the read and write performance as the number of concurrent threads is increased. The peak read performance of 34.5 GB/s is observed at 32 threads. Write performance attains the peak value of 17.6 GB/s at 16 threads.

Random writes and reads N-N

To evaluate random IO performance, IOzone was used in the random mode. Tests were conducted on thread counts starting from a single thread up to 1024 threads. Direct IO option (-I) was used to run IOzone so that all operations bypass the buffer cache and go directly to the devices. BeeGFS stripe count of 2 and chunk size of 2MB was used. A 4KiB request size is used on IOzone. Performance is measured in I/O operations per second (IOPS). The command used for executing the random writes and reads is as follows:

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

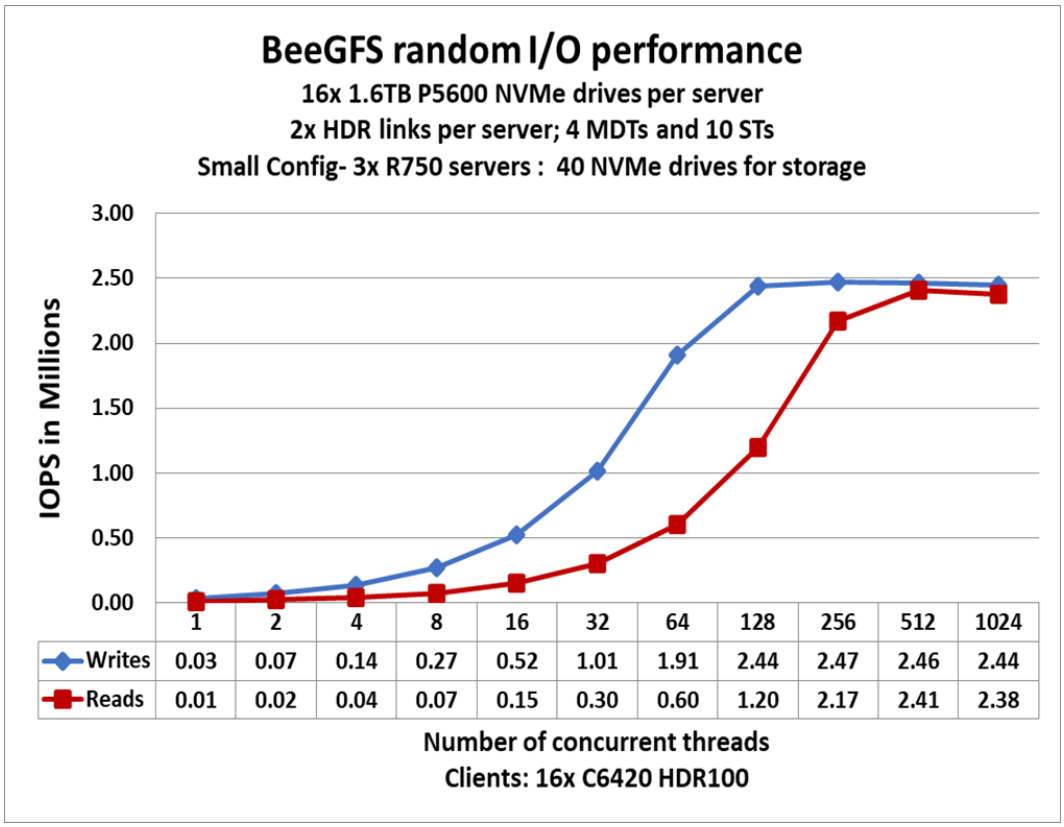


Figure 11. Random I/O performance

Two iterations of the random write and read tests were done and the mean value of the results is shown in Figure 11. The random writes peak at ~2.5 Million IOPS at 512 threads and the random reads peak at ~2.45 Million IOPS at 512 threads. Both the write and read performance show a higher performance when there are a higher number of IO requests. This is because NVMe standard supports up to 64K I/O queue and up to 64K commands per queue. This large pool of NVMe queues provide higher levels of I/O parallelism and hence we observe IOPS exceeding 2 million.

Metadata performance

Metadata performance was measured with MDtest version 3.3.0, assisted by OpenMPI 4.1.0rc5 to run the benchmark over the 16 compute nodes. Tests executed varied from single thread up to 512 threads. The benchmark was used for files only (no directories metadata), getting the number of creates, stats, reads and removes the solution can handle.

The following command was used to execute the benchmark, where threads was the variable with the number of threads used (1 to 512 incremented in powers of two), and $hostfile is the corresponding file that allocated each thread on a different node, using round robin to spread them homogeneously across the 16 compute nodes.

```
mpirun -machinefile $hostlist --map-by node -np $threads
~/bin/mdtest -i 3 -b $Directories -z 1 -L -I 1024 -y -u -t -F
```

Since performance results can be affected by the total number of IOPs, the number of files per directory and the number of threads, consistent number of files across tests was chosen to be able to compare them on similar grounds. The total number of files is ~ 2M

in powers of two (2^21 = 2097152). The number of files per directory was fixed at 1024, and the number of directories varied as the number of threads changed as shown in Table 7. Figure 12 shows the metadata performance using empty (i.e., 0 KB) files.

Table 7. Distribution of files on directories for MDtest

Number of threads	Number of files per directory	Number of directories per thread	Total number of files
1	1024	2048	2,097,152
2	1024	1024	2,097,152
4	1024	512	2,097,152
8	1024	256	2,097,152
16	1024	128	2,097,152
32	1024	64	2,097,152
64	1024	32	2,097,152
128	1024	16	2,097,152
256	1024	8	2,097,152
512	1024	4	2,097,152

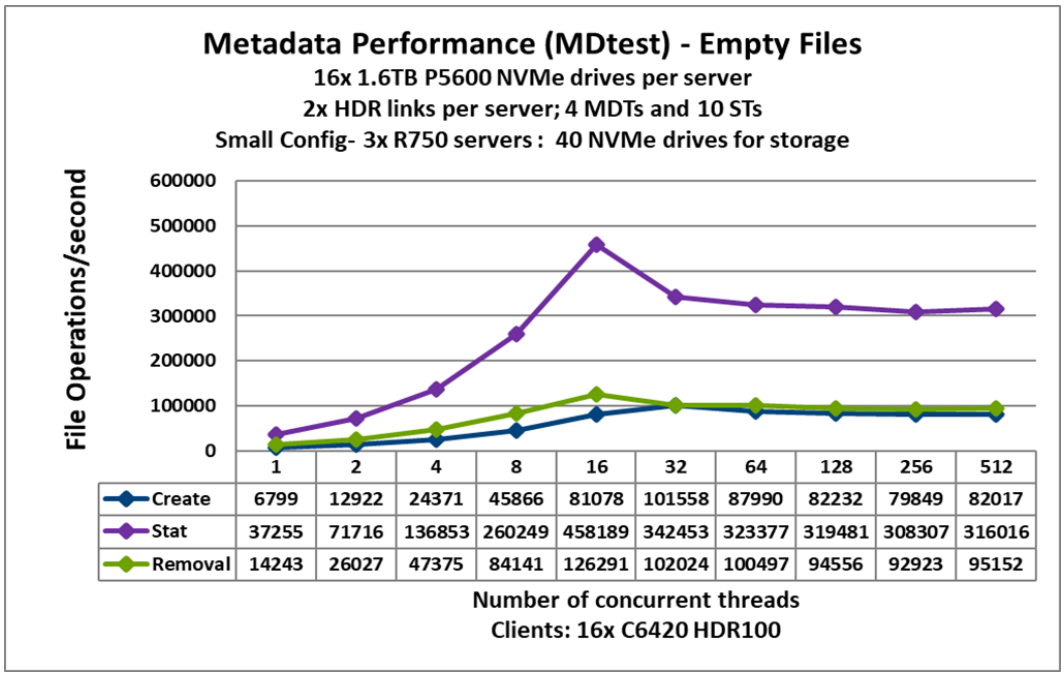


Figure 12. Metadata performance

The system gets very good results with Stat and Removal operations reaching their peak value at 16 threads with ~458K op/s and ~126K op/s respectively. Create operations attained the maximum value at 32 threads with ~101K op/s.

Once they reach their peak value, for stat operations the performance does not drop below 300K op/s. The create and removal operations remain stable after reaching a plateau.

Sizing guidelines

Overview

The Validated Design for BeeGFS high-performance storage solution has been designed to be flexible and one can easily and seamlessly scale performance and/or capacity by adding additional servers. In addition to the base configurations shown in Figure 8, the following additional configurations shown in Figure 13 are also available:

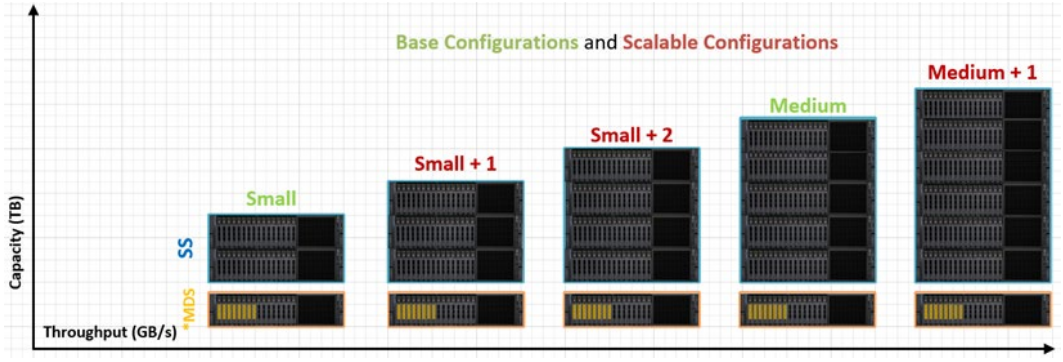


Figure 13. Scalable configurations of BeeGFS High-Performance Solution

The metadata portion of the stack remains the same for all the above configurations described in this White Paper. This is because the storage capacity requirements for BeeGFS metadata are typically 0.5% to 1% of the total storage capacity. However, it really depends on the number of directories and files in the file system. As a general guideline, the user can add an additional metadata server when the percentage of metadata capacity to the storage falls below 1%. Table 2 shows the usable capacity for the different flexible configurations of the BeeGFS storage solution.

Table 8. Usable capacity of the scaled configurations

Configuration		Small	Small+1	Small+2	Medium	Medium+1
# of dedicated storage servers		2	3	4	5	6
# of NVMe drives for data storage		40	56	72	88	104
Estimated Usable Space (1.6 TB/3.2 TB /6.4 TB P5600 NVMe SSDs)	1.6 TB	58 TiB	81 TiB	104 TiB	127 TiB	150 TiB
	3.2 TB	115 TiB	161 TiB	207 TiB	254 TiB	300 TiB
	6.4 TB	231 TiB	323 TiB	415 TiB	507 TiB	599 TiB

Estimated usable space is calculated in TiB (since most tools show usable space in binary units) using the following formula:

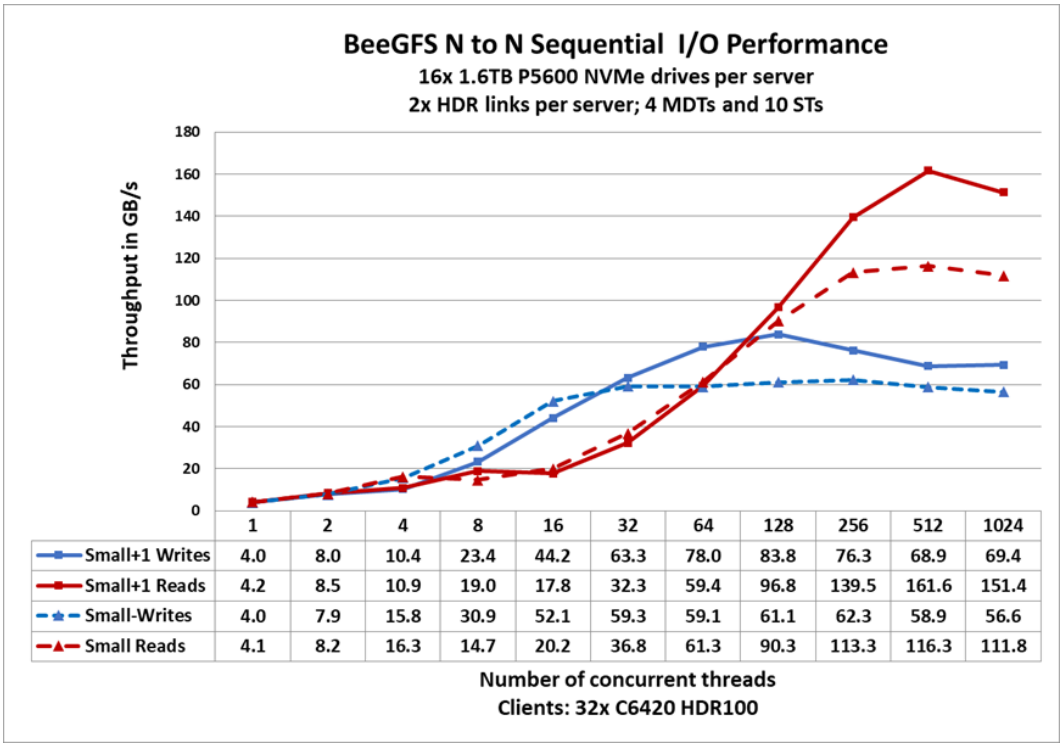
```
BeeGFS Usable Space in TiB = (0.99* # of Drives* size in TB *
(10^12/2^40)
```

In this formula, 0.99 is the factor arrived at by assuming conservatively that there is a 1% overhead from the file system. For arriving at the number of drives for storage, eight drives from the MDS are also included. This is because, in the MDS, the eight drives in NUMA zone 0 are used for metadata and the 8 drives in the NUMA zone 1 are used for storage. The last factor in the formula 10^12/2^40 is to convert the usable space from TB to TiB.

Performance of scalable configurations

Sequential writes and reads IOzone N-N

The performance characterization of the small configuration was carried out using the IOzone, IOR and MDtest benchmarks as described above. An additional dedicated storage server was added to the existing small configuration to know how the solution scales and the IOzone N-N sequential benchmark was run exactly as described in the corresponding section above. However, while only 16 clients were used for performance evaluation of small configuration for a small+1 configuration, 32 clients were used.



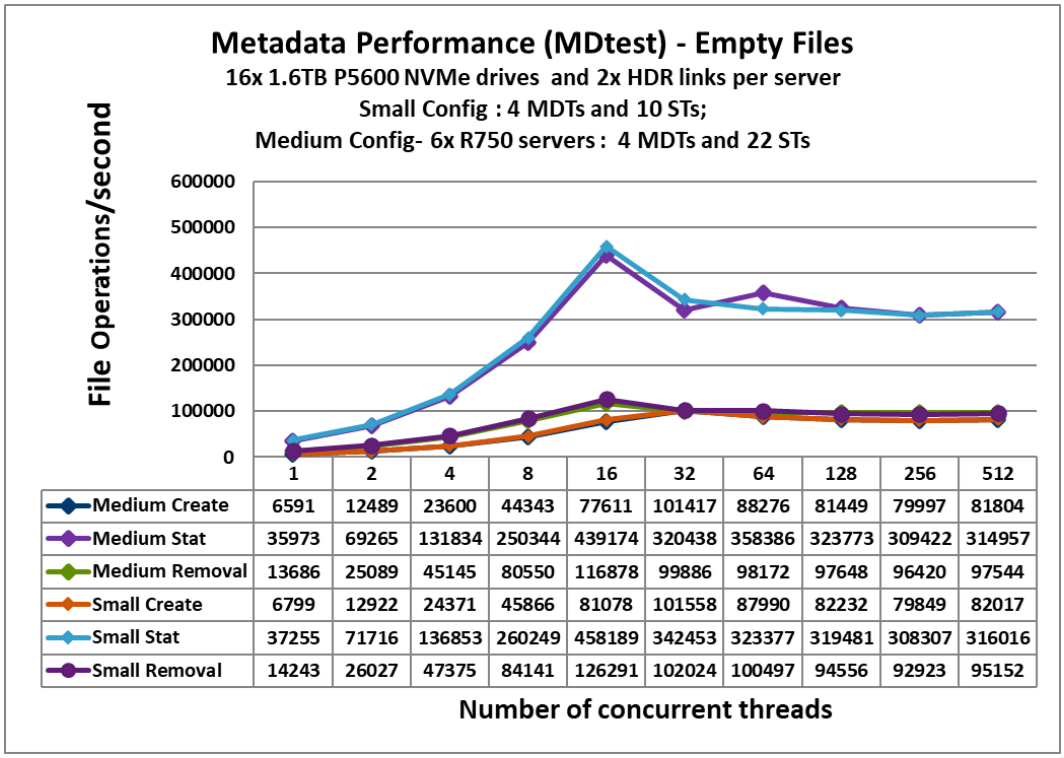
	1	2	4	8	16	32	64	128	256	512	1024
Small+1 Writes	4.0	8.0	10.4	23.4	44.2	63.3	78.0	83.8	76.3	68.9	69.4
Small+1 Reads	4.2	8.5	10.9	19.0	17.8	32.3	59.4	96.8	139.5	161.6	151.4
Small-Writes	4.0	7.9	15.8	30.9	52.1	59.3	59.1	61.1	62.3	58.9	56.6
Small Reads	4.1	8.2	16.3	14.7	20.2	36.8	61.3	90.3	113.3	116.3	111.8

Figure 14. Sequential IOzone performance of Small and Small+1 configurations

Figure 14 shows a comparison of the sequential I/O performance for a Small and a Small+1 configuration. The theoretical network performance for is 125 GB/s and 175 GB/s respectively for the Small and Small+1 configuration since the former has 5x HDR links and the latter has 7x HDR links for data. The achieved peak read performance is 92% and 93% respectively of the theoretical maximum network performance for the respective configuration. The write performance is limited by the technical specifications of the P5600 NVMe drives, each of which can provide 1.7 GB/s in writes which is 68 GB/s for a small configuration and 95.2 GB/s for a Small +1 configuration. For the mixed-use devices used in the test bed, the achieved write performance is 92% and 88% of the theoretical maximum values for the Small and Small+1 configuration respectively.

Metadata performance

Since the metadata component is the same across the scalable configurations, it is expected that, irrespective of the number of dedicated servers used in the configuration, there will not be any change in the metadata performance. Figure 15 below shows a comparison of the metadata performance between the Small and Medium configurations which use three and six dedicated storage servers respectively.



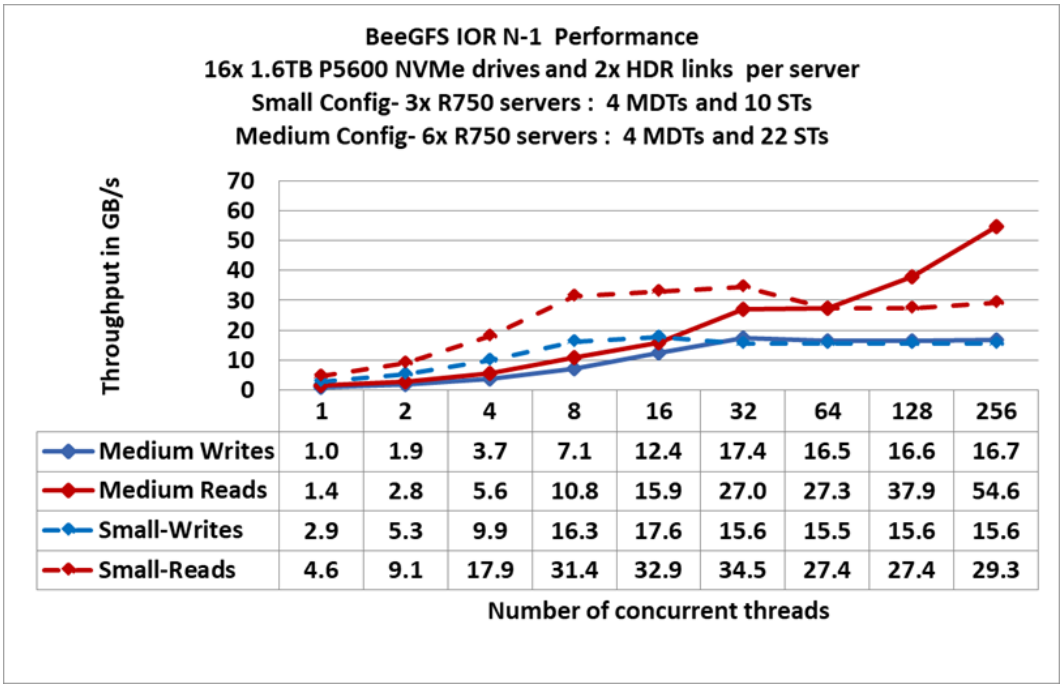
	1	2	4	8	16	32	64	128	256	512
Medium Create	6591	12489	23600	44343	77611	101417	88276	81449	79997	81804
Medium Stat	35973	69265	131834	250344	439174	320438	358386	323773	309422	314957
Medium Removal	13686	25089	45145	80550	116878	99886	98172	97648	96420	97544
Small Create	6799	12922	24371	45866	81078	101558	87990	82232	79849	82017
Small Stat	37255	71716	136853	260249	458189	342453	323377	319481	308307	316016
Small Removal	14243	26027	47375	84141	126291	102024	100497	94556	92923	95152

Figure 15. Metadata performance of Small and Medium configurations

It can be inferred from the above chart that, except for the slight variation which might be due to the run-to-run variability of the benchmark run itself, for all the metadata operations, file create, file stat and file removal there is very little variation in the metadata performance for the small and medium configurations which is the expected performance.

IOR performance

Figure 16 shows the IOR N-1 Sequential I/O Performance of the Small and Medium configurations that use three and six PowerEdge R750 servers respectively. While the write performance is similar for the small and medium configurations, the read performance shows significant increase for the medium configuration with the increase in the number of IO requests. This is again because NVMe standard supports up to 64K I/O queue and up to 64K commands per queue thereby providing higher levels of I/O parallelism. Another reason for the higher throughput at higher thread count is due to the split transaction model adopted by PCIe express for handling reads as explained in the section **Sequential writes and reads IOzone N-N**.



	1	2	4	8	16	32	64	128	256
Medium Writes	1.0	1.9	3.7	7.1	12.4	17.4	16.5	16.6	16.7
Medium Reads	1.4	2.8	5.6	10.8	15.9	27.0	27.3	37.9	54.6
Small-Writes	2.9	5.3	9.9	16.3	17.6	15.6	15.5	15.6	15.6
Small-Reads	4.6	9.1	17.9	31.4	32.9	34.5	27.4	27.4	29.3

Figure 16. IOR N-1 (single shared file) performance

Extrapolation of performance for 24-drive configuration

Considerations

The R750 configuration supports twenty-four NVMe drives. The NVMe drives are connected through PCIe switches, which allows the system to overprovision PCIe lanes to more NVMe drives while persevering I/O slots, therefore enabling low latency CPU access to twelve devices per CPU. Performance can easily be scaled for various dense workloads, such as big data analytics. This configuration appeals to customers wanting to consolidate Storage media to NVMe (from SAS/SATA). Customers requiring large NVMe capacity will benefit from the low latency of NVMe with up to twenty-four NVMe drives available for population.

The primary difference between the 16-drive configuration and the 24-drive configuration is that in case of the 16-drive configuration, there is a direct cable connection from NVMe ports on the system board of the system whereas in case of 24-drive configuration the connection is through NVMe switch adapter with 32x PCIe lanes.

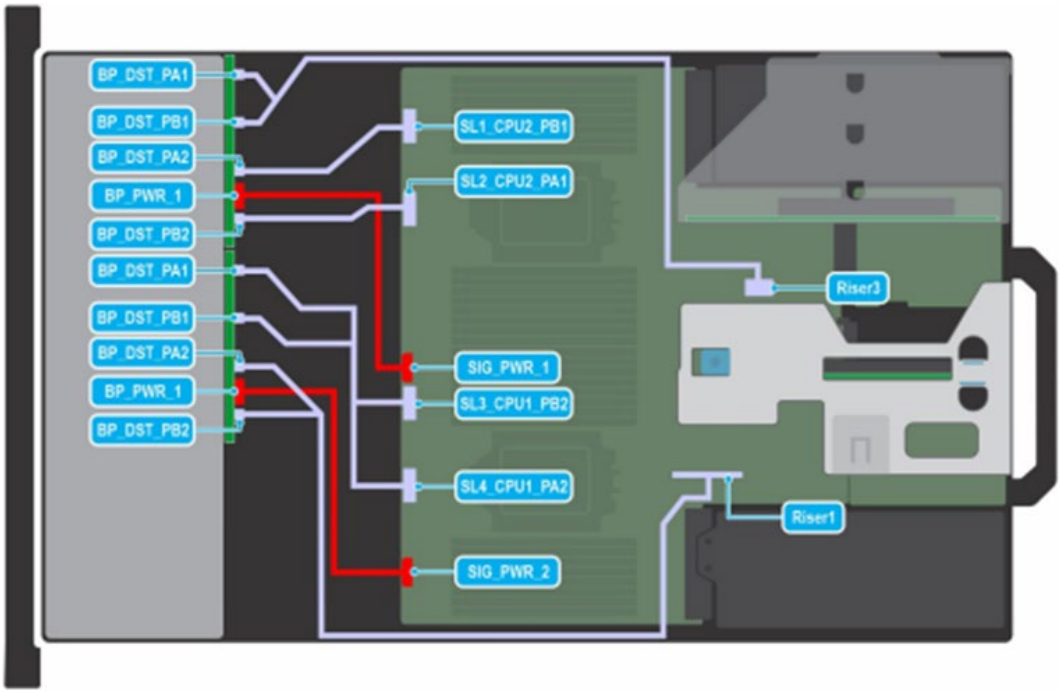


Figure 17. 16x 2.5-inch NVMe direct

Figure 17 shows the connections from the system board power connector to the backplane power connector and the connections from the *signal connector on system board to the backplane signal connector.*

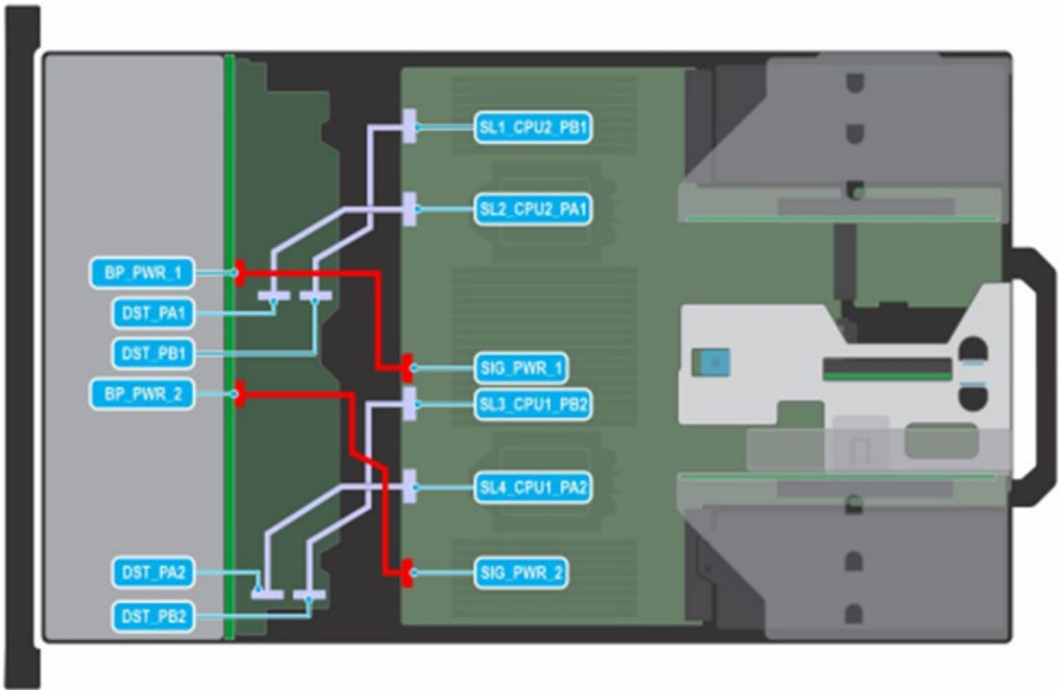


Figure 18. 24x 2.5-inch NVMe switch

Figure 18 shows the connections from the system board power connector to the backplane power connector and the connections from the *signal connector on system board to the backplane switch signal connector.*

Except for the use of a PowerEdge R750 server that has an NVMe backplane which supports 24x NVMe drives, the architecture and configuration of the Validated Design for BeeGFS high-performance storage is essentially the same. Figure 19 shows the design of the 24-drive configuration for easy reference:

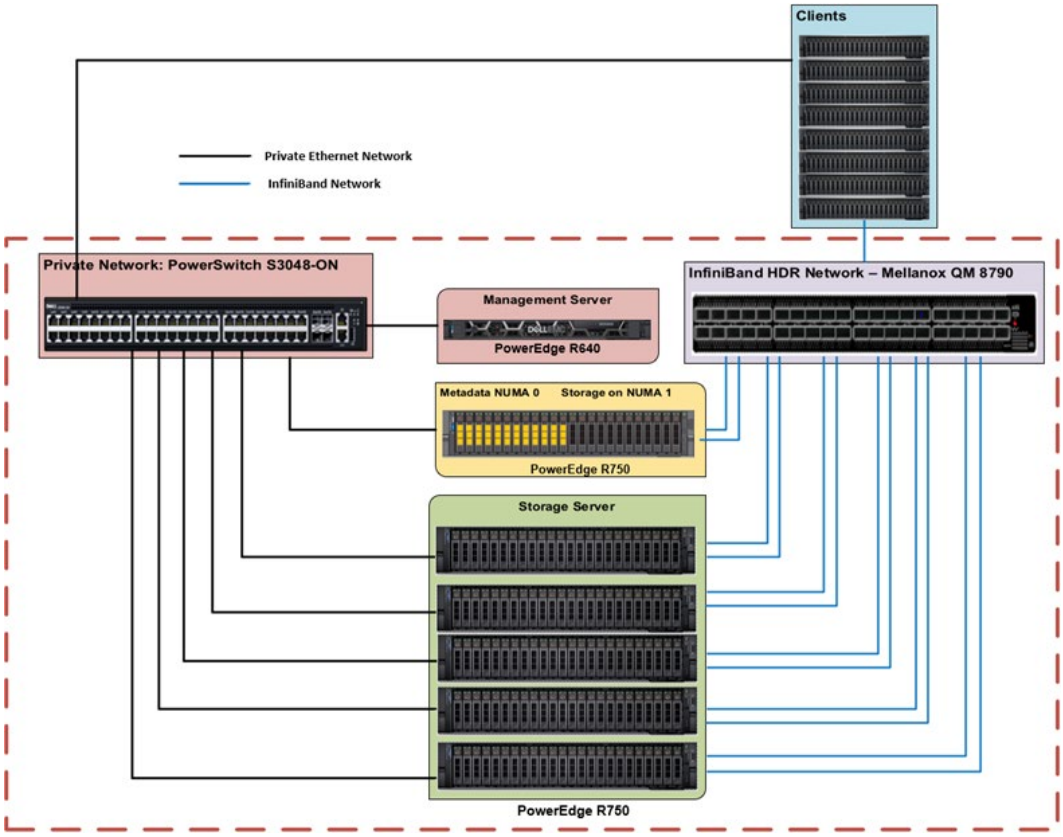


Figure 19. Design for BeeGFS high-performance storage PowerEdge servers equipped with 24x 2.5-inch NVMe drives (Medium configuration)

The NVMe CPU Mapping in the PowerEdge R750 server is given in Figure 20 below which shows that each CPU handles twelve NVMe devices. Such a balanced configuration provides maximum performance by ensuring that the processors are equally occupied in handling the I/O requests to and from the NVMe devices. There is technically a 3:1 oversubscription of the NVMe PCIe lanes from the processors. This is because we bring x32 lanes to the PCIe switch which fans out to feed 24x x4 devices (96 lanes).

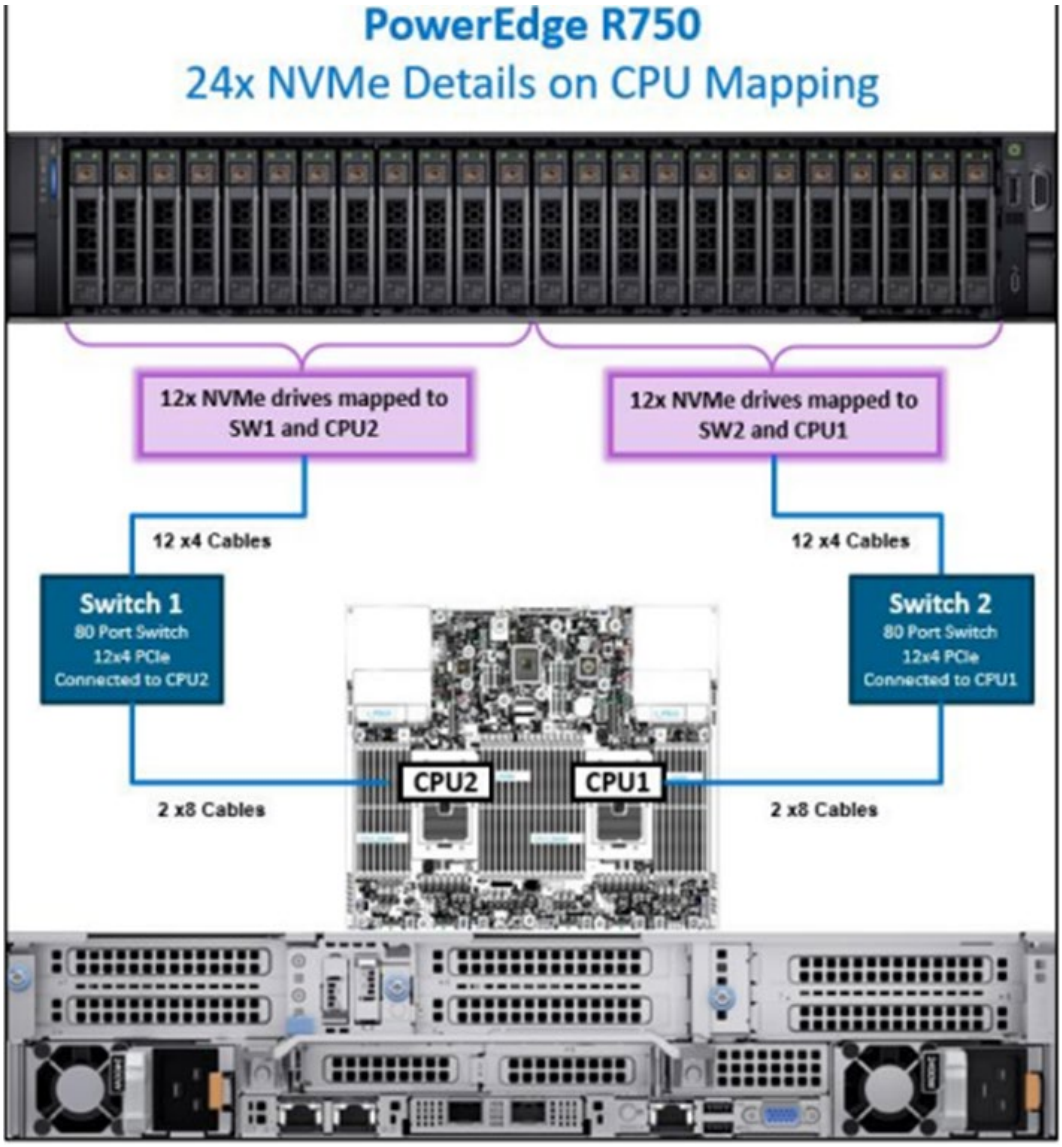


Figure 20. NVMe CPU mapping on PowerEdge R750 equipped with 24x NVMe drives

Impact of PCIe connection on performance

Although the actual performance of the 24-drive configuration has not been measured and validated at the HPC & AI Innovation Lab, it is reasonable to assume that the type of PCIe connection does not limit the performance. This is because although there is some processing overhead for the switch, *for both the direct cable and the x16 adapter configurations, each NVMe SSD in the system has the bandwidth of four PCIe lanes only.* As a result, the impact of the PCIe connection on sequential as well as random workloads is expected to be insignificant. This assumption is also based on the results of the performance studies carried out using the NetBench tool which is a built-in benchmarking tool of BeeGFS.

NetBench for evaluating network throughput

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 process described above, the netbench tool was used to benchmark the network infrastructure. 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=10.140.63.10
```

Given below is the partial output of the netbench benchmark results for the Small configuration with 3x PowerEdge R750 servers.

```
$ cat iozone_results_write_1_512.081021130600
0 1m 8g 512 ~/2021Q2_BeePerf/15G-beeperf-08042021/15g-
perf.backup/hosts.309571
              Iozone: Performance Test of File I/O
                      Version $Revision: 3.492 $
                      Compiled for 64 bit mode.
                      Build: linux

              Contributors:William Norcott …..

              Run began: Tue Aug 10 13:06:00 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 8388608 kB
              No retest option selected
              Network distribution mode enabled.
Command line used: ~/2021Q2_BeePerf/15G-beeperf-08042021/15g-
perf.backup/iozone3_492/src/current/iozone -i 0 -c -e -w -r 1m -I
-s 8g -t 512 -+n -+m ~/2021Q2_BeePerf/15G-beeperf-08042021/15g-
perf.backup/hosts.309571
              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 512 processes
              Each process writes a 8388608 kByte file in 1024
kByte records

              Test running:
Children see throughput for 512 initial writers        =
123460328.84 kB/sec
```

```
              Min throughput per process                        =
167061.00 kB/sec
              Max throughput per process                        =
304901.41 kB/sec
              Avg throughput per process                        =
241133.45 kB/sec
              Min xfer                                          =
7742464.00 kB
```

Note: While running IOzone with netbench enabled, since the actual read and write operations are not taking place from the underlying file system, neither the IOzone binary not the hostlist should be saved on BeeGFS mount point. Instead, it must be saved on a shared location accessible to all clients such as the home directory. Finally, the benchmark must be run with the mountpoint of the BeeGFS file system as the working directory.

Given below is the output of the netbench benchmark results for the Small+1 configuration with 4x PowerEdge R750 servers.

```
$ ./iozone_testing_sequential_netbench
      /mnt/15g-perf/Benchmarks ~/netbenchRHEL
      0 1m 8g 512 /home/nirmala/netbenchRHEL/hosts.157606
              Iozone: Performance Test of File I/O
                      Version $Revision: 3.492 $
                      Compiled for 64 bit mode.
                      Build: linux

              Contributors:William Norcott      …

              Run began: Tue Sep 14 11:14:43 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 8388608 kB
              No retest option selected
              Network distribution mode enabled.
Command line used: ~/netbenchRHEL/current/iozone -i 0 -c -e -w -r
1m    -I -s 8g -t 512 -+n -+m ~/netbenchRHEL/hosts.157606
              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 512 processes
              Each process writes a 8388608 kByte file in 1024
kByte records
                                                                Test
running:
```

```
                Children see throughput for 512 initial writers  =
163330853.41 kB/sec
                Min throughput per process                       =
308184.50 kB/sec
                Max throughput per process                       =
333010.00 kB/sec
                Avg throughput per process                       =
319005.57 kB/sec
                Min xfer                                         =
7764992.00 kB
```

Performance of 24-drive configuration

Table 9 below shows the extrapolation of the performance if the 24-drive configuration of the PowerEdge R750 server is used instead of the 16-drivce CPU direct attach version of the PowerEdge R750 server. In the table below the theoretical maximum write performance is arrived at based on the technical specs of P5600 NVMe drives. (Each drive can provide 1.7 GB/s write performance). The read performance is limited by the theoretical maximum network performance.

Table 9. Extrapolation of performance from 16-drive to 24-drive configuration

Configuration	Theoretical maximum		NetBench	Peak performance: 16x drive configuration		Peak performance: 24x drive configuration	
	Write	Network		Write	Read	Write	Read
Small with 40x NVMe drives and 5x HDR links for storage	68 GB/s	123 GB/s	123 GB/s	62.3 GB/s	116.3 GB/s	~ 89 GB/s but <102 GB/s*	>=116.3 but <125 GB/s
Small+1 with 56x NVMe drives and 7x HDR links for storage	95 GB/s	175 GB/s	163 GB/s	83.8 GB/s	161.6 GB/s	~ 125 GB/s but <142 GB/s*	>=161.6 but <175 GB/s

Conclusion

Summary

This White Paper validates the performance of the Dell Technologies Validated Design for BeeGFS high-performance storage solution in Small configuration that uses 3x PowerEdge R750 servers, each of which is equipped with 16x 1.6 TB P5600 NVMe SSDs and dual InfiniBand HDR ConnectX-6 HCAs. This paper describes the design, configuration and tuning parameters used in the solution. It provides the test parameters used in the different storage benchmarks used for performance characterization.

The solution uses five InfiniBand HDR links for storage, for the Small configuration. Each HDR link can provide a theoretical peak performance of 25GB/s. So, the theoretical peak network performance is 125 GB/s. With the high performance NVMe SSDs used as internal storage, the achieved peak sequential peak performance is 116 GB/s with a small footprint of three servers, which is 93% of the theoretical peak network performance.

As per the technical specs of the P5600 NVMe SSDs, each NVMe SSD is capable of providing 1.7 GB/s in writes. With 40x NVMe drives for data in the configuration, the theoretical maximum write performance of the solution works out to 68GB/s. The achieved peak sequential write performance is 62 GB/s which is 92% of the theoretical peak performance.

Table 10 summarizes the performance of the solution in Small configuration that uses PowerEdge R750 servers equipped with 16x NVMe drives and 2x HDR links per server.

Table 10. Performance of Small configuration

Benchmark test	Peak performance	
	Write	Read
Large Sequential N clients to N files	62.3 GB/s	116.3 GB/s
Large Sequential N clients to single shared file	17.6 GB/s	34.5 GB/s
Random small blocks N clients to N files	2.47M IOps	2.41M IOps
Metadata create empty files	101K IOps	
Metadata stat empty files	458K IOps	
Metadata remove empty files	126K IOps	

We value your feedback

Dell Technologies and the authors of this document welcome your feedback on the solution and the solution documentation. Contact the Dell Technologies Solutions team by email or provide your comments by completing our documentation survey.

Author: Nirmala Sundararajan

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

Terminology

BeeGFS is an open source, software defined, robust parallel filesystem. The primary advantage of BeeGFS is its ease of use. BeeGFS is very flexible and works with any underlying filesystem such ext4, xfs, zfs, etc. BeeGFS can be reexported using NFS and SMB protocols.

The server components of BeeGFS are user space daemons. The client is a native kernel module. But does not need any patches to the kernel itself. All BeeGFS components can be installed and updated without even rebooting the machine. The following table provides definitions for some of the terms that are used in this document.

Table 11. Terminology

Term	Definition
Service	This refers to a single daemon, like metadata, storage, or mgmtd service. Multiple services of one or different types can run on a single server
Client (service)	The BeeGFS client is a kernel module. Sometimes also the term client service is used, but characteristics that are valid for other services usually are not valid for the client service.
Mgmtd	The Management daemon is the central point for registration of all server (meta, storage) and client services. A BeeGFS file system is defined by its mgmtd service. All services that register there are part of the same BeeGFS mount point.
Storage Service	The storage service manages the storage (user) data of the BeeGFS file system. One storage service can manage multiple storage targets.
Metadata Service	The metadata service stores the structure and meta information of the file system. One metadata service can manage only a single metadata target.
Target	A target can technically be any directory on a standard Linux file system that fulfills some basic characteristics. However, in most cases it refers to a partition on a block device, but in theory there could be multiple targets on a single block device.
Multi-mode	Refers to a configuration where several services of one kind run on a single server. Here it is used to have multiple metadata targets (as one service can control only one target) and to take advantage of multiple IB ports per server, by binding different services to different IB ports. The mgmtd service interprets each service to be on an individual node. (In this documentation the term “server” will refer to a physical machine and “node” to a specific beegfs service).
IDs of nodes and targets	In a BeeGFS environment, every service has a numeric ID and a string ID, which are named nodeID and nodeNumID. Moreover, storage targets have their specific numeric ID and string ID, which are targetID and targetNumID. The IDs can either be assigned at initialization or, if not initialized, the beegfs-mgmtd will assign free IDs to them upon first registration.

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 Performance Storage: HDR100 Refresh
- Dell EMC Ready Solutions for HPC BeeGFS High Performance Storage - Technical White Paper
- Scalability of Dell EMC Ready Solutions for HPC BeeGFS Storage
- Features of Dell EMC Ready Solutions for HPC BeeGFS Storage
- Dell EMC Ready Solutions for HPC BeeGFS High Performance Storage
- Dell.com/support: Focused on meeting customer needs with services and support for Dell EMC products.
- hpcatdell.com: Provides a consolidated list of all the publications from the HPC Engineering Team at the Dell Technologies HPC & AI Innovation Lab.
- Dell Technologies InfoHub provides a variety engineering publications.

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: Benchmarks and test tools

Benchmarks run

- The IOzone benchmark tool 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 IOR benchmark tool was used to measure sequential N to 1 read- and write throughput (GB/s).
- The MDtest benchmark was used for files only (no directories metadata), to get the number of creates, stats, reads and removes the solution can handle when using empty files.

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

IOzone

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 12. 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

IOzone argument	Description
-t	Number of threads
-e	Includes flush in the timing calculations
-r	Records size
-s	File size
-t	Number of threads
-+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

IOR

The command used to run the test is given below:

```
mpirun -machinefile $hostlist --map-by node -np $threads -mca btl
openib,self ~/bin/ior -w -r -i 3 -d 3 -e -o $working_dir/ior.out -
s 1 -g -t $record_size -b $file_size
```

The following table describes the `IOR` command-line arguments.

Table 13. IOR command-line arguments

mpirun argument	Description
-machinefile	Tells `mpirun` where the hostfile is located
-map-by node	Launch processes one per node, cycling by node in a round robin fashion. This spreads processes evenly among nodes and assigns MPI_COMM_WORLD ranks in a round-robin, “by-node” manner.
-np	Number of processes
IOR argument	**Description**
-C	reorderTasksConstant – changes task ordering to n+1 ordering for readback
-w	Write benchmark
-r	Read benchmark
-i	Number of repetitions
-d	Delay between repetitions in seconds
-e	perform fsync upon POSIX write close (make sure reads are only started after all writes are done)
-O	String of IOR directives
-o	path to file for the test
-s	Segment count

IOR argument	Description
-g	intraTestBarriers - use barriers between open, write/read, and close
-t	Transfer size
-b	Block size (amount of data for a process)

MDtest

`MDtest` is used with `mpirun`. For these tests, OpenMPI version < > was used. The command used to run the MDtest is given below:

```
mpirun -machinefile $hostlist --map-by node -np $threads
~/bin/mdtest -i 3 -b $Directories -z 1 -L -I 1024 -y -u -t -F
```

The following table describes the `MDtest` command-line arguments.

Table 14. MDtest command-line arguments

mpirun argument	Description
-machinefile	Tells mpirun where the hostfile is located
-map-by node	Launch processes one per node, cycling by node in a round robin fashion. This spreads processes evenly among nodes and assigns MPI_COMM_WORLD ranks in a round-robin, “by-node” manner.
-np	Number of processes
MDtest argument	**Description**
-d	The directory MDtest should run in
-i	The number of iterations the test will run
-b	Branching factor of directory structure
-z	Depth of the directory structure
-L	Files only at leaf level of tree
-I	Number of files per directory tree
-y	Sync the file after writing
-u	Unique working directory for each task
-C	Create files and directories
-R	Randomly stat files
-T	Only stat files and directories
-r	Remove files and directories left over from run