Western Digital.

REFERENCE ARCHITECTURE MAY 2022

High-Performance Lustre Filesystem Reference Architecture & Deployment Guide on Openflex™ Composable Infrastructure

Pavan Gururaj | Saravanakumar Pandian | Puspanjali Panda

Contents

1. Executive Summary

In high-performance computing (HPC), the efficient delivery of data to and from the compute nodes is critical and often complicated to execute. Researchers and end users can generate and consume data in HPC systems at such speed that the storage components become a major bottleneck. Getting maximum performance for their applications requires a scalable storage solution. OpenFlex™ Composable Infrastructure along with open-source Lustre solutions can deliver on these performance and storage capacity needs.

The data requirements around performance and capacity keep growing rapidly. Increasing the throughput and scalability of storage systems supporting the HPC system can require a great deal of planning and configuration. The Western Digital Lustre solution using OpenFlex Composable Infrastructure is designed for HPC and industry users who need to deploy a fully supported, easy-to-use, high-throughput, scale-out and cost-effective parallel file system solution. The solution greatly simplifies deploying, managing, and monitoring all the hardware and storage system components. It is easy to scale in capacity, performance, or both, thereby providing a convenient path to grow in the future.

The storage solution utilizes the OpenFlex F3200 fabric devices that leverage the Open Composable Infrastructure (OCI) approach in the form of disaggregated data storage using NVMe-over-Fabrics (NVMe-oF™) servers. The solution delivers a superior combination of performance, reliability, density, ease of use, and cost-effectiveness. The following paper will describe the Lustre parallel file system deployment on OpenFlex Composable Infrastructure.

2. Challenges

Network-centric computing environments demand reliable, high-performance storage systems which can properly authenticate clients and enable data storage and delivery. Simple cooperative computing environments such as enterprise networks typically satisfy these requirements using distributed file systems based on a standard client/server model. Distributed file systems such as NFS and SAN have been successful in a variety of enterprise scenarios but do not satisfy the requirements of today's high-performance computing environments because the NFS server can quickly become a major bottleneck as it does not scale well when used in large cluster environments. The NFS server also becomes a single point of failure, and the consequences of it crashing can become severe. SAN file systems are capable of very high performance, but are extremely expensive to scale up since they are implemented using Fibre Channel and therefore, each node that connects to the SAN must have a Fibre Channel card to connect to the Fibre Channel switch.

The Solution Is High-Performance Parallel Filesystem Solution with Lustre on OpenFlex Composable Infrastructure

Since Lustre is a global parallel file system with a global name space, it provides wide scalability of both performance and storage capacity and the ability to distribute very large files across many nodes. Because large files are shared across many nodes in the typical cluster environment, a parallel file system such as Lustre is ideal for high-end HPC cluster I/O systems. And Western Digital's OpenFlex F3200 is a fabric device that leverages the OCI approach in the form of disaggregated data storage using NVMe-oF. Composable Disaggregated Infrastructure (CDI) represents the modern architectural approach to data center infrastructure, disaggregating compute, storage, and network resources into shared pools that can be composed for on-demand allocation.

The Lustre distributed file system provides significant performance and scalability advantages over existing distributed file systems. Lustre leverages the power and flexibility of the open source Linux® operating system to provide a truly modern POSIX-compliant file system that satisfies the requirements of large clusters today, while providing a clear design and extension path for even larger environments tomorrow. Distributed file systems have well-known advantages. They decouple computational and storage resources, enabling client systems to focus on user and application requests while file servers focus on reading, delivering, and writing data. Centralizing storage on file servers facilitates centralized system administration, simplifying operational tasks such as backups, storage expansion, and general storage reconfiguration without requiring desktop downtime or other interruptions in service.

Lustre is a popular file system widely used on some of the fastest computers in the world. In this document we demonstrate a solution to growing demand of high-performance parallel file systems for HPC clusters, by deploying a high-performance distributed parallel file system on a composable infrastructure.

By combining a Lustre distributed file system with Western Digital's OpenFlex Composable Infrastructure, organizations can realize:

- Robust performance
- Scalability
- High availability
- Data durability
- Data integrity
- Data throughput

3. Technology Overview

3.1 OpenFlex™ E3000 Enclosure with F3200 Fabric Device Overview

OpenFlex is Western Digital's architecture that supports OCI through storage disaggregation. The OpenFlex E3000 is a 3U rack-mounted data storage enclosure and the OpenFlex F3200 is a fabric device that leverages this OCI approach in the form of disaggregated data storage using NVMe-oF. NVMe-oF is a networked storage protocol that allows storage to be disaggregated from compute to make that storage widely available to multiple applications and servers. For more details refer to Openflex-Composable-Infrastructure.

Enabling applications to share a common pool of storage capacity makes data easily shareable between applications, or needed capacity allocatable to an application regardless of location. Exploiting NVMe™ device-level performance, NVMe-oF promises to deliver the lowest end-to-end latency from application to shared storage. NVMe-oF enables composable infrastructures to deliver the data locality benefits of NVMe DAS (low latency, high performance) while providing the agility and flexibility of sharing storage and compute.

The maximum data storage capacity of F3000 is 614TB[1] when leveraging a full set of 10 F3200 fabric devices. F3200 is capable of scaling up to 2 million IOPs and cumulatively we can scale for each F3000 up to 20 million IOPs in a 3U solution.

Composable Infrastructure seeks to disaggregate compute, storage, and networking fabric resources into shared resource pools that can be available for on-demand allocation (i.e., "composable"). Composability occurs at the software level, while disaggregation occurs at the hardware level using NVMe-oF. NVMe-oF will vastly improve compute and storage utilization, performance, and agility in the data center.

[1] One terabyte(TB) is equal to one trillion bytes. Actual user capacity may be less due to operating environment.

Western Digital's vision for OCI is based on four key pillars:

Open

- Open in both API and form factor.
- Designed for robust interoperability of multi-vendor solutions.

Scalable

- Delivering the ability to compose solutions at the width of the network.
- Enable self-organizing systems of composable elements that communicate horizontally.

Disaggregated

- Pools of resources available for any use case that is defined at run time.
- Independent scaling of compute and storage elements to maximize efficiency and agility.

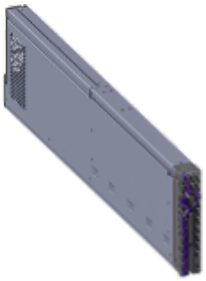
Extensible

- Inclusive of both disk and flash.
- Entire ecosystem of composable elements managed and orchestrated using a common API framework.
- Prepared for yet-to-come composable elements – e.g., memory, accelerators.

Open Composable API – Western Digital's new Open Composable API is designed for data center composability. It builds upon existing industry standards utilizing the best features of those standards as well as practices from proprietary management protocols.

OpenFlex is Western Digital's architecture that supports OCI through storage disaggregation – both disk and flash natively attached to a scalable fabric. OpenFlex does not rule out multiple fabrics, but whenever possible, ethernet will be used as a unifying connect for both flash and disk because of its broad applicability and availability.

OpenFlex F3200 Specification Summary



Specification	Value
Max raw data storage capacity per device	61.4TB
Data ingest capability	2 × 50Gb Ethernet
Data transfer rates	12GBps*
Number per enclosure	Up to 10
Hot swappable	Yes

3.2 RAIDIX ERA Overview

RAIDIX ERA allows for the creation of a highly performative RAID from NVMe and SAS/SATA SSD for the most demanding enterprise-grade tasks, ensuring fast and effective access to data. It is easy to maintain and more suitable for operating in large server infrastructures. For more details refer to RADIX.

RAIDIX ERA is a software RAID presented by the Linux kernel module and management utility (CLI).

- Adjusted for the most popular Linux distribution (Ubuntu, CentOS, and Oracle®).
- Works with local and remote drives.
- Provides RAID as a standard Linux block device.
- POSIX API support.

I/O handling parallelization and a lockless data path in RAIDIX ERA allow for the removal of array internal barriers and deliver unprecedented performance. RAIDIX ERA is also able to sustain high performance levels and low latency (< 0.5ms) even in mixed workloads. To protect data, RAIDIX ERA delivers a wide range of RAID level support: RAID 1/0/5/6/7.3/50/60/70. Moreover, performance loss during drive failures is reduced, which helps business applications run smoothly. That comes from an innovative approach to erasure coding calculations.

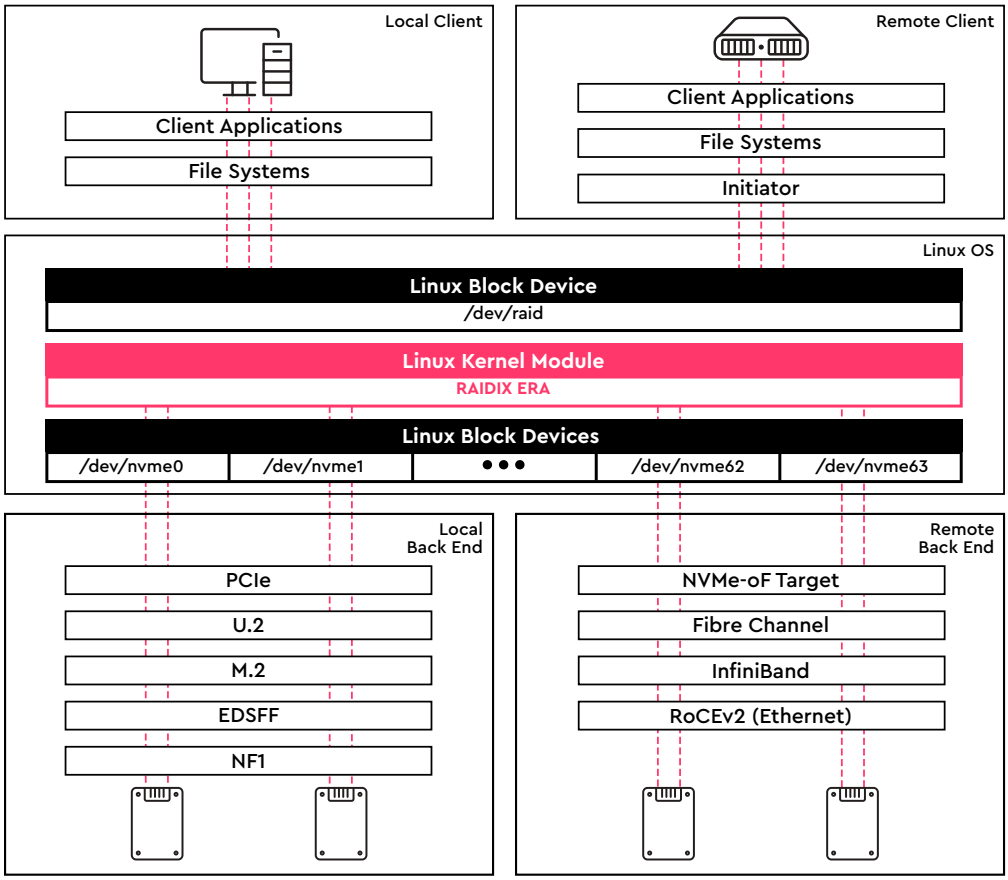


Figure 1 – RAIDIX ERA System Architecture

High Speed of the Checksums Calculations

The core innovation of the RAIDIX product is the unique software RAID that calculates array parity faster than any other alternatives in the storage industry. The RAID engine reads and writes parity blocks with record speed (about 25GBps for 1 CPU core) and therefore keeps high array performance even when the drive goes down.

During the sequential workloads, a drive failure reduces total storage performance by less than 10%. This result is better than any other existing storage solution.

Reduced RAID Rebuild Time

Rebuild (or reconstruction) of the RAID after a drive failure is a potentially fraught and dangerous time frame for storage administrators.

First of all, reconstructing data to a new drive usually consumes a significant part of the total array performance. Second, it increases the risk of data loss because the number of drives acceptable for the failure is down by one. With fast checksums calculation, RAIDIX software arrays require significantly less time to perform rebuild operations compared to existing solutions on the global storage market.

Advantages of the RAIDIX Software RAID

Due to its fast coding and decoding ability, RAID provides a stable performance level needed for smooth and uninterrupted business operations. Fast RAID rebuild protects storage from extensive system downtime and mitigates the impact on workflows even if a few drives fail. That is crucial for data-intensive systems and high-density storage infrastructures where even a single drive failure can cause the checksum recalculation for a vast amount of data. RAIDIX has developed a range of technologies that apply a fast RAID engine to enhance software-defined storage functionality.

3.3 Lustre Overview

The Lustre architecture is a storage architecture for clusters. It is best known for powering many of the largest high-performance computing (HPC) clusters worldwide, with tens of thousands of client systems, petabytes (PiB) of storage and hundreds of gigabytes per second (GB/sec) of I/O throughput. For more information refer to Lustre.

The central component of the Lustre architecture is the Lustre file system, a parallel file system which is supported on the Linux operating system and provides a POSIX standard-compliant UNIX file system interface. It is used in a wide range of HPC environments, small to large, such as AI/ML, oil and gas, seismic processing, the movie industry, and scientific research. A typical deployment of Lustre on OpenFlex Composable Infrastructure is shown in the System Architecture diagram shown below:

The Lustre software supports a variety of high-performance, low-latency networks and permits Remote Direct Memory Access (RDMA) for InfiniBand (utilizing Open Fabrics Enterprise Distribution – OFED), Intel Omni Path, and other advanced networks for fast and efficient network transport. Multiple RDMA networks can be bridged using Lustre routing for maximum performance. The Lustre software also includes integrated network diagnostics.

3.3.1 Lustre Components

An installation of the Lustre software includes a Management Server (MGS) and one or more Lustre file systems interconnected with Lustre networking (LNet) as shown in the figure below.

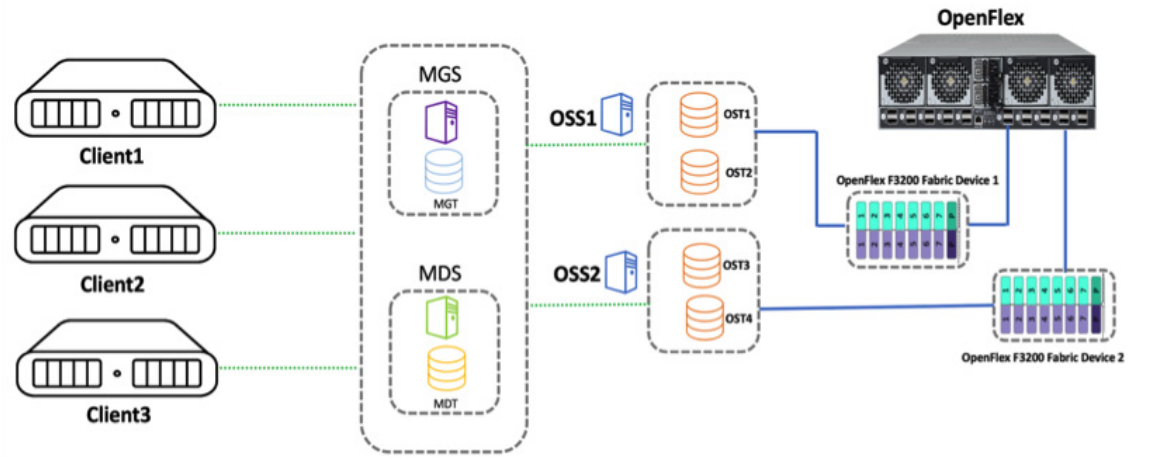


Figure 2 – Lustre File System Deployment using OpenFlex F3200 Volumes

Management Server (MGS)

The MGS stores configuration information for all the Lustre file systems in a cluster and provides this information to other Lustre components. Each Lustre target contacts the MGS to provide information, and Lustre clients contact the MGS to retrieve information.

It is preferable that the MGS have its own storage space so that it can be managed independently. However, the MGS can be co-located and share storage space with an MDS.

Lustre File System Components

Each Lustre file system consists of the following components:

Metadata Servers (MDS)

- The MDS makes metadata stored in one or more MDTs available to Lustre clients.
- Each MDS manages the names and directories in the Lustre file system(s) and provides network request handling for one or more local MDTs.

Metadata Targets (MDT)

- Each file system has at least one MDT, which holds the root directory.
- It stores metadata (such as file names, directories, permissions, and file layout) on storage attached to an MDS.

- An MDT on a shared storage target can be available to multiple MDSs, although only one can access it at a time. If an active MDS fails, a second MDS node can serve the MDT and make it available to the clients. This is referred to as MDS failover.
- Multiple MDTs are supported with the Distributed Namespace Environment (DNE). In addition to the primary MDT that holds the file system root, it is possible to add additional MDS nodes, each with its own MDTs, to hold sub-directory trees of the file system.

Object Storage Servers (OSS)

- The OSS provides file I/O service and network request handling for one or more local OSTs.
- Typically, an OSS serves between two and eight OSTs, up to 16TiB each.

Object Storage Target (OST)

- User file data is stored in one or more objects, each object on a separate OST in a Lustre file system.
- The number of objects per file is configurable by the user and can be tuned to optimize performance for a given workload.

Lustre Clients

- Lustre clients are computational, visualization, or desktop nodes that are running client software, allowing them to mount the Lustre file system.

3.3.2 Lustre Networking (LNet)

In a cluster using a Lustre file system, the Lustre network is the network connecting the OSS, MGS, MDS servers, and the clients. A Lustre network can have multiple LNet subnets and it permits end-to-end read/write throughput at or near peak bandwidth rates on a variety of network interconnects. For more information refer to LNet.

Key features of LNet include:

- Support for many commonly used network types, such as InfiniBand and IP networks.
- RDMA, when supported by underlying networks (InfiniBand, RDMA over Converged Ethernet (RoCE), Omni Path Architecture (OPA)).
- High availability and recovery using Multi-Rail.
- Support of multiple network types simultaneously.
- Routing among disparate networks.
- End-to-end read/write throughput at or near peak bandwidth rates.

The Lustre networking stack is composed of two layers, the LNet code module and the Lustre Network Driver (LND). The LNet layer is connectionless and asynchronous, and does not verify that data has been transmitted, while the LND layer is connection-oriented and typically does verify data transmission.

LNets are uniquely identified by a label composed of a string corresponding to an LND and a number, such as tcp0, o2ib0, or o2ib1, that uniquely identifies each LNet. Each node on an LNet has at least one network identifier (NID). A NID is a combination of the address of the network interface and the LNet label in the form: address@LNet_label.

Examples:

192.168.1.2@tcp0

10.13.24.90@o2ib1

4. Lustre Solution Design on OpenFlex

Western Digital, a pioneer in reliable, high-density industry-standard hardware for software-defined storage projects, is partnering with Lustre to provide a global parallel file system which is ideal for high-end HPC cluster I/O systems. The following sections of this paper provide an overview of the storage solution.

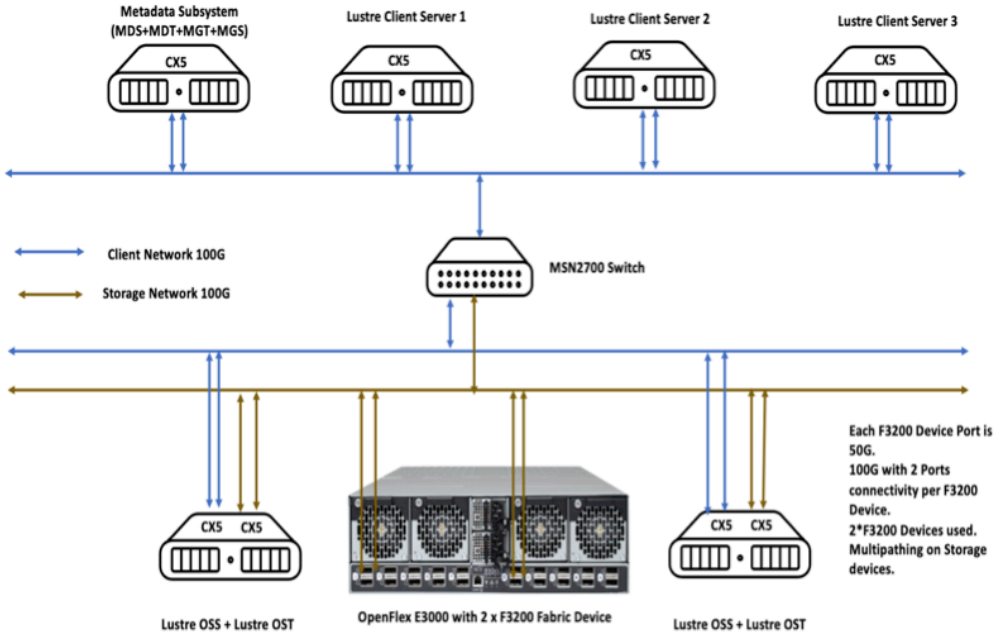


Figure 3 – Topology Diagram of Lustre File System on OpenFlex F3200

Hardware Requirement

Lustre Storage Server (OSS) Details

Storage product	OpenFlex E3000 with 2 x F3200 fabric devices
Storage interface	Dual QSFP28 (2 × 50Gb) per fabric device
Host OS	Red Hat® Enterprise Linux release 8.3 (Ootpa)
Kernel	4.18.0–240.1.1.el8_lustre.x86_64
Host NIC	2 x CX5 – MCX516A-CCAT (100Gbps)
CX5 OFED package version	5.0–0 (Inbox Drivers)
CPU	Intel® Xeon® Gold 6240R CPU @ 2.40GHz
CPU core details	Dual socket server with 24-core CPU each. 96 logical cores in total with HT enabled
Memory	128GiB
NIC firmware version	16.29.1016 (MT_0000000012)
No of volumes on each storage fabric	16

Lustre Management (MGS) / Metadata Server (MDS) Details

Host OS	Red Hat Enterprise Linux release 8.3 (Ootpa)
Kernel	4.18.0–240.1.1.el8_lustre.x86_64
Host NIC	1 x CX5 – MCX516A-CCAT (100Gbps)
CX5 OFED package version	5.0–0 (Inbox Drivers)
CPU	Intel Xeon Gold 6242 CPU @ 2.80GHz
CPU core details	Dual socket server with 16-core CPU each. 64 logical cores in total with HT enabled
NIC firmware version	16.29.1016 (MT_0000000012)
Memory	128GiB
NVMe disk	6 (7.68TB each)

Lustre Client Server Details	
Host OS	Red Hat Enterprise Linux release 8.3 (Ootpa)
Kernel	4.18.0–240.el8.x86_64
Host NIC	1 x CX5 – MCX516A-CCAT (100Gbps)
CX5 OFED package version	5.0–0 (Inbox Drivers)
CPU	Intel Xeon Gold 6240R CPU @ 2.40GHz
CPU core details	Dual socket server with 24-core CPU each. 96 logical cores in total with HT enabled
NIC firmware version	16.29.1016 (MT_0000000012)
Memory	128GiB

Below are the detailed configurations used for this deployment:

- 1 x OpenFlex E3000 fabric enclosure.
- 2 x OpenFlex F3200 fabric devices. 16 volumes on each F3200 device.
- 2 x storage servers and 2 x CX5 cards per server. Each storage server connected to 16 volumes of F3200 device. Each OST is configured using RAIDIX RAID5 (7+1) volumes created with 8 volumes mapped from OpenFlex F3200 fabric device.
- 1 x management server and 1 x CX5 card per server. 2 x NVMe SSDs on each management server to create RAIDIX RAID1 volume.
- 1 x metadata server and 1 x CX5 card per server. 4 x NVMe SSDs each per metadata server to create RAIDIX RAID10 volume.
- 3 x client servers and 1 x CX5 card per server.
- Mellanox® MSN2700 – 32-port switch.
- Each F3200 device is 100G-capable.

Lustre deployment with OpenFlex requires the below steps.

- Set up the OpenFlex system.
- Set up the storage servers, metadata servers and client servers' prerequisites.
- Install Lustre packages on all servers (management, metadata, storage, and client).
- Set up LNet on all servers (management, metadata, storage, and client).
- Set up Lustre services on all servers (management, metadata, storage, and client).
- Apply tuning parameters to improve performance.

Pre-requisites on all servers

1. Configure NTP on all servers and sync the servers.

2. Disable Selinux on all systems. Edit the below file and set the value to disabled.

```
$ vi /etc/selinux/config
$ SELINUX=disabled
```

3. Reboot the server after disabling Selinux.

4. Install the InfiniBand tools to support the inbox drivers for Lustre RDMA support.

```
$ yum-config-manager --enable rhel-8-for-x86_64-appstream-rpms,rhel-8-for-x86_64-baseos-rpms
$ yum groupinstall infiniband
```

4.1 Set Up OpenFlex System

Refer to Openflex-Composable-Infrastructure for more details on OpenFlex. Follow the steps mentioned in OpenFlex User Guide to set up the OpenFlex system.

1. Set up lossless configuration on OpenFlex server. Lossless settings need to be done on the switch and servers too. For details regarding lossless configuration contact @WD Support.

2. Initialize OpenFlex fabric with SET8 configuration.
3. Create 2 volumes per pool with 1TB capacity each. So in total create 16 volumes out of 8 pools on each OpenFlex fabric device.
4. Map the OpenFlex volumes using the nvme-cli command to appropriate servers, after configuring lossless settings on the OpenFlex devices, switch, and servers.

4.2 Set Up Storage Server

4.2.1 Install Mellanox Tools and MFT Packages

1. Install Mellanox tools on the servers by downloading the latest OFED iso for RHEL 8.3 Mellanox site https://www.mellanox.com/products/infiniband-drivers/linux/mlnx_ofed.

```
$ [root@lustre-server1 ~]# mount -o ro,loop MLNX_OFED_LINUX-5.4-1.0.3.0-rhel8.4-x86_64.iso /mnt
$ [root@lustre-server1 ~]# cd /mnt/RPMS
$ [root@lustre-server1 RPMS]# rpm -ivh mlnx-tools-5.2.0-0.54103.x86_64.rpm
```

2. Download mft tools from the link https://www.mellanox.com/downloads/MFT/mft-4.17.0-106-x86_64-rpm.tgz, then extract the packages and install them.

```
$ [root@lustre-server1 mft]# wget https://www.mellanox.com/downloads/MFT/mft-4.17.0-106-x86_64-rpm.tgz
$ [root@lustre-server1 mft]# tar -zxf mft-4.17.0-106-x86_64-rpm.tgz
$ [root@lustre-server1 mft]# cd mft-4.17.0-106-x86_64-rpm
$ [root@lustre-server1 mft-4.17.0-106-x86_64-rpm]# ./install.sh
```

3. Run "mst start" then if it fails to start reboot the host.
4. Once the servers boot up, run "mst start" again.

4.2.2 Set Up Lossless Configuration

1. Set up lossless configuration on the host servers by running the below commands on all the Lustre storage servers.

```
modprobe -v nvme-core multipath=no
modprobe -v nvme
modprobe -v nvme-rdma
mlnx_qos -i ens1f0
mlnx_qos -i ens1f1
mst start
mst status
yes | mlxconfig -d /dev/mst/mt4119_pciconf0 set LLDP_NB_DCBX_P1=FALSE LLDP_NB_TX_MODE_P1=2 LLDP_NB_RX_MODE_P1=2 LLDP_NB_DCBX_P2=FALSE LLDP_NB_TX_MODE_P2=2 LLDP_NB_RX_MODE_P2=2
yes | mlxconfig -d /dev/mst/mt4119_pciconf0.1 set LLDP_NB_DCBX_P1=FALSE LLDP_NB_TX_MODE_P1=2 LLDP_NB_RX_MODE_P1=2 LLDP_NB_DCBX_P2=FALSE LLDP_NB_TX_MODE_P2=2 LLDP_NB_RX_MODE_P2=2
mlnx_qos -i ens1f0 --trust dscp
mlnx_qos -i ens1f1 --trust dscp
mlnx_qos -i ens1f0 --pfc 0,0,0,1,0,0,0,0
mlnx_qos -i ens1f1 --pfc 0,0,0,1,0,0,0,0
mlnx_qos -i ens1f0 ----buffer_size 130944,130944,0,0,0,0,0,0
mlnx_qos -i ens1f1 --prio2buffer 0,0,0,1,0,0,0,0
mlnx_qos -i ens1f0 --tsa ets,ets,ets,ets,ets,ets,strict,ets --tcbw 14,15,14,15,14,14,0,14
```

```
mlnx_qos -i ens1f1 ----buffer_size 130944,130944,0,0,0,0,0,0
mlnx_qos -i ens1f0 --prio2buffer 0,0,0,1,0,0,0,0
mlnx_qos -i ens1f0 --tsa ets,ets,ets,ets,ets,ets,strict,ets --tcbw
14,15,14,15,14,14,0,14
cma_roce_mode -d mlx5_0 -m 2
cma_roce_mode -d mlx5_1 -m 2
cma_roce_tos -d mlx5_0 -t 96
cma_roce_tos -d mlx5_1 -t 96
cma_roce_tos -d mlx5_0
cma_roce_tos -d mlx5_1
sleep 3
```

2. Set up lossless configuration on the switch and on all Lustre storage servers where volumes from OpenFlex devices will be mapped. For more details regarding lossless configuration on OpenFlex devices and switches, visit www.westerndigital.com/support and follow the steps mentioned in the lossless configuration guide for your switch brand.
3. Set up the IPs. Set the MTU value to 4500 for all storage ports and the MTU value to 9216 for other ports on the host as well as on the switch. Verify all the ports are up and configured properly.

4.2.3 Set Up DM Multipath on All Storage Servers

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

```
Defaults {
uid_attribute ID_WWN
user_friendly_names yes
path_grouping_policy  multibus
failback immediate
path_selector “round-robin 0”
no_path_retry queue
}
blacklist_exceptions {
property “(ID_WWN|SCSI_IDENT_.*|ID_SERIAL|DEVTYPE)”
devnode “nvme*”
}
```

2. Enable and restart the multipathd service.

```
$ systemctl enable multipathd
$ systemctl restart multipathd
```

3. Add the below details under the grub file to unload native multipath. Reboot the system.

```
$ grubby --update-kernel=ALL --args=”nvme_core.multipath=N”
```

4. Check the server after reboot and follow the steps mentioned in section 4.2.2 Set up Lossless Configuration to set up lossless, and to load NVMe modules. Map the volumes. Verify the multipath configuration details once volumes are mapped.

```
$ [root@lustre-server1 ~]# multipath -ll
mpatham (uuid.ae7dbf5a-1717-47f6-8b99-47cfa011b0af) dm-4 NVME,OpenFlex F3200
size=1.0T features=’1 queue_if_no_path’ hwhandler=’0’ wp=rw
`-+- policy=’round-robin 0’ prio=1 status=active
|- 4:1:1:1  nvme4n1  259:2  active ready running
`- 5:2:1:1  nvme5n1  259:3  active ready running
```

```
mpathal (uuid.deeccc75-f1dd-4c0b-a0d4-6a30f319bda0) dm-3 NVME,OpenFlex F3200
size=1.0T features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
|- 2:1:1:1  nvme2n1  259:0  active ready running
`- 3:2:1:1  nvme3n1  259:1  active ready running
```

5. Repeat the above steps for all storage servers.
6. Verify that the following conditions are true:
   - The hosts can ping each other's IPs and host names.
   - Volumes are assigned to the appropriate hosts.

4.2.4 Set Up DM Multipath When Native Multipath Is Already Configured

1. Disconnect all previously mapped volumes. Ensure the volume got disconnected successfully.

```
$ [root@ lustre-server1 ~]# nvme disconnect-all
$ [root@ lustre-server1 ~]# nvme list
```

2. Unload all the NVMe modules.

```
$ [root@ lustre-server1 ~]# lsmod | grep nvme
$ [root@ lustre-server1 ~]# rmmod nvme_rdma
$ [root@ lustre-server1 ~]# rmmod nvme_fabrics
$ [root@ lustre-server1 ~]# rmmod nvme
$ [root@ lustre-server1 ~]# rmmod nvme_core
$ [root@ lustre-server1 ~]# lsmod | grep nvme
```

3. Add the below details under the grub file to unload native multipath. Reboot the system.

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

4. Check the server after reboot and follow the steps mentioned in @4.2.2 Set up Lossless Configuration and set up lossless, reload NVMe modules, and configure volume mapping. Verify the multipath configuration details once volumes are mapped after boot.

```
$ [root@lustre-server1 ~]# multipath -ll
mpatham (uuid.ae7dbf5a-1717-47f6-8b99-47cfa011b0af) dm-4 NVME,OpenFlex F3200
size=1.0T features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
|- 4:1:1:1  nvme4n1  259:2  active ready running
`- 5:2:1:1  nvme5n1  259:3  active ready running
mpathal (uuid.deeccc75-f1dd-4c0b-a0d4-6a30f319bda0) dm-3 NVME,OpenFlex F3200
size=1.0T features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
|- 2:1:1:1  nvme2n1  259:0  active ready running
`- 3:2:1:1  nvme3n1  259:1  active ready running
```

4.2.5 Set Up RAIDIX ERA Software on All Storage Servers

Get a valid license and download the ERA RAID utility on the server. The RAIDIX ERA 3.3.0 distribution consists of four software packages for Linux OS.

- eraraid – the main functionality which is the driver for RAID arrays.
- eraraid-util – a user management utility for RAIDs and licenses.

The package type depends on your Linux distribution. To get the license and rpm packages refer to RAIDIX-ERA.

1. After downloading the rpm, to install RAIDIX ERA 3.3.0, unpack the archive packages with the distribution to a folder and go to the folder by executing the following commands.

```
$ tar xzf raidix_era.tar.gz
$ cd raidix_era 3.3.0
```

2. To install RAIDIX ERA 3.3.0, run the installation script installer.sh, which is located in the archive with program packages.

```
$ ./installer.sh
```

3. While running the script it will ask you to accept the terms of the license agreement. You will see the text of the agreement on your screen. It will ask you to confirm the installation of DKMS. Check the presence of the required packages with the kernel header files. Install the required dependencies from the repositories. For more details refer to RAIDIX_ERA_Installation_Guide.

4. If the installation fails then check and install all the required dependencies. After installing all required dependencies, rerun ./installer.sh script. Once the installation is done, it will ask you to start the ERA services.

5. To ensure that the packages were installed successfully, run the below command, and check the status.

```
$ eraraid show
```

6. Once you have your license file, copy it to the server (for example, to the /root/raidix_era directory), and apply the license key by running the command before proceeding with RAIDIX installation and configuration.

```
$ eraraid license --update /root/license.txt
```

7. Create a RAIDIX ERA RAID disk of level 5 with a set of eight volumes mapped from OpenFlex fabric using multipath. Similarly create one more RAID5 disk with another set of eight volumes.

```
$ eraraid create -n era1 -l 5 -d /dev/mapper/mpatham /dev/mapper/mpathan /dev/mapper/
mpathao /dev/mapper/mpathap /dev/mapper/mpathaq /dev/mapper/mpathar /dev/mapper/
mpathas /dev/mapper/mpathat -ss 16 -bs 4096
$ eraraid create -n era2 -l 5 -d /dev/mapper/mpathau /dev/mapper/mpathav /dev/mapper/
mpathaw /dev/mapper/mpathax /dev/mapper/mpathay /dev/mapper/mpathaz /dev/mapper/
mpathba /dev/mapper/mpathbb -ss 16 -bs 4096
```

8. Check the ERA RAID disk initialization status. It will take time to initialize the RAID disk. Once RAID is initialized, /dev/era_era1 and /dev/era_era2 can be used for OST creation.

```
[root@lustre-server1 ~]# eraraid show
RAIDs
```

name	static	state	devices	info
era1	size: 7166 GiB level: 5 strip_size: 16 block_size: 4096 sparepool: - active: True config: True	online initialized	0 /dev/mapper/mpatham online 1 /dev/mapper/mpathan online 2 /dev/mapper/mpathao online 3 /dev/mapper/mpathap online 4 /dev/mapper/mpathaq online 5 /dev/mapper/mpathar online 6 /dev/mapper/mpathas online 7 /dev/mapper/mpathat online	memory_usage_mb : -
era2	size: 7166 GiB level: 5 strip_size: 16 block_size: 4096 sparepool: - active: True config: True	online initialized	0 /dev/mapper/mpathau online 1 /dev/mapper/mpathav online 2 /dev/mapper/mpathaw online 3 /dev/mapper/mpathax online 4 /dev/mapper/mpathay online 5 /dev/mapper/mpathaz online 6 /dev/mapper/mpathba online 7 /dev/mapper/mpathbb online	memory_usage_mb : -

4.3 Set Up Management/Metadata Server

1. Set up RAIDIX ERA software on all management/metadata servers. Please follow the same steps mentioned in section 4.2.5 Set Up RAIDIX ERA Software on All Storage Servers up to and including step 6.
2. Create a RAIDIX ERA RAID disk of level 1 with two NVMe SSDs on the management server.

```
$ eraraid create -n era1 -l 1 -d /dev/mapper/mpathb /dev/mapper/mpathc -ss 16 -bs 4096
```

3. Create a RAIDIX ERA RAID disk of level 10 with four NVMe SSDs on metadata server.

```
$ eraraid create -n era1 -l 10 -d /dev/mapper/mpathd /dev/mapper/mpathe /dev/mapper/mpathf /dev/mapper/mpathg -ss 16 -bs 4096
```

4. Check the ERA RAID disk initialization status using the below command.

```
[root@lustre-meta1 ~]# eraraid show
RAIDs
```

name	static	state	devices	info
era1	size: 7153 GiB level: 1 strip_size: 16 block_size: 4096 sparepool: - active: True config: True	online initialized	0 /dev/mapper/mpathb online 1 /dev/mapper/mpathc online	memory_usage_mb : -
era2	size: 14307 GiB level: 10 group_size: 2 strip_size: 16 block_size: 4096 sparepool: - active: True config: True	online initialized	0 /dev/mapper/mpathd online 1 /dev/mapper/mpathe online 2 /dev/mapper/mpathf online 3 /dev/mapper/mpathg online	memory_usage_mb : -

5. Once RAID is initialized, /dev/era_era1 and /dev/era_era2 can be used for Lustre deployment as MGT and MDT storage.

5. Lustre Deployment with OpenFlex

5.1 Set Up Lustre Repository

1. Set up a Lustre repository on a Lustre manager server. To set up a local repository, follow below steps.
2. Install the httpd service.

```
$ yum install httpd
```

3. Configure httpd. Edit the below file and replace the server name with your own environment in the httpd.conf file.

```
$ Edit httpd.conf file
vi /etc/httpd/conf/httpd.conf
$ Change to your server's name
ServerName <servername>:80
```

4. Enable httpd.

```
$ systemctl enable httpd
```

5. If the firewall service is running, allow the HTTP service. HTTP uses 80/TCP.
6. Install the below utility to create the repo.

```
$ yum install yum-utils createrepo
```

7. Create a repository file (example: lustre.repo) under the /etc/yum.repos.d/ directory with the below contents.

```
$ cat > /etc/yum.repos.d/lustre.repo <<\__EOF
[lustre-server]
name=lustre-server
baseurl=https://downloads.whamcloud.com/public/lustre/lustre-2.14.0/el8.3/server
# exclude=*debuginfo*
gpgcheck=0

[lustre-client]
name=lustre-client
baseurl=https://downloads.whamcloud.com/public/lustre/lustre-2.14.0/el8.3/client
# exclude=*debuginfo*
gpgcheck=0

[e2fsprogs-wc]
name=e2fsprogs-wc
baseurl=https://downloads.whamcloud.com/public/e2fsprogs/latest/el8
# exclude=*debuginfo*
gpgcheck=0
__EOF
```

8. Use the repo sync command (distributed in the yum-utils package) to download mirrors of the Lustre repositories to the manager server (example: lustre-server1).

```
$ mkdir -p /var/www/html/repo
$ cd /var/www/html/repo
$ chmod -R 755 /var/www/html/repo
$ reposync -c /etc/yum.repos.d/lustre.repo -n --repo lustre-server --repo lustre-client --repo e2fsprogs-wc
```

9. Create the repository metadata.

```
$ cd /var/www/html/repo
for i in e2fsprogs-wc lustre-client lustre-server; do
(cd $i && createrepo.)
done
```

10. Enable the python39-devel module.

```
$ yum module enable python39-devel
```

11. Create a yum repository definition file. The following script creates a file containing repository definitions for the Lustre packages and stores it in the web server static content directory. This makes it easy to distribute to the Lustre servers and clients. Review the content and adjust according to the requirements of the target environment. Run the script on the web server host.

```
$ hn=`hostname --fqdn`
cat >/var/www/html/lustre.repo <<__EOF
```

```
[lustre-server]
name=lustre-server
baseurl=https://$hn/repo/lustre-server
enabled=0
gpgcheck=0
proxy=_none_

[lustre-client]
name=lustre-client
baseurl=https://$hn/repo/lustre-client
enabled=0
gpgcheck=0

[e2fsprogs-wc]
name=e2fsprogs-wc
baseurl=https://$hn/repo/e2fsprogs-wc
enabled=0
gpgcheck=0
__EOF
```

12. Apply any configuration changes that may be necessary for the web server to incorporate the new bundle directories. You may need to reload the configuration, or restart the web service when done.
13. Copy the Lustre repo definition file onto each of the Lustre servers and clients, in the directory /etc/yum.repos.d/. You can use utilities like curl and wget to retrieve the file from the web server as part of a configuration management system rule/promise or during system provisioning.
14. Start the apache httpd service on the Lustre server.

```
$ systemctl start httpd
```

15. Install the below packages on the servers that will use DKMS to install the Lustre software.

```
$ yum install asciidoc audit-libs-devel automake bc binutils-devel bison device-
mapper-devel elfutils-devel elfutils-libelf-devel expect flex gcc gcc-c++ git glib2
glib2-devel hmaccalc keyutils-libs-devel krb5-devel ksh libattr-devel libblkid-
devel libselinux-devel libtool libuuid-devel libyaml-devel lsscsi make ncurses-
devel net-snmp-devel net-tools newt-devel numactl-devel parted patchutils
pciutils-devel perl-ExtUtils-Embed pesign python-devel redhat-rpm-config rpm-build
systemd-devel tcl tcl-devel tk tk-devel wget xmlto yum-utils zlib-devel
```

5.2 Lustre Server Software Installation with LDISKFS OSD Support

1. Install the Lustre e2fsprogs distribution.

```
$ yum --nogpgcheck --disablerepo=* --enablerepo=e2fsprogs-wc install e2fsprogs
```

2. Install the Lustre-patched kernel packages. Ensure that the Lustre repository is picked for the kernel packages, by disabling the OS repos.

```
$ yum --nogpgcheck --disablerepo=base,extras,updates --enablerepo=lustre-server
install kernel kernel-devel kernel-headers
```

3. Reboot the server.
4. After the server is back from reboot, install the LDISKFS kmod and other Lustre packages.

```
$ yum --nogpgcheck --enablerepo=lustre-server install kmod-lustre kmod-lustre-
osd-ldiskfs lustre-osd-ldiskfs-mount lustre
```

5. Load the Lustre kernel modules using the below command and verify that the software has installed correctly.

```
$ [root@lustre-server2 ~]# modprobe -v lustre
insmod /lib/modules/4.18.0-240.1.1.el8_lustre.x86_64/extra/lustre/net/libcfs.ko
insmod /lib/modules/4.18.0-240.1.1.el8_lustre.x86_64/extra/lustre/net/lnet.ko
insmod /lib/modules/4.18.0-240.1.1.el8_lustre.x86_64/extra/lustre/fs/obdclass.ko
insmod /lib/modules/4.18.0-240.1.1.el8_lustre.x86_64/extra/lustre/fs/ptlrpc.ko
insmod /lib/modules/4.18.0-240.1.1.el8_lustre.x86_64/extra/lustre/fs/fld.ko
insmod /lib/modules/4.18.0-240.1.1.el8_lustre.x86_64/extra/lustre/fs/fid.ko
insmod /lib/modules/4.18.0-240.1.1.el8_lustre.x86_64/extra/lustre/fs/osc.ko
insmod /lib/modules/4.18.0-240.1.1.el8_lustre.x86_64/extra/lustre/fs/lov.ko
insmod /lib/modules/4.18.0-240.1.1.el8_lustre.x86_64/extra/lustre/fs/mdc.ko
insmod /lib/modules/4.18.0-240.1.1.el8_lustre.x86_64/extra/lustre/fs/lmv.ko
insmod /lib/modules/4.18.0
```

6. Upon verification, unload the Lustre modules from the kernel.

```
$ lustre_rmmod
```

5.3 Lustre Client Software Installation

The Lustre client software comprises packages containing the kernel modules and separate packages for user-space tools used to manage the client software. The Lustre clients do not require a "Lustre-patched" kernel, which simplifies installation.

Execute the following steps on each machine that will be used as the Lustre client.

1. Install the kernel packages that match the latest supported version for the Lustre release.

```
$ yum install kernel kernel-devel kernel-headers kernel-abi-whitelists
```

2. Reboot the client server.

3. After the client server is back from reboot, install the EPEL repository definition. EPEL provides the DKMS software.

```
$ yum install epel-release
```

4. Install the Lustre client user-space tools and DKMS kernel module packages.

```
$ yum --nogpgcheck --enablerepo=lustre-client install lustre-client-dkms lustre-client
```

5. Load the Lustre kernel modules to verify that the software has installed correctly.

```
$ modprobe -v lustre
```

6. Upon verification, unload the Lustre modules from the kernel.

```
$ lustre_rmmod
```

5.4 Set Up LNet on All Servers

Dynamic LNet configuration is powerful and versatile, and provides administrators with easy tools to view and alter the running configuration of LNet on a host.

You must load the LNet modules into the kernel prior to making any configuration changes using lnetctl.

1. Load the LNet modules, using the below modprobe command.

```
$ modprobe -v lnet
$ modprobe -v ko2iblnd
```

2. After the modules are loaded successfully, run the LNet configure sub-command to initialize the LNet service in the kernel.

```
$ lnetctl lnet configure
```

3. If multiple interfaces of the same host need to be added to o2ib network, then an ARP flux issue will occur. Run the below commands after every reboot to fix ARP flux issue for Multi-Rail node.

```
#Setting ARP so it doesn't broadcast
sysctl -w net.ipv4.conf.ens1f0.arp_ignore=1
sysctl -w net.ipv4.conf.ens1f0.arp_filter=0
sysctl -w net.ipv4.conf.ens1f0.arp_announce=2
sysctl -w net.ipv4.conf.ens1f0.rp_filter=0

sysctl -w net.ipv4.conf.ens1f1.arp_ignore=1
sysctl -w net.ipv4.conf.ens1f1.arp_filter=0
sysctl -w net.ipv4.conf.ens1f1.arp_announce=2
sysctl -w net.ipv4.conf.ens1f1.rp_filter=0

ip neigh flush dev ens1f0
ip neigh flush dev ens1f1

echo 201 ens1f0 >> /etc/iproute2/rt_tables
echo 202 ens1f1 >> /etc/iproute2/rt_tables

ip route add 192.168.10.0/26 dev ens1f0 proto kernel scope link src 192.168.10.15
table ens1f0
ip route add 192.168.10.0/26 dev ens1f1 proto kernel scope link src 192.168.10.16
table ens1f1
ip rule add from 192.168.10.15 table ens1f0
ip rule add from 192.168.10.16 table ens1f1
ip route flush cache
```

4. Run the below commands to configure LNet with Multi-Rail to add local interfaces to LNet o2ib0 and add peer details.

```
$ lnetctl net add --net o2ib0 --if ens1f0,ens1f1
$ lnetctl peer add --prim_nid 192.168.10.17@o2ib -nid 192.168.10.17
@o2ib,192.168.10.18@o2ib
$ lnetctl peer add --prim_nid 192.168.10.19@o2ib -nid 192.168.10.19
@o2ib,192.168.10.20@o2ib
```

5. Check the added LNet local configuration and peer details using the below commands.

```
$ [root@lustre-server1 ~]# lnetctl net show
net:
  - net type: lo
    local NI(s):
      - nid: 0@lo
        status: up
  - net type: o2ib
    local NI(s):
      - nid: 192.168.10.15@o2ib
        status: up
        interfaces:
            0: ens1f0
      - nid: 192.168.10.16@o2ib
        status: up
        interfaces:
            0: ens1f1
```

```
$ [root@lustre-server1 ~]# lnetctl peer show
peer:
  - primary nid: 192.168.10.17@o2ib
    Multi-Rail: True
    peer ni:
      - nid: 192.168.10.17@o2ib
        state: NA
      - nid: 192.168.10.18@o2ib
        state: NA
  - primary nid: 192.168.10.19@o2ib
    Multi-Rail: True
      peer ni:
      - nid: 192.168.10.19@o2ib
        state: NA
      - nid: 192.168.10.20@o2ib
        state: NA
```

6. LNet dynamic configuration using lnetctl is not persistent across reboot. So after adding all the peer host details using the lnetctl command, export the LNet configuration to the lnet.conf file. Once the server is up after reboot, importing this lnet.conf file will load all the LNet configuration.

```
$ lnetctl export > /etc/sysconfig/lnet.conf
$ lnetctl import /etc/sysconfig/lnet.conf
```

5.5 Set Up Lustre on MGS and MDS Server

1. Format the RAIDIX RAID1 volume (example: /dev/era_era1) to be used as a storage partition for MGT (Management Target) with lustrefs using the below command.

```
$ mkfs.lustre --fsname=lustrefs --mgs /dev/era_era1
```

2. Run the below command on the management node to format the RAIDIX RAID10 volume (example: /dev/era_era2) to be used as a storage partition for MDT (Metadata Target) with lustrefs.

```
$ mkfs.lustre --fsname=lustrefs  --mgsnode=192.168.10.11@o2ib0 --mdt /dev/era_era2
```

3. Create mgs and mdt directories to mount formatted disks.

```
$ mkdir -p /mnt/mgs /mnt/mdt
```

4. Mount the formatted disks for MGT (Management Target) and MDT (Metadata Target) to start management and metadata services.

```
$ mount -t lustre /dev/era_era1  /mnt/mgs
$ mount -t lustre /dev/era_era2  /mnt/mdt
```

5.6 Set Up Lustre on OSS Servers

1. Format the RAIDIX RAID5 volumes created for OSTs (Object Storage Targets) with lustrefs by providing the correct OST index.

```
$ mkfs.lustre --fsname=lustrefs --mgsnode=192.168.10.11@o2ib0 --ost --index=0
/dev/era_era1
$ mkfs.lustre --fsname=lustrefs --mgsnode=192.168.10.11@o2ib0 --ost --index=1
/dev/era_era2
```

2. Create ost0 and ost1 directories to mount formatted disks.

```
$ mkdir -p /mnt/ost0 /mnt/ost1
```

3. Mount the formatted disks for the OST (Object Storage Target) to start object storage services.

```
$ mount -t lustre /dev/era_era1 /mnt/ost0
$ mount -t lustre /dev/era_era2 /mnt/ost1
```

4. Run the above three steps on all the OSS servers by providing the correct OST index and mount point. Once the OSTs are formatted and mounted, the Lustre file system is ready to be mounted on Lustre client servers.

5.7 Set Up Lustre on Client Servers

1. Create a Lustre directory to mount the Lustre file system.

```
$ mkdir -p /mnt/lustre
```

2. Mount the Lustre file system on all client servers.

```
$ mount -t lustre 192.168.10.11@o2ib:/lustrefs /mnt/lustre
```

3. Run the below command to check the Lustre file system size.

```
$ [root@lustre-client1 ~]# lfs df -h
UUID                   bytes  Used     Available    Use%   Mounted on
lustrefs-MDT0000_UUID  9.9T   5.6M     9.2T         1%     /mnt/lustre[MDT:0]
lustrefs-OST0000_UUID  6.9T   1.4T     5.2T         21%    /mnt/lustre[OST:0]
lustrefs-OST0001_UUID  6.9T   800.0G   5.8T         12%    /mnt/lustre[OST:1]
lustrefs-OST0002_UUID  6.9T   940.0G   5.7T         14%    /mnt/lustre[OST:2]
lustrefs-OST0003_UUID  6.9T   1.3T     5.3T         20%    /mnt/lustre[OST:3]
filesystem_summary:    27.7T  4.4T     22.0T        17%    /mnt/lustre
```

6. Tuning Parameters

There are several performance tuning parameters that you can configure for an optimal system depending on the intended workload patterns to be used. This section will detail the tuning parameters that we configured on the Lustre testbed system. You must set the below values on the servers and clients to realize better performance.

Lustre server-specific tuning

```
$ [root@lustre-server1]# lctl set_param timeout=600
$ [root@lustre-server1]# lctl set_param ldlm_timeout=200
$ [root@lustre-server1]# lctl set_param at_min=250
$ [root@lustre-server1]# lctl set_param at_max=600
$ [root@lustre-server1]# lctl set_param obdfilter.lustrefs-OST*.brw_size=16
```

Validate the settings:

```
$ [root@lustre-server1]# lctl get_param obdfilter.lustrefs-OST*.brw_size
obdfilter.lustrefs-OST0000.brw_size=16
obdfilter.lustrefs-OST0002.brw_size=16
```

Lustre client-specific tuning

```
$ [root@lustre-client1]#lctl set_param osc.*.checksums=0
$ [root@lustre-client1]#lctl set_param timeout=600
$ [root@lustre-client1]#lctl set_param at_min=250
$ [root@lustre-client1]#lctl set_param at_max=600
$ [root@lustre-client1]#lctl set_param ldlm.namespaces.*.lru_size=2000
$ [root@lustre-client1]#lctl set_param osc.*OST*.max_rpcs_in_flight=32
$ [root@lustre-client1]#lctl set_param osc.*OST*.max_dirty_mb=64
```

The "max_pages_per_rpc" parameter is a tuneable that sets the maximum number of pages that will undergo I/O in a single RPC to that OST.

```
$ [root@lustre-client1]# lctl set_param osc.lustrefs-OST*.max_pages_per_rpc=1024
```

The "max_read_ahead_mb" is a tuneable that sets the maximum amount of data readahead on a file. These are read ahead in an RPC-sized chunk.

```
$ [root@lustre-client1]# lctl set_param llite.lustre*.max_read_ahead_mb=1024
```

The "max_rpcs_in_flight" is a tuneable that sets the maximum number of concurrent RPCs in flight to the OST. This parameter in the majority of cases will help with small file IO patterns.

```
$ [root@lustre-client1]# lctl set_param osc.lustrefs-OST*.max_rpcs_in_flight=32
```

Validate the settings:

```
$ [root@lustre-client1]# lctl get_param osc.lustrefs-OST*.max_pages_per_rpc
osc.lustrefs-OST0000-osc-ffff99a61eace000.max_pages_per_rpc=1024
osc.lustrefs-OST0001-osc-ffff99a61eace000.max_pages_per_rpc=1024
osc.lustrefs-OST0002-osc-ffff99a61eace000.max_pages_per_rpc=1024
osc.lustrefs-OST0003-osc-ffff99a61eace000.max_pages_per_rpc=1024
$ [root@lustre-client1]# lctl get_param llite.lustre*.max_read_ahead_mb
llite.lustrefs-ffff99a61eace000.max_read_ahead_mb=1024
$ [root@lustre-client1]# lctl get_param osc.lustrefs-OST*.max_rpcs_in_flight
osc.lustrefs-OST0000-osc-ffff99a61eace000.max_rpcs_in_flight=32
osc.lustrefs-OST0001-osc-ffff99a61eace000.max_rpcs_in_flight=32
osc.lustrefs-OST0002-osc-ffff99a61eace000.max_rpcs_in_flight=32
osc.lustrefs-OST0003-osc-ffff99a61eace000.max_rpcs_in_flight=32
```

7. Performance Test Details

The performance studies presented in this paper profile the capabilities of the Lustre configuration deployed on the OpenFlex E3000 enclosure with two OpenFlex F3200 devices. The goal is to quantify the capabilities of the solution, and to identify points of peak performance and the most appropriate methods for scaling.

Several performance studies were executed, stressing the configuration with different types of workloads to determine the limitations of performance and define the sustainability of that performance. We generally try to maintain a "standard and consistent" testing environment and methodology. There may be some areas where we purposely optimize server or storage configurations.

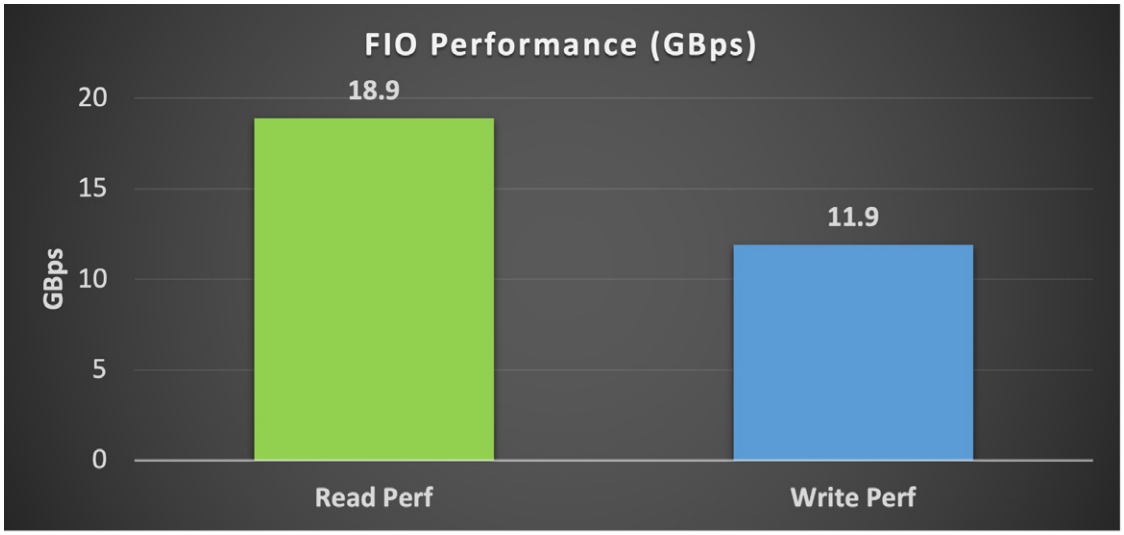
Fio tool is used to measure the performance. Fio spawns a number of threads or processes doing a particular type of IO action as specified by the user. Fio takes a number of global parameters, each inherited by the thread except when parameters are given to them and override that setting. We used three clients and ran IO simultaneously from all of them using fio to measure the performance on this deployment.

7.1 Lustre Performance on OpenFlex F3200 (Non-RAID Volumes)

Note:

- OSTs are configured on raw volumes without any RAID.
- 2 x NVMe SSD used with MDADM RAID1 configuration to store management data.
- 4 x NVMe SSD used with MDADM RAID10 configuration to store metadata.

Achieved Performance with FIO on Lustre using non-RAID Volumes	
Read	18.9GBps
Write	11.9GBps

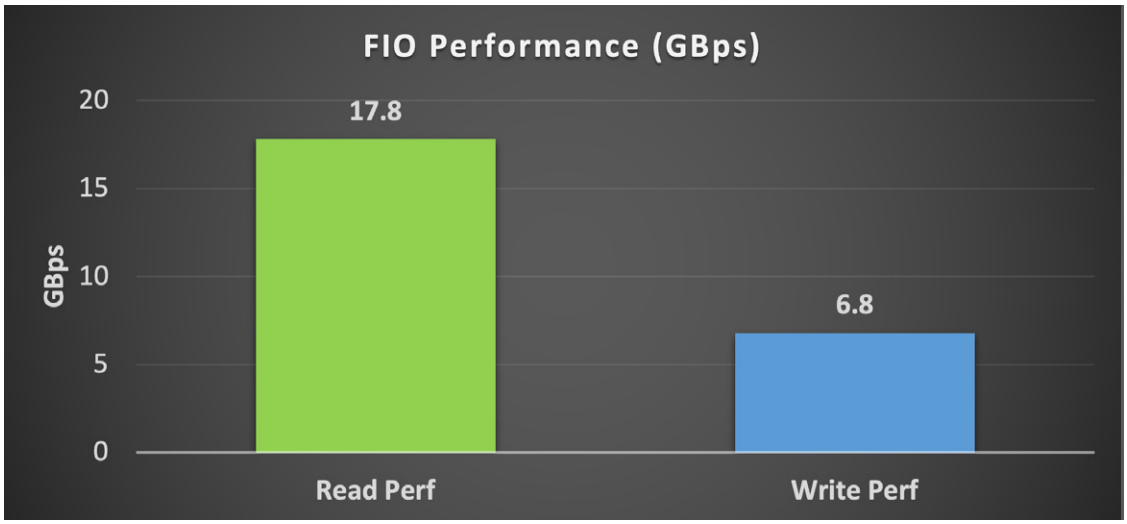


7.2 Lustre Performance on OpenFlex F3200 (RAIDIX Volumes)

Note:

- OSTs are configured using RAIDIX RAID5(7+1) configuration.
- 2 x NVMe SSD used with RAIDIX RAID1 configuration to store management data.
- 4 x NVMe SSD used with RAIDIX RAID10 configuration to store metadata.

Achieved Performance with FIO on Lustre using RAIDIX Volumes	
Read	17.8GBps
Write	6.8GBps



8. Use Cases

The combined solution of Western Digital's OpenFlex Composable Infrastructure with Lustre supports a wide range of use cases and environments, including:

- High-performance computing
- Media processing and transcoding
- Autonomous vehicles
- Big Data and financial analytics
- Machine learning
- Electronic design automation

9. Conclusion

As HPC clusters continue to grow in performance and more applications adopt parallel I/O, traditional file system such as NFS, NAS, SAN, and DAS are failing to keep up with HPC I/O demand. The Lustre parallel file system is gaining ground within the departmental and workgroup HPC space since it has proved capable of meeting the high I/O workload within HPC environments, as well as presenting a range of additional operational benefits. This paper presents a structured description of how to build a Lustre parallel file system solution using Western Digital's OpenFlex Composable Infrastructure. With this solution Western Digital is providing a hybrid approach to HPC, allowing IT organizations to re-think how they want to deploy HPC infrastructure and evolve from fixed-cost models to usage-based models. Cloud providers can provide a spectrum of services from IaaS to PaaS to full-service application offerings (SaaS) delivered by professionals with application domain expertise.

By leveraging Western Digital's customizable framework, enterprises and service providers can:

- Reduce risk by leveraging proven infrastructure components.
- Reduce development costs .
- Accelerate time to market for new application services .

Using this solution, organizations can avail themselves of the benefits of the latest NVMe-oF and CDI technologies; i.e., they can quickly deploy a proven, self-service, composable infrastructure solution, helping customers move to a more flexible, variable cost model. The solution as described shows good performance, I/O throughput, high redundancy, and high availability with a good set of data safety features, all of which are important within the HPC environment. The storage solution shows very good Lustre file system performance. We have found the Western Digital OpenFlex Composable Infrastructure and Lustre PFS solution is a good match for HPC environments.

10. References

https://www.westerndigital.com/products/data-center-platforms/openflex-composable-infrastructure

https://documents.westerndigital.com/content/dam/doc-library/en_us/assets/public/western-digital/product/platforms/openflex/user-guide-openflexf3100-western-digital.pdf

https://fio.readthedocs.io/en/latest/fio_doc.html

https://wiki.whamcloud.com/display/PUB/Lustre+Support+Matrix

https://www.lustre.org/getting-started-with-lustre/

https://www.lustre.org/download/

https://doc.lustre.org/lustre_manual.pdf

https://wiki.lustre.org/Installing_the_Lustre_Software

https://wiki.lustre.org/LNet_Router_Config_Guide#Multi-Rail_LNet_Configuration_Example

https://support.mellanox.com/s/article/getting-started-with-mellanox-firmware-tools--mft--for-linux

Western Digital.

5601 Great Oaks Parkway
San Jose, CA 95119, USA
www.westerndigital.com