LUSTRE HPC FILE SYSTEM – Architecture & DEPLOYMENT METHOD

RAHUL DHAL

SENIOR ADMINISTRATOR – HPC & BIG DATA

Agenda

- Introduction to Lustre File System
- Lustre Architecture and its components
- Understanding Failover in a Lustre File System
- System Configuration for Lustre file System
- Lustre File System Storage and I/O
- Configuring Storage for Lustre File System
- Deployment of Luster (on 4 VMs)
- Configuring a Lustre File System
- Few Useful commands hands-on

introduction of lustre file system

- Lustre is a GNU General Public licensed, open-source distributed parallel file system originally developed by Sun Microsystem and now developed and maintained by an open source community, WhamCloud.
- As per information available on the Internet, 15 of the 30 fastest supercomputers in the world use Lustre file system for high performance computing.
- It is best known for tens of thousands of client systems, petabytes (PiB) of storage and hundreds of gigabytes per second (GB/sec) of I/O throughput for HPC.
- Lustre file system can perform better than other file systems due to its strong locking and data coherency

Lustre software features are

- **NFS and CIFS export:** Lustre files can be re-exported using NFS (via Linux knfsd or Ganesha) or CIFS (via Samba)
- **Capacity growth:** The size of a Lustre file system and aggregate cluster bandwidth can be increased without interruption by adding new OSTs and MDTs to the cluster
- **Byte-granular file and fine-grained metadata locking:** Many clients can read and modify the same file or directory concurrently. The Lustre distributed lock manager (LDLM) ensures that files are coherent between all clients and servers in the file system.
- **MPI I/O:** The Lustre architecture has a dedicated MPI ADIO layer that optimizes parallel I/O to match the underlying file system architecture
- **Network data integrity protection:** A checksum of all data sent from the client to the OSS protects against corruption during data transfer.

Count…

- **Performance-enhanced ext4 file system:** The Lustre file system uses an improved version of the ext4 journaling file system to store data and metadata. This version, called ldiskfs , has been enhanced to improve performance and provide additional functionality needed by the Lustre file system.

NOTE: With the Lustre software release 2.4 and later, it is also possible to use ZFS as the backing file system for Lustre for the MDT, OST, and MGS storage.

Lustre Architecture and its components

- MDS (Metadata Server)

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.

- MDT (Metadata Target)

MDT stores metadata (such as filenames, directories, permissions and file layout) on storage attached to an MDS. Each file system has one MDT. An MDT on a shared storage target can be available to multiple MDSs, although only one can access it at a time

- MGS (Management Server)

MGS stores configuration information for all the Lustre file systems in a cluster and provides this information to other Lustre components

COUNT…

- MGT (Management Target)

MGS stores configuration on storage attached called an MGT. MGT storage requirements are small (less than 100 MB even in the largest Lustre file systems), and the data on an MGT is only accessed on a server/client mount, so disk performance is not a consideration

- OSS (Object Storage Server)

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 16 TiB each.

COUNT…

- OST (Object Storage Target)

User file data is stored in one or more objects, each object on a separate OST in a Lustre file system.

- LNET

Lustre Networking (LNet) is a custom networking API that provides the communication infrastructure that handles metadata and file I/O data for the Lustre file system servers and clients

COUNT…

- Lustre Client

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

A logical object volume (LOV) aggregates the Object Storage Clients (OSC) to provide transparent access across all the OSTs. Thus, a client with the Lustre file system mounted sees a single, coherent, synchronized namespace. Several clients can write to different parts of the same file simultaneously, while, at the same time, other clients can read from the file.

Figure 1.5. File striping on a Lustre file system

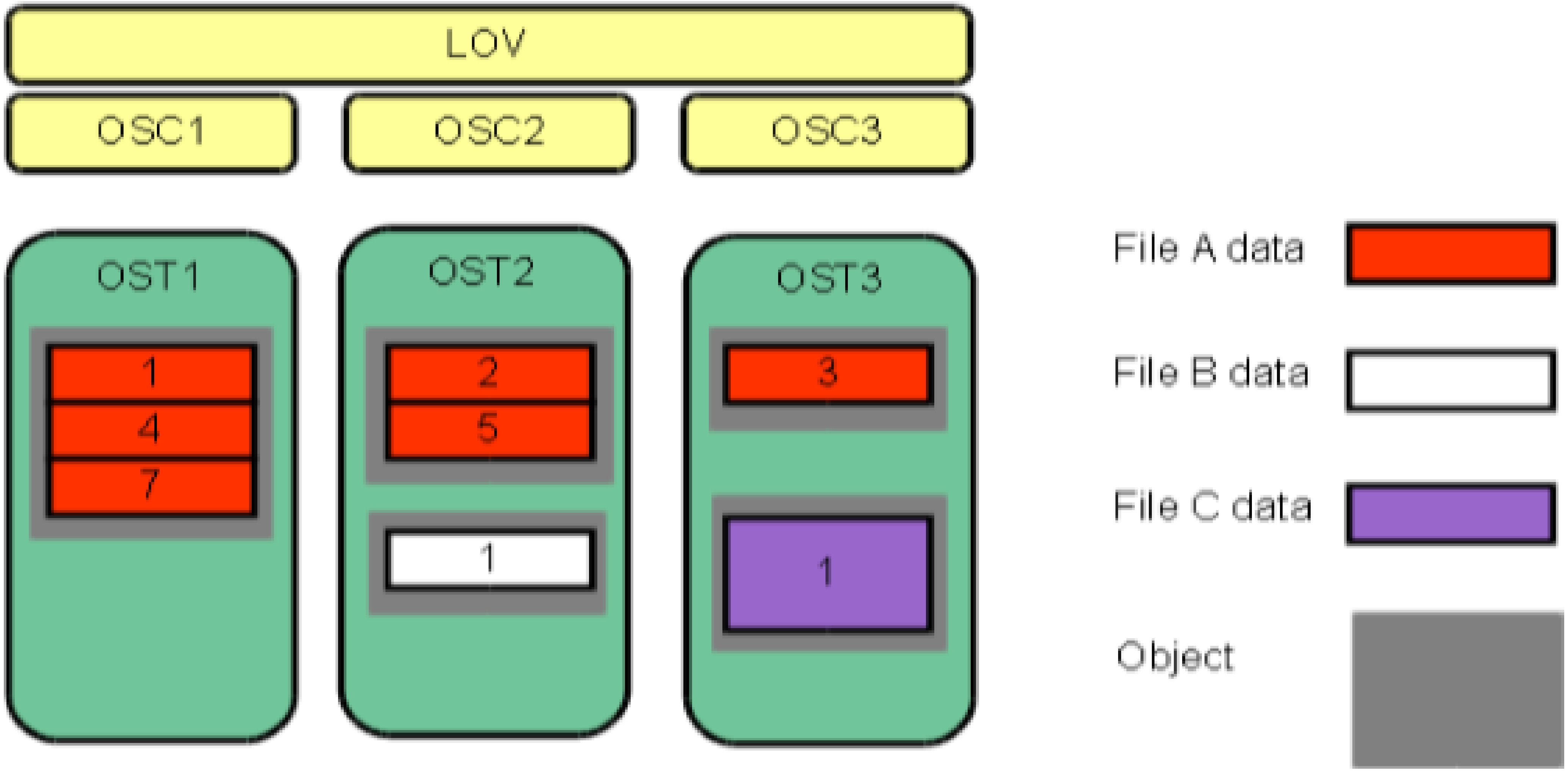
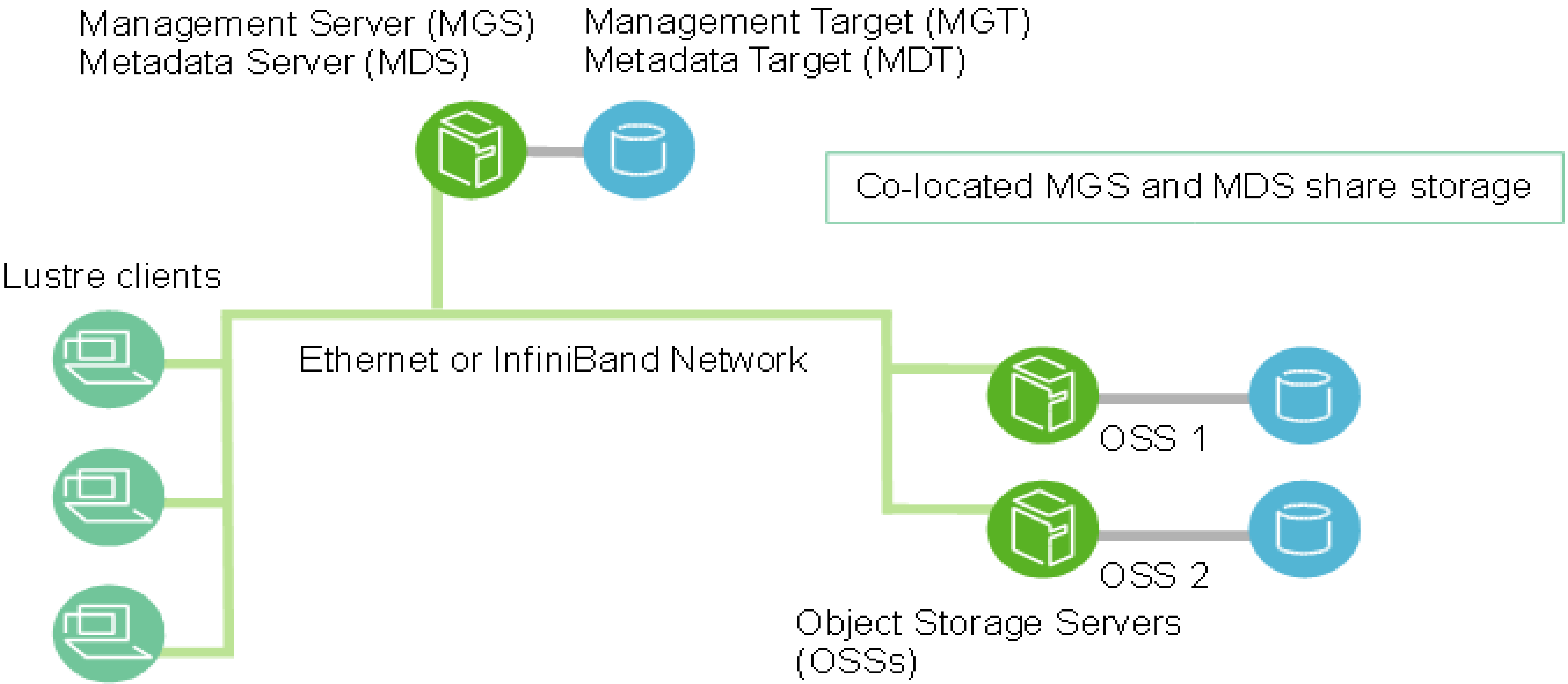


FIGURE **Lustre components in a basic cluster**



Management Server (MGS)
Metadata Server (MDS)

Management Target (MGT)
Metadata Target (MDT)

Co-located MGS and MDS share storage

Lustre clients

Ethernet or InfiniBand Network

OSS 1

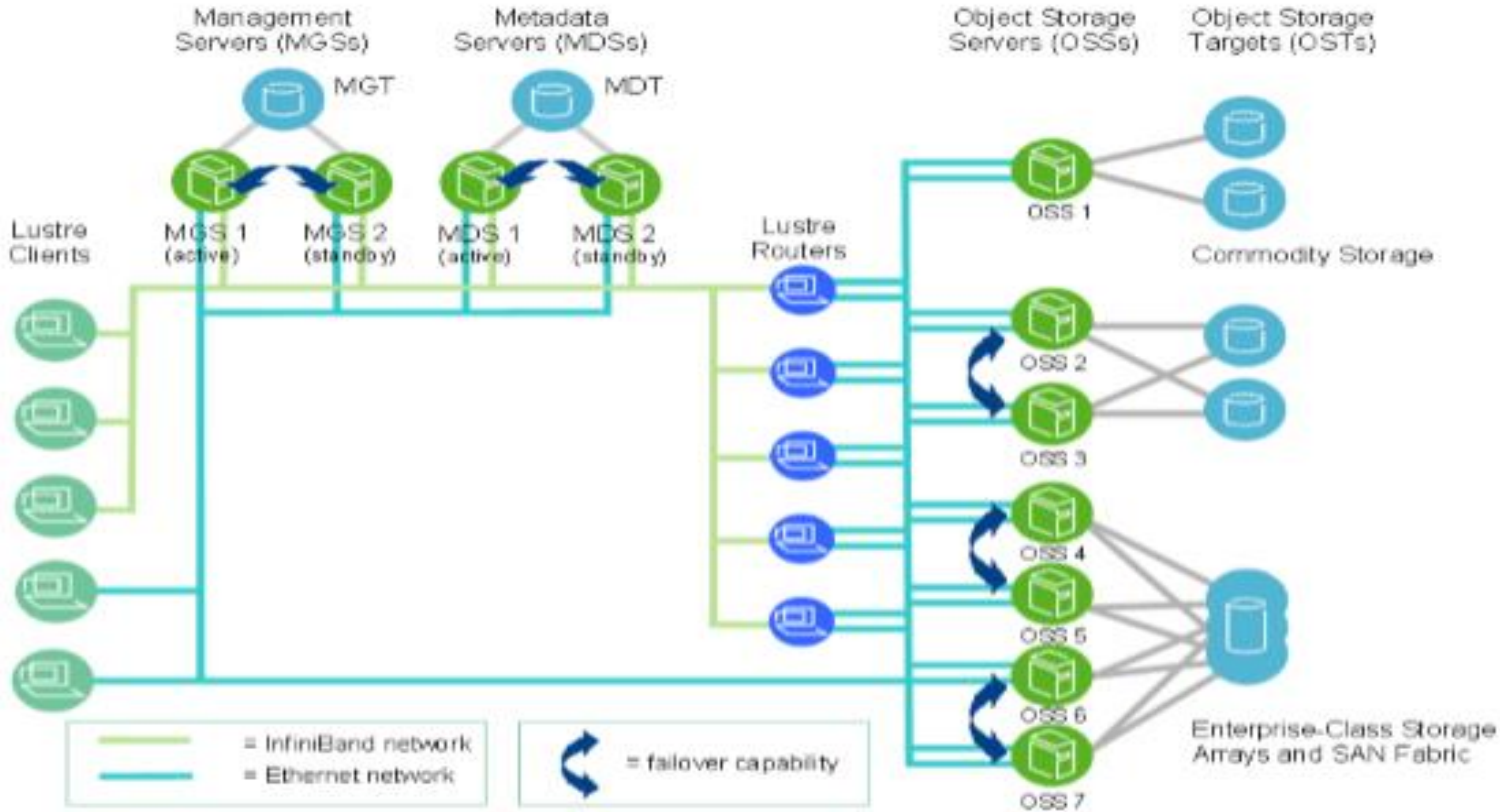
OSS 2

Object Storage Servers
(OSSs)

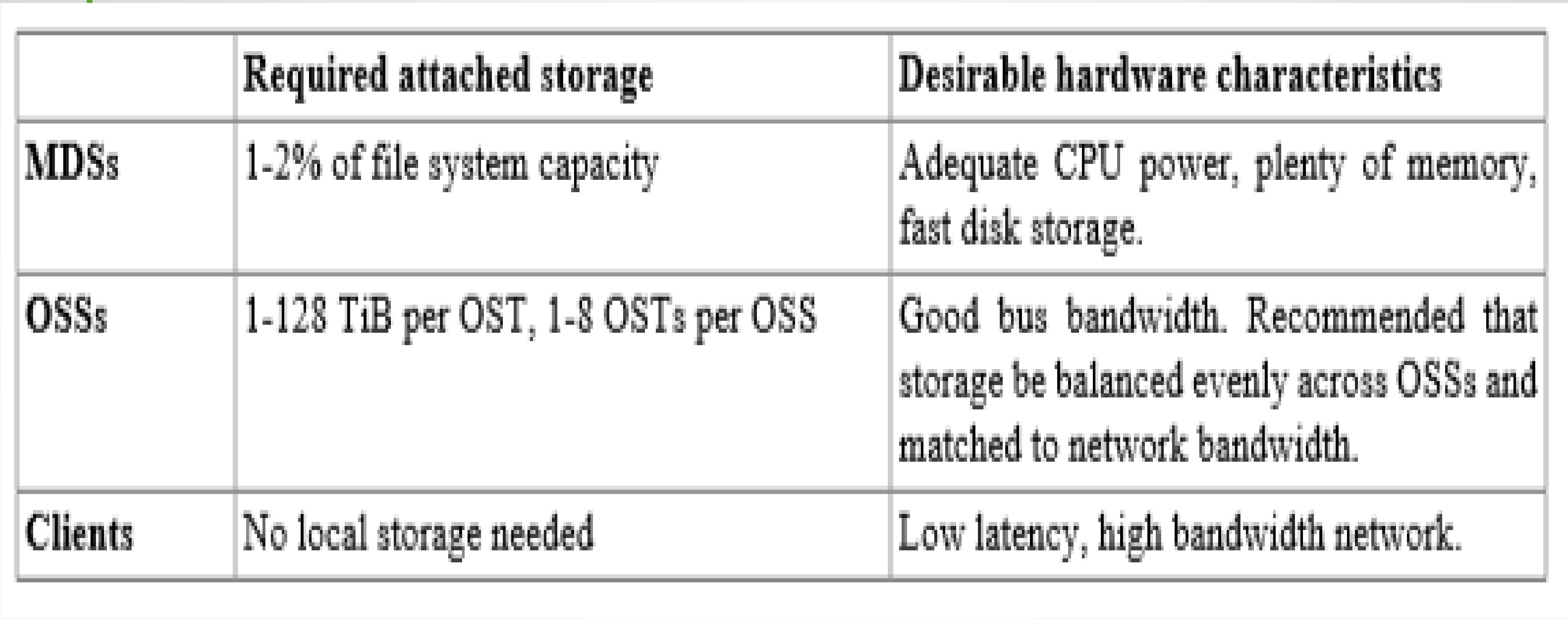
Understanding Failover in a Lustre File System

- Failover in a Lustre file system requires that two nodes be configured as a failover pair, which must share one or more storage devices. A Lustre file system can be configured to provide MDT or OST failover.
- For MDT failover, two MDSs can be configured to serve the same MDT. By placing two or more MDT partitions on storage shared by two MDSs, one MDS can fail and the remaining MDS can begin serving the unserved MDT. This is described as an active/active failover pair. (> version 2.4)
- For OST failover, multiple OSS nodes can be configured to be able to serve the same OST. However, only one OSS node can serve the OST at a time. An OST can be moved between OSS nodes that have access to the same storage device using umount/mount commands.

Figure 1.2. Lustre cluster at scale



System Configuration for Lustre file System



	Required attached storage	Desirable hardware characteristics
MDSs	1-2% of file system capacity	Adequate CPU power, plenty of memory, fast disk storage.
OSSs	1-128 TiB per OST, 1-8 OSTs per OSS	Good bus bandwidth. Recommended that storage be balanced evenly across OSSs and matched to network bandwidth.
Clients	No local storage needed	Low latency, high bandwidth network.

MDS Memory Requirements:

- Number of Clients
- Size of the directories
- Load Place on the server

Minimum requirements for the MDS is 4GB of RAM, However additional memory may significantly improve the performance.

My Recommendation: Till one PB

Memory for useable data (10) TB > 32 MDS1 and 32GB MDS2

Memory for useable data (10-30)TB > 64GB MDS1 and 64GB MDS2

More than that I will suggest to go with 128 GB Memory

OSS Memory Requirements:

- Service Threads: On OSS 4 MB I/O buffer for each OST I/O
- OSS read cache: Read Only Caching of the data using Linux page Cache to store the data its like caching from regular file system in Linux OS

So the data load is distributed among the OSS nodes and required memory is based on inode cache, locks etc.

Minimum Memory required by OSS is 4GB in case of 2 OSTs but in case of failover its 6 GB. If 4 OSTs in 2 OSS case of failover its 10GB.

COUNT….

My Recommendation: Till one PB

Memory for useable data (10) TB > 64 OSS1 and 64GB OSS2

Memory for useable data (10-30)TB > 128GB OSS1 and 128GB OSS2

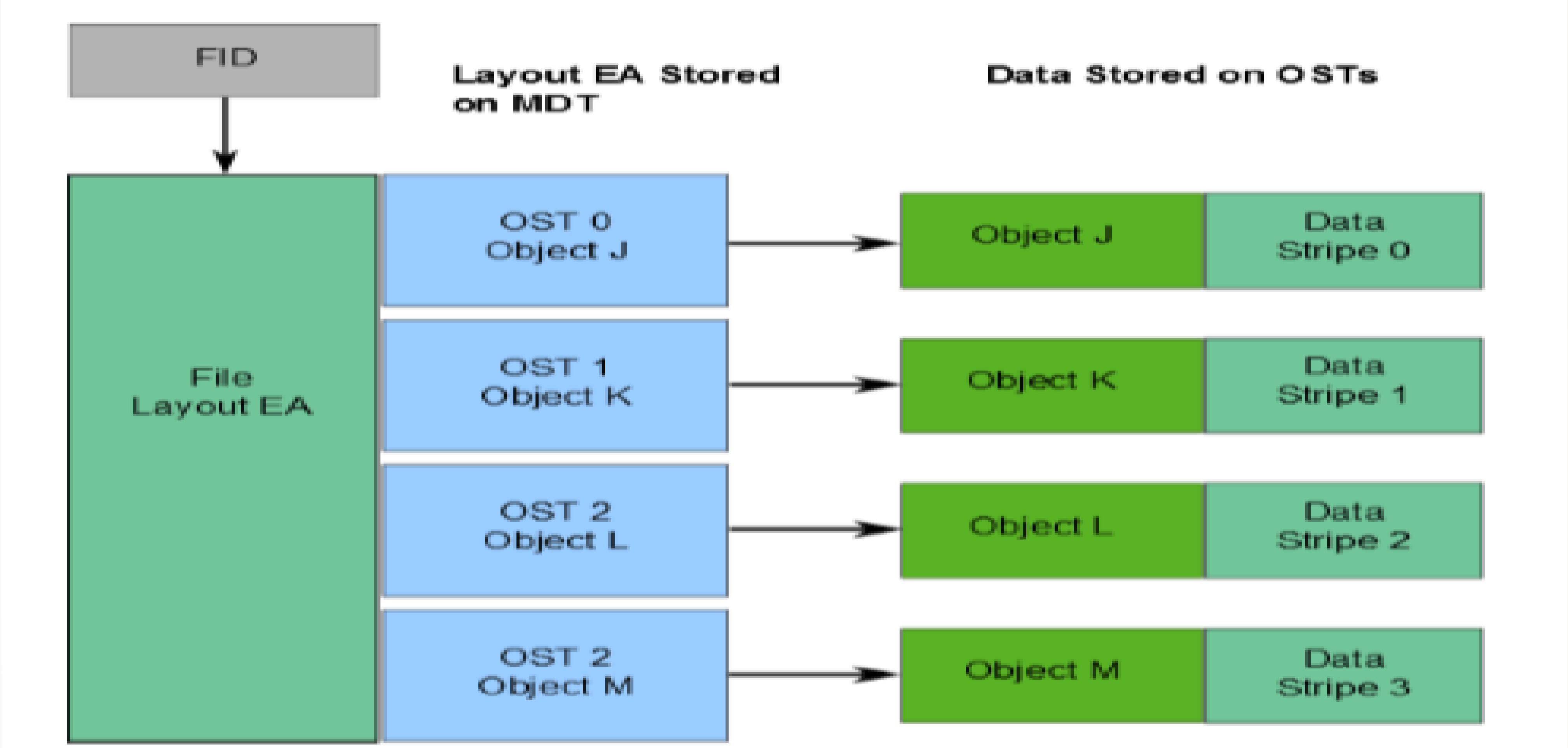
More than that I will suggest to go with 256 GB Memory

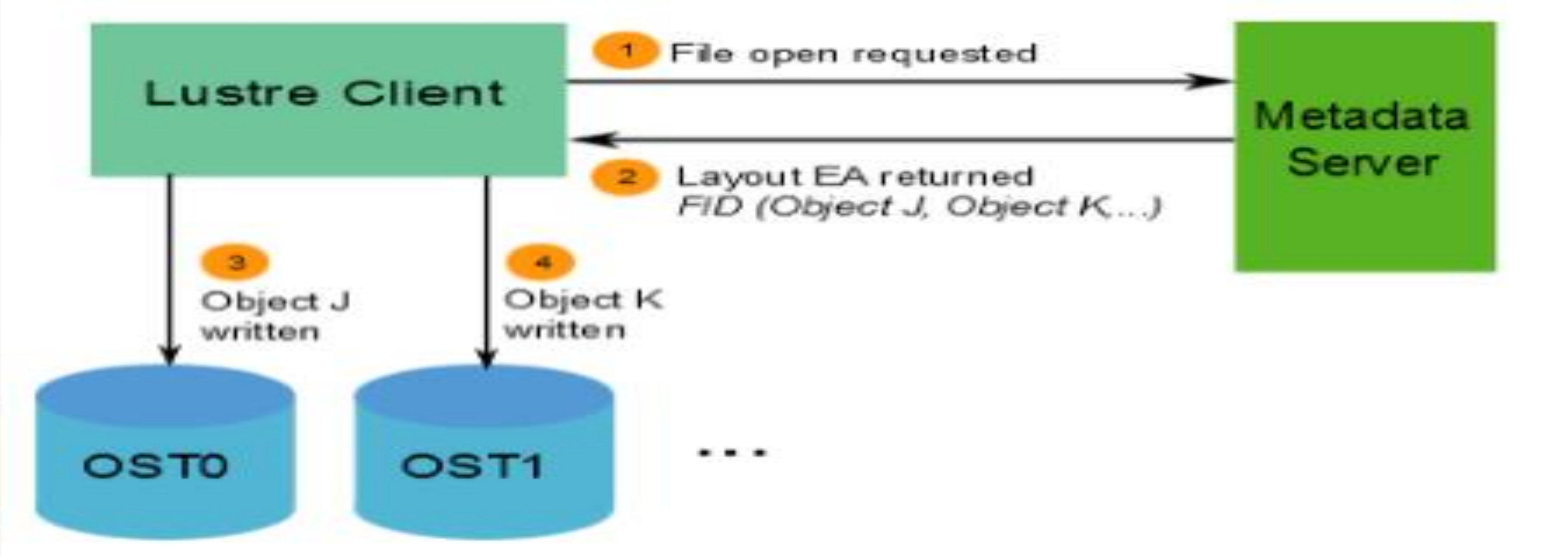
Lustre File System Storage and I/O

- Lustre file identifiers (FIDs) were introduced to replace UNIX inode numbers for identifying files or objects. A FID is a 128-bit identifier that contains a unique 64-bit sequence number, a 32-bit object ID (OID), and a 32-bit version number. The sequence number is unique across all Lustre targets in a file system (OSTs and MDTs).
- ldiskfs feature named FID-in-dirent(also known as dirdata) in which the FID is stored as part of the name of the file in the parent directory. This feature significantly improves performance for ls command executions by reducing disk I/O.

Layout EA

- Information about where file data is located on the OST(s) is stored as an extended attribute called layout EA in an MDT object identified by the FID for the file. If the file is a regular file (not a directory or symbol link), the MDT object points to 1to-N OST object(s) on the OST(s) that contain the file data. If the MDT file points to one object, all the file data is stored in that object. If the MDT file points to more than one object, the file data is striped across the objects using RAID 0, and each object is stored on a different OST


FID
Layout EA Stored on MDT
Data Stored on OSTs
File Layout EA
OST 0 Object J
OST 1 Object K
OST 2 Object L
OST 2 Object M
Object J
Object K
Object L
Object M
Data Stripe 0
Data Stripe 1
Data Stripe 2
Data Stripe 3



When a client wants to read from or write to a file, it first fetches the layout EA from the MDT object for the file. The client then uses this information to perform I/O on the file, directly interacting with the OSS nodes where the objects are stored.

Lustre File System and Striping

- One of the main factors leading to the high performance of Lustre file systems is the ability to stripe data across multiple OSTs in a round-robin fashion.
- Striping allows segments or 'chunks' of data in a file to be stored on different OSTs. In the Lustre file system, a RAID 0 pattern is used in which data is "striped" across a certain number of objects.

COUNT….

- Stripe_size

The number of bytes in each stripe this much data is written to each stripe before starting to write in the next stripe.

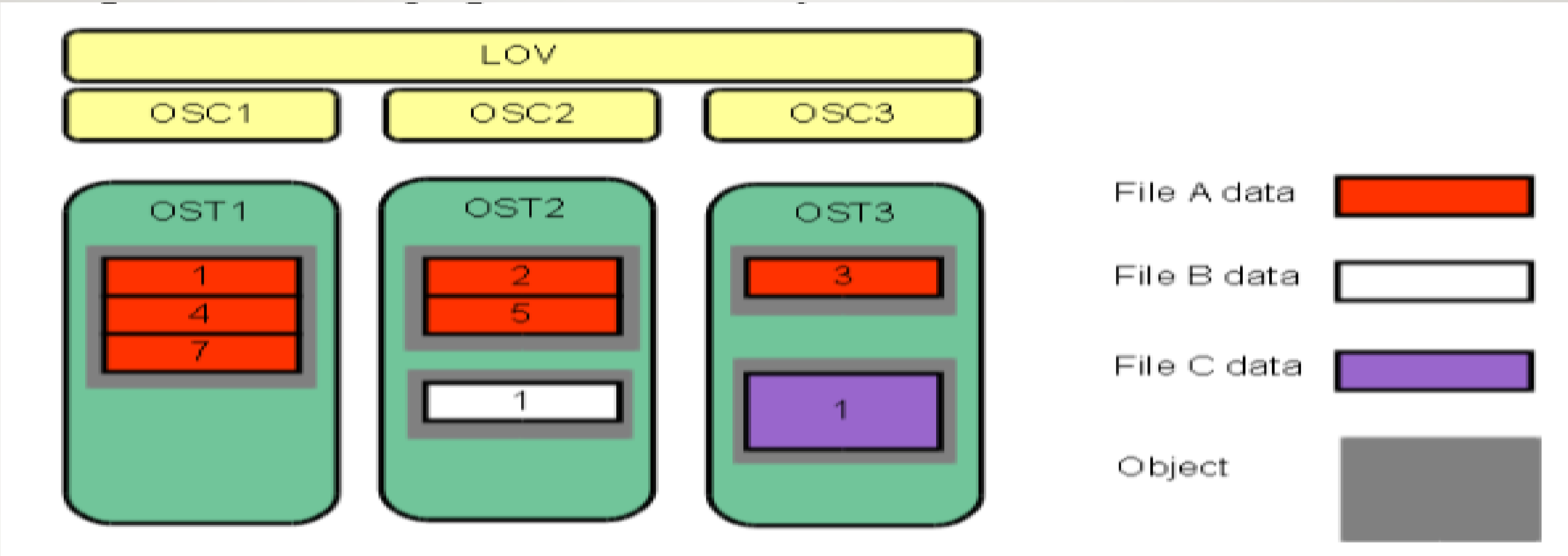
Default value : 1 MB, Max value: 4 GB, Min value : 512 KB, Recommended value : 1 MB or 4MB

- Stripe_count:

The number of the OSTs that each file is striped across. Default value is 1. Stripe count greater than 2 give good performance. So we recommend stripe count value 1 to 4 as increase the stripe count value decrease the reliability, damage to a single OST cause loss of data in many files.

- Start_OST:

The first OST where the Objects are created for each file. The default -1 allows the MDS to choose the starting index based on available space and load balancing. (Not recommended to change)



The stripe_size for File C is larger than the stripe_size for File A, allowing more data to be stored in a single stripe for File C. The stripe_count for File A is 3, resulting in data striped across three objects, while the stripe_count for File B and File C is 1.

COUNT….

- The maximum file size is not limited by the size of a single target. In a Lustre file system, files can be striped across multiple objects (up to 2000), and each object can be up to 16 TiB in size with ldiskfs, or up to 256PiB with ZFS. This leads to a maximum file size of 31.25 PiB for ldiskfs or 8EiB with ZFS.

commands

- lfs setstripe –S 4M /lustre/test/new_file
- lfs setstripe –c -1 /lustre/test/full_stripe

-1 (one) stripe over all OSTs

- lfs setstripe –stripe-count 1 –index 0 /lustre/file1

Create a file on specific OST

- lfs getstripe /lustre/full_stripe

Configuring Storage for Lustre File System

It is strongly recommended that storage (MDTs and OSTs) used in a Lustre file system be configured with hardware RAID.

- Metadata Target (MDT)

I/O on the MDT is typically mostly reads and writes of small amounts of data. For this reason, we recommend that you use RAID 1 for MDT storage. If you require more capacity for an MDT than one disk provides, we recommend RAID 1 + 0 or RAID 10

- Object Storage Server (OST)

RAID 6 is required for large clusters and RAID 5 is not acceptable. we recommend that you create RAID sets with 4 or 8 data disks plus one or two parity disks. Using larger RAID sets will negatively impact performance compared to having multiple independent RAID sets.

To maximize performance for small I/O request sizes, storage configured as RAID 1+0 can yield much better results but will increase cost or reduce capacity.

COUNT….

RAID monitoring software is recommended to quickly detect faulty disks and allow them to be replaced to avoid double failures and data loss. Hot spare disks are recommended so that rebuilds happen without delays.

Deployment of luster

- **Manual Operations**
- **Intel® Manager for Lustre* Software Installation using IEEL (https://whamcloud.github.io/Online-Help/)**

Manual Operations
1. Make systems ready

- Make 4 nodes VMs installed with CentOS-7.6 (master, clientnode, oss, mds)
- Make this configuration on all nodes

set hostname for all nodes accordingly...

set static IPs for all nodes(internal IPs).....

- Password-less ssh for all nodes

```
ssh-keygen -t rsa
cd .ssh
cat id_rsa.pub > authorized_keys
chmod 600 authorized_keys
vim config
Host *
StrictHostKeyChecking no
chmod 600 config
```

- copy .ssh folder to all nodes
- Run these commands for all nodes…

```
systemctl stop firewalld
systemctl stop NetworkManager
systemctl disable firewalld
systemctl disable NetworkManager
vim /etc/selinux/config
SELINUX=disabled
```

- reboot
- Install httpd on master node

- Create local repo for centos7 in master node.

copy iso to /var/www/html/centos7.6/

vim /etc/yum.repos.d/local.repo

```
    [localrepo]
    name=centos7.6
    baseurl=http://192.168.1.10/centos7.6
    enabled=1
    gpgcheck=0
```

- Systemctl start httpd
- Systemctl enable httpd

2.Lustre installation options

- Building Lustre RPM files from sources
- Download RPMs directly install
- Configure a local repo

https:// wiki.whamcloud.com/display/PUB/Lustre+Releases

3. Configure a local repo

- On master node

```
cd /var/www/html

wget --mirror --convert-links --adjust-extension --page-requisites --no-parent https://downloads.whamcloud.com/public/lustre/lustre-2.12.2/el7.6.1810/server/

wget --mirror --convert-links --adjust-extension --page-requisites --no-parent https://downloads.whamcloud.com/public/lustre/lustre-2.12.2/el7.6.1810/patchless-ldiskfs-server/

wget --mirror --convert-links --adjust-extension --page-requisites --no-parent https://downloads.whamcloud.com/public/e2fsprogs/1.45.2.wc1/el7/

wget --mirror --convert-links --adjust-extension --page-requisites --no-parent https://downloads.whamcloud.com/public/lustre/lustre-2.12.2/el7.6.1810/client/
```

- vim /etc/yum.repos.d/lustre.repo [do this on all nodes]

```
[lustre-server]

name=lustre-server

baseurl=http://192.168.1.10/downloads.whamcloud.com/public/lustre/lustre-2.12.2/el7.6.1810/patchless-ldiskfs-server

enabled=1

gpgcheck=0


[lustre-server]

name=lustre-server

baseurl=http://192.168.1.10/downloads.whamcloud.com/public/lustre/lustre-2.12.2/el7.6.1810/server

enabled=1

gpgcheck=0
```

```
[lustre-client]
name=lustre-client
baseurl=http://192.168.1.10/downloads.whamcloud.com/public/lustre/lustre-2.12.2/el7.6.1810/client
enabled=1
gpgcheck=0

[lustre-e2fsporgs]
name=lustre-e2fsprogs
baseurl=http://192.168.1.10/downloads.whamcloud.com/public/e2fsprogs/1.45.2.wc1/el7
enabled=1
gpgcheck=0
```

- Verify by -- yum repolist

LNet Configuration

- cat /etc/modprobe.d/lustre.conf (Any entry)

```
options lnet networks=o2ib0

options lnet networks=tcp0(enp0s3)

options lnet networks=tcp0(bond0)
```

Or

```
lnetctl lnet configure

lnetctl net add --net tcp0 --if enp0s3


lnetctl global show

lnetctl net del --net tcp0
```

http://wiki.lustre.org/LNET_Selftest

Lustre packages

- Lustre Server Packages

e2fsprogs-xx.xx.rpm

lustre-2.12.2-1.el7.x86_64.rpm

kmod-lustre-2.12.2-1.el7.x86_64.rpm

kmod-lustre-osd-ldiskfs-2.12.2-1.el7.x86_64.rpm

lustre-osd-ldiskfs-mount-2.12.2-1.el7.x86_64.rpm

lustre-resource-agents-2.12.2-1.el7.x86_64.rpm

-------kernel-3.10.0-957.10.1.el7_lustre.x86_64.rpm

COUNT..

- Lustre Clients Packages

kmod-lustre-client

lustre-client

e2fsprogs

------lustre-client-dkms

The version of the kernel running on a Lustre client must be the same as the version of the

kmod-lustre-client-ver package being installed, unless the DKMS package is installed

- Server Pkg installation

yum install e2fsprogs kmod-lustre kmod-lustre-osd-ldiskfs lustre-osd-ldiskfs-mount lustre lustre-resource-agents

- Client Pkg installation

yum install e2fsprogs kmod-lustre-client lustre-client

COUNT…

- ON OSS and MDS servers ((((((firsttime it may create issue with mounting))))))

systemctl start lustre

systemctl enable lustre

- ON ALL Node

systemctl start lnet

systemctl enable lnet

Configuring Lustre

- Configuring mds node

Create a combined MGS and MDT File System on a block device

```
mkfs.lustre --fsname=fsname --mgs --mdt --index=0 /dev/block_device
```

```
e.g. mkfs.lustre --fsname=lustre --mgs --mdt --index=0 /dev/sdb
```

- Mount the combined MGS and MDT file system on the block device

```
mount -t lustre /dev/block_device /mnt_point
```

```
e.g. mount -t lustre /dev/sdb /mdt0
```

- Configure Lustre OSS servers

Create an OST File System on a block device

```
mkfs.lustre --fsname=fsname --mgsnode=<node_ip@tcp0> --ost -index=OST_index /dev/block_device
```

```
e.g mkfs.lustre --fsname=lustre --mgsnode=mds1@tcp0 --ost --index=0 /dev/sdb
```

COUNT..

Mount the OST file system on the block device

```
mount -t lustre /dev/block_device /mnt_point
```

e.g. mount -t lustre /dev/sdb /ost0

Lustre File system on the client node

Mount the Lustre File system on the client node

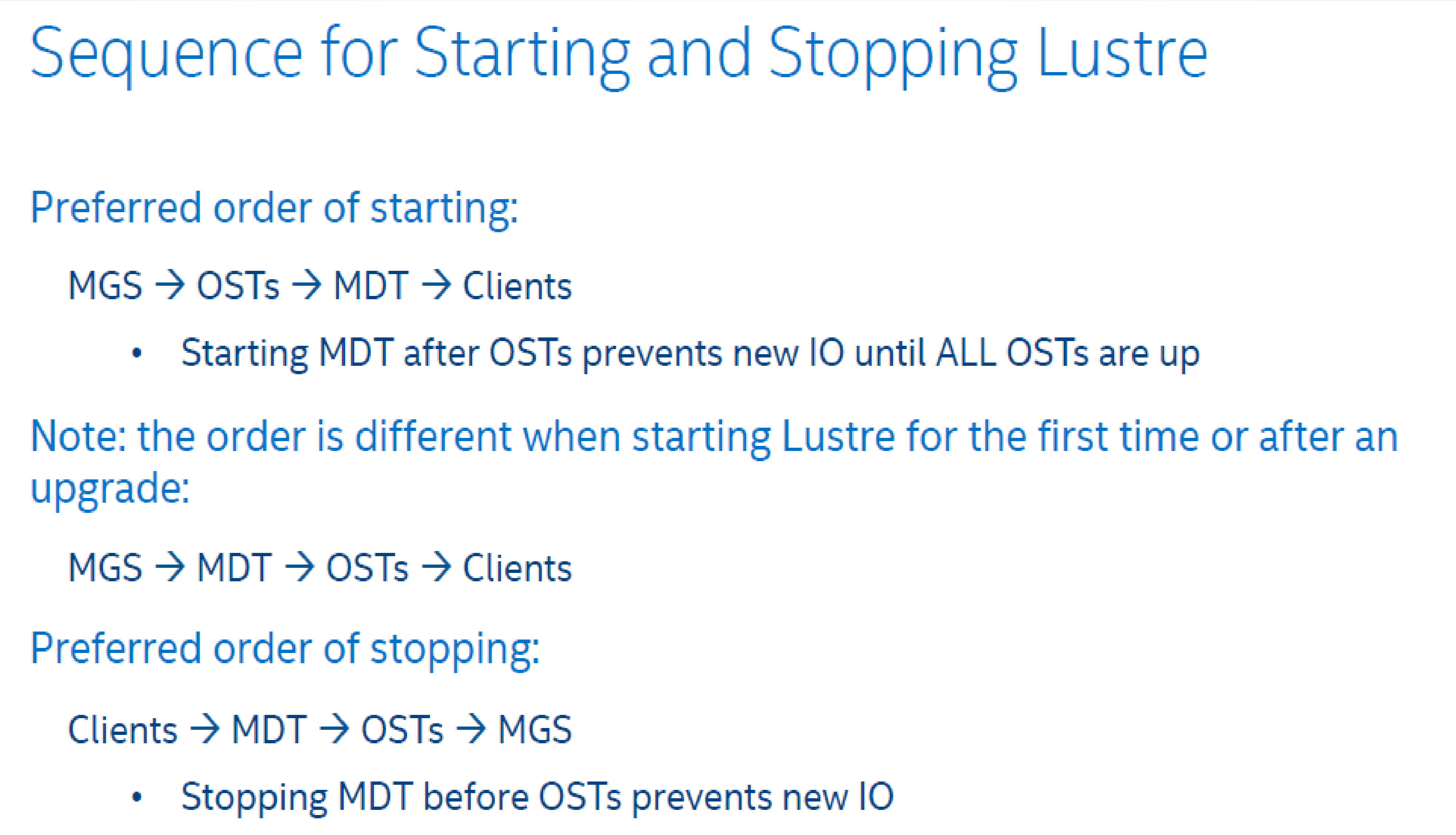
```
mount -t lustre MGS_node:/fsname /mount_point
```

e.g. mount -t lustre mds1:/lustre /lustre

Verifying the Lustre Cluster

o Log in to Lustre client
o List all MDT devices in the Lustre cluster – “lfs mdts”
o List all OST devices in the Lustre cluster – “lfs osts”
o Show space usage per MDT and OSTs -"lfs df -h"
o Show all files residing on Lustre file system on /lustre - "ls -lsah /lustre
o Run dd command "dd if=/dev/zero of=/lustre/zero1.dat bs=400M count=2" to write some 800MB of data on Lustre FS
o Show the newly created file Lustre file system on /mnt - "ls -lsah /mnt”
o Show the usage of Lustre file system -"lfs df -h"
o This will show percentage and amount of data written on each OST in Lustre file system

Starting / Stopping Sequence



Sequence for Starting and Stopping Lustre

Preferred order of starting:

MGS → OSTs → MDT → Clients

- Starting MDT after OSTs prevents new IO until ALL OSTs are up

Note: the order is different when starting Lustre for the first time or after an upgrade:

MGS → MDT → OSTs → Clients

Preferred order of stopping:

Clients → MDT → OSTs → MGS

- Stopping MDT before OSTs prevents new IO

example

- On each client…

```
umount -a -t lustre
```

```
lfs df -h
```

- On the MDS node(s), use the umount command:

```
umount -a -t lustre
```

- Unmount all the OSTs

On each OSS node, use the umount command:

```
umount -a -t lustre
```

- On the MGS use the umount command:

```
umount -a -t lustre
```

- Shutdown storage box
- Shutdown switch

Few Useful commands hands-on

- modprobe lustre
- modprobe lnet
- lsmod | grep lustre
- lctl network up
- lctl get_param version
- lctl list_nids
- lfs df –h
- lfs check servers
- llstat --i 2 /proc/fs/lustre/mds/MDS/mdt/stats or LMT (like ganglia)

Lustre Best Practices

- Get the latest Lustre sources from www.whamcloud.com. You can find Lustre-related information on www.lustre.org as well.
- Ensure that you use the same version of Linux Operating System and Kernel as mentioned on WhamCloud for the Lustre version you plan to deploy in your environment.
- All Lustre servers and Lustre clients must be running the same Operating System and Kernel version
- While configuring Lustre cluster and file system, ensure all Lustre servers and Lustre clients can access each other on the network with proper network configuration, or update the /etc/hosts file.
- While configuring Lustre cluster, ensure that SELINUX is disabled
- Always install and configure a separate MGS / MDS server and do not merge it with OSS server. This will prevent any latency during Read / Write I/Os on Lustre file system.

Count…

- For long term data integrity, Lustre filesystem devices (OST/MDT) must be RAID protected.
- Higher availability of Lustre cluster, configure high availability using Linux clustering for each Lustre Server – OSS, MDS / MGS.
- Subscribe to Lustre mailing list hpdd-discuss@lists.01.org