Data Management and File Systems

Manu Shantharam

mshantharam@sdsc.edu

CIML Summer Institute, June 22, 2021

An ML case study

Automated classification of In Situ Ichthyoplankton Imaging System (ISIIS) images using Convolutional Neural Nets on parallel computing infrastructure

Objective: effective utilization Comet's GPU compute resources for the automated classification of ISIIS images using Convolutional Neural Nets

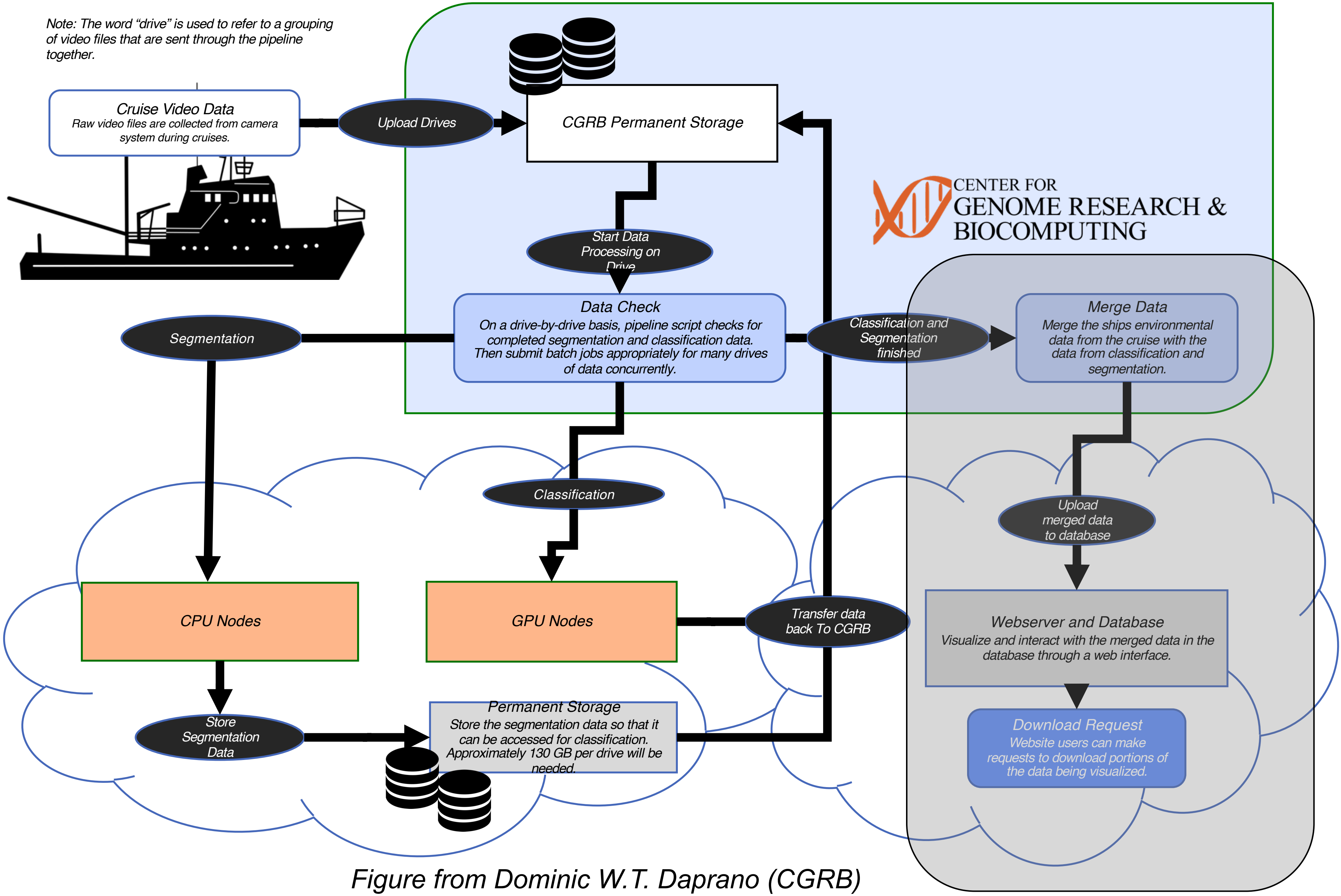


Figure from Dominic W.T. Daprano (CGRB)

Workflow

- Copy (scp) the source code to the $HOME
- Login (ssh) to the supercomputer and compile the source code
- file_processing.sh
  - rsync (pull) *.tar.gz files from a drive (in CGRB) to project space
  - Split the tar.gz files into groups to fit into the local SSD
- Start_processing_parallel.sh
  - Copy the required tar.gz files from project space to local SSD
  - Prepare_classification.sh
  - Run_classification.sh
  - Zip the output files and rsync (push) it to CGRB

Data Management

- Managing data before / after computation
  - data collection, copy, sync

- Managing data during computation
  - staging, generation

Why Data Management?

Seriously, why Data Management?

- Various data sources
  - sensors
  - undersea expedition cameras
  - satellite images
  - simulations
- Different storage mechanisms
  - external hard drives / remote machine
  - hard drives on a local machine
  - on a supercomputer
    - local, home, parallel file system…

Data Management Tools

- Globus: a tool that provides fast, secure and reliable file transfer and data sharing mechanisms.
  - web interface: https://www.globus.org/
  - globus connect personal: https://www.globus.org/globus-connect-personal
  - command line interface: https://docs.globus.org/cli/
  - globus-url-copy: https://portal.xsede.org/web/xup/data-management#globusurlcopy
- Others:
  - scp, rsync, sftp..

Why File Systems

- Place to store data / files – manage data
- Computations involving 1000s of files – temporary files during genome sequencing, images…
- Large shared files due to checkpointing – weather forecasting, long running machine learning jobs

Expanse’s tiered storage

- Node local NVMe drives for workloads that don’t need to share data files across nodes
- Lustre filesystem for I/O workloads that require high-bandwidth and large capacity shared storage
- Network Files System (NFS) cluster for user home directory storage
- Ceph Object Storage for short-term archival storage and staging data transfers to cloud-based storage (coming soon)

Why Various File Systems

- Performance
- Shared access across nodes
- Backup / long-term
- Quota

Expanse File Systems: $HOME

- Location of the home directory – when you login to Comet

- Network File System (NFS) storage
  - Typically used to store source codes, important files…
  - Storage limit around 100 GB

- Backup / long-term

Expanse File Systems: Lustre scratch

- Location: /expanse/lustre/scratch/$USER/temp_project

- Lustre File System (LFS) performance storage
  - Typically used to store input / output data, large files…
  - Allows distributed access
  - Storage limit around 1TB
  - Purged after 90 days (creation)

- No Backup

Expanse File Systems: Lustre projects

- Location: /expanse/lustre/projects/…

- Lustre File System (LFS) performance storage
  - Typically used to store input / output data, large files…
  - Project specific data
  - Allows distributed access
  - Storage limit around 2.5 PB

- No Backup

Expanse File Systems: Node Local Storage

- Location: /scratch/$USER/job_$SLURM_JOB_ID…

- Node local SSD storage
  - Typically used to store large number of files…
  - Fast node-local access
  - Storage limits: compute, shared: 1 TB; gpu, gpu-shared: 1.6 TB; large-shared: 3.2 TB
  - Only accessible from a compute node
  - purged after the job

Expanse File Systems

Path	Purpose	User Access Limits	Lifetime
$HOME	NFS storage; Source code, important files	100 GB	Backed-up
/expanse/lustre/scratch/$USER/temp_project	Parallel Lustre FS; temp storage for distributed access	Need-based	No backup
/expanse/lustre/projects/	Parallel Lustre FS; project storage	Need-based	No backup
/scratch/$USER/job_$SLURM_JOB_ID	Local SSD on batch job node fast per-node access	More than 1 TB	Purged after job ends

File Systems Guidelines (from Comet)

```
[2] Filesystems:

     (a) Lustre scratch filesystem : /oasis/scratch/comet/$USER/temp_project
         (Preferred: Scalable large block I/O)
            *** Meant for storing data required for active simulations
            *** Not backed up and should not be used for storing data long term
            *** Periodically clear old data not required for active simulations

     (b) Compute/GPU node local SSD storage: /scratch/$USER/$SLURM_JOBID
         (Meta-data intensive jobs, high IOPs)

     (c) Lustre projects filesystem: /oasis/projects/nsf

     (d) /home/$USER : Only for source files, libraries, binaries.
         *Do not* use for I/O intensive jobs.
```

Order of Magnitude Guide

Storage	file/directory	file sizes	BW
Local HDD	1000s	GB	100 MB/s
Local SSD	1000s	GB	500 MB/s
RAM FS	10000s	GB	GB/s
NFS	100s	GB	100 MB/s
Lustre	100s	TB	100 GB/s

Local file systems are good for small and temporary files (low latency, modest bandwidth)

Parallel file systems very convenient for sharing data between the nodes (high latency, high bandwidth)

Application Focus

Storage choices should be driven by application need, not just what's available.

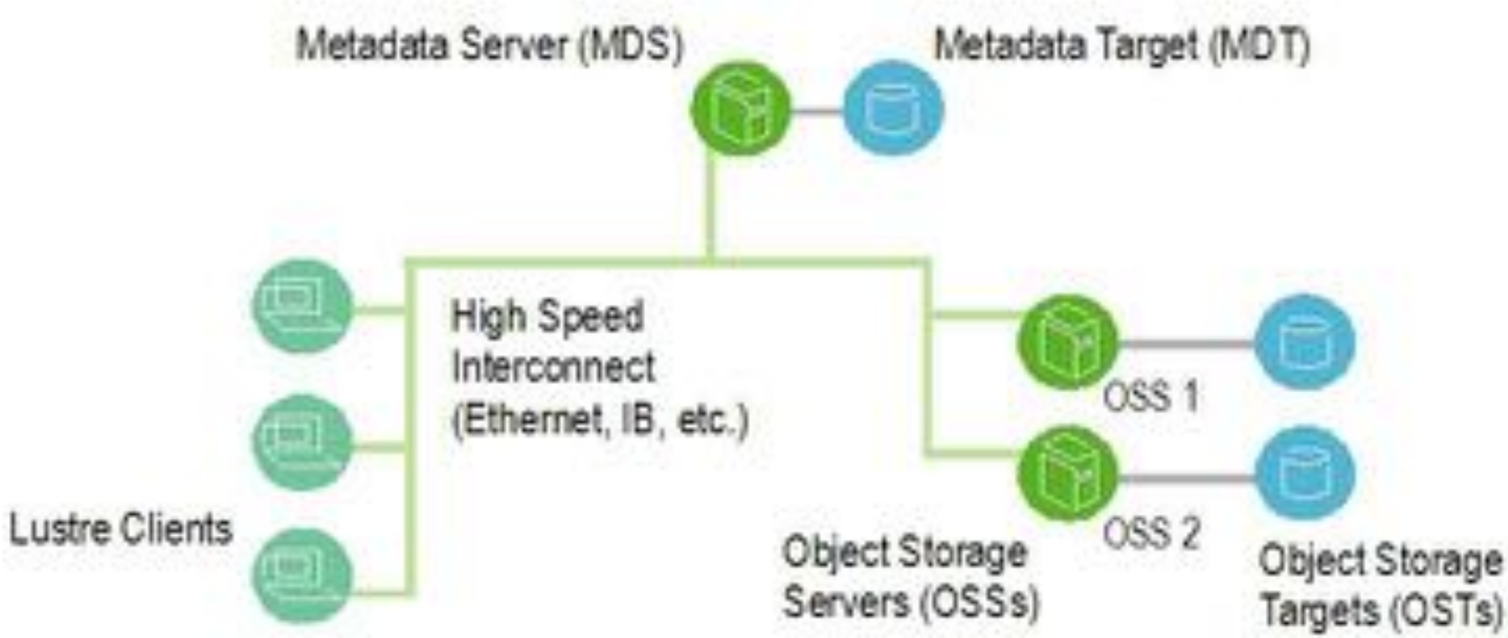
Writing a few small files to an NFS server is fine…
writing 1000's simultaneously will wipe out the server.

But, applications need to adapt as they scale.

Application Focus

d to

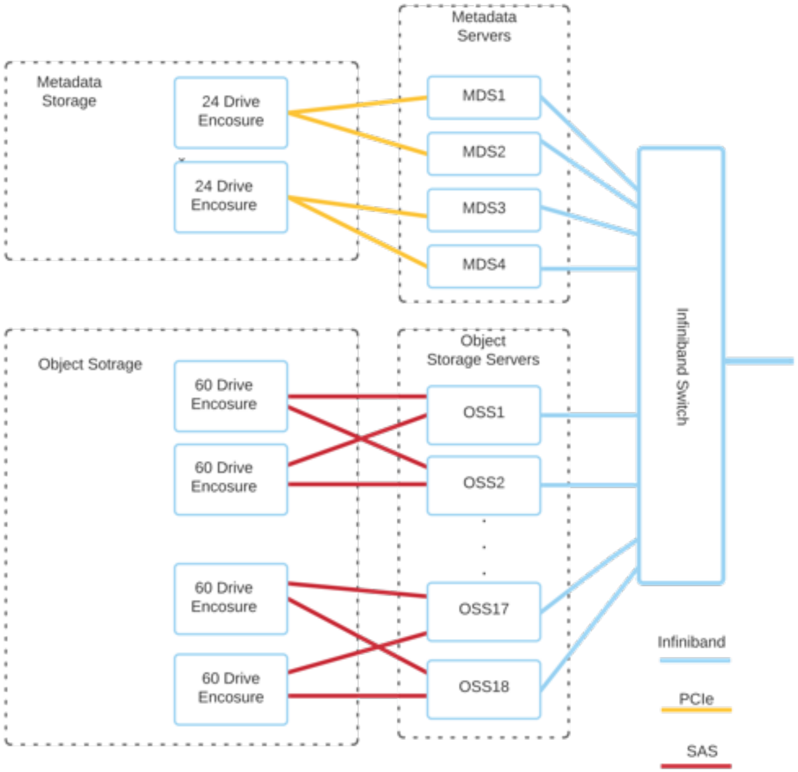
Lustre File System



Ref: Cornell Virtual Workshop

Expanse Lustre File System Architecture

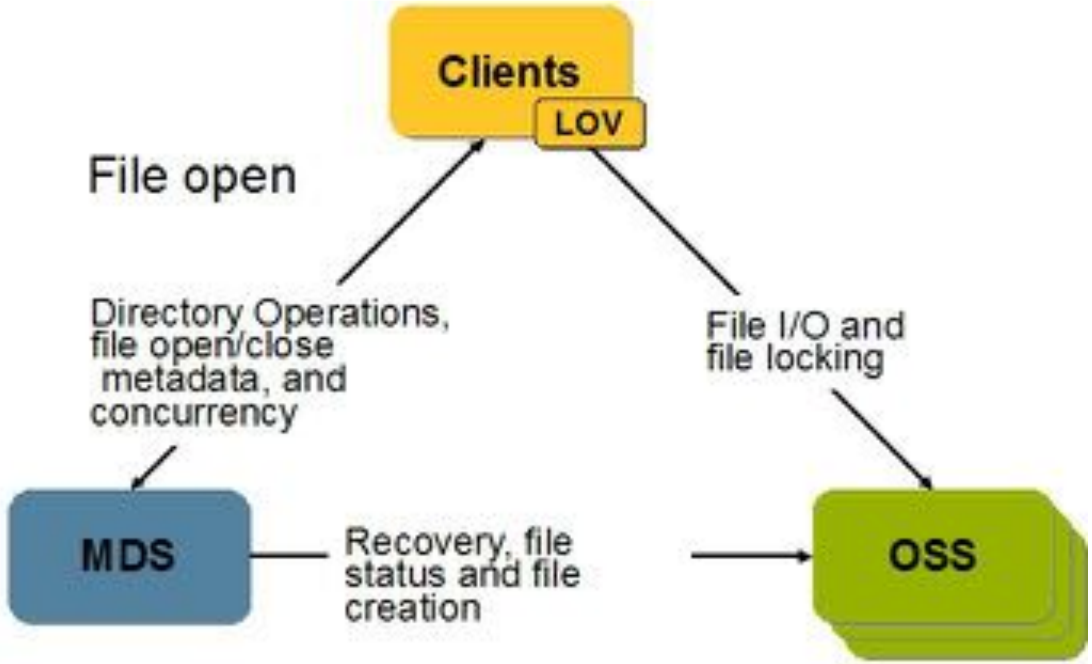
- 12 Peta Bytes of RAW capacity, approx. 11 PB formatted
- File Capacity of approx. 3 billion files.
- 140 GB/s Filesystem Bandwidth
- 200K IOPS
- Data on MDT (DoM) for small file performance



4 Lustre MDS	
Processor	2 X AMD Epyc 7302 (16 Cores)
Memory	512 GB (16 X 32GB DDR4 3200)
MDT Drives	24 X 3.8 TB NVMe per pair
Interconnect	InfiniBand HDR 200
System Drives	2 X 240 GB Intel SSDs

18 Lustre OSS	
Processor	1 AMD Epyc 7402 (24 Cores)
Memory	512 GB (16 X 32 GB DDR4 3200)
JBODS	2 Cross Connected 60 Bay JBODS
OSS Drives	120 X 14 TB 7200 SAS Drives
Interconnect	InfiniBand HDR 200
System Drives	2 X 240 GB Intel SSDs

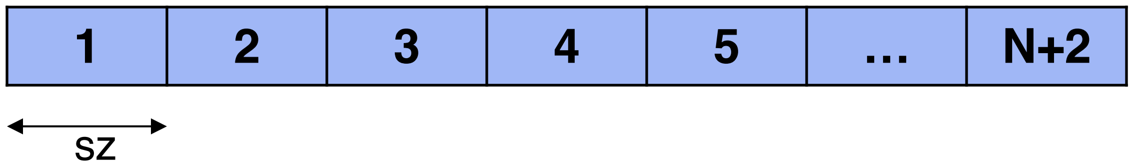
LFS Interactions



Ref: Cornell Virtual Workshop

File View

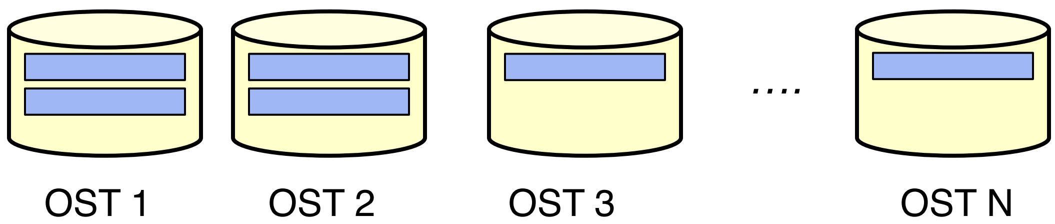
Logical view of a file with N+2 segments



Stripe count = N

Stripe size = sz

Physical view of the file across OSTs



Why is striping useful?

- a way to store a large file
- file can be accessed in parallel, increasing the bandwidth

LFS Commands

lfs help – lists all options

lfs osts – lists all the OSTs

lfs mdts – lists all the MDTs

lfs getstripe – retrieves the striping information of a file / directory

lfs setstripe – sets striping information of a file / directory

Quiz: application requirement

My application needs to:

Write a checkpoint dump from memory from a large parallel simulation.

I should consider:

A parallel file system and a binary file format like HDF5.

Quiz: application requirement

My application needs to:

write and read 1000s of small files local to each process, store all the files across all the processes

I should consider:

a combination of local SSDs and Lustre!

Thank You!

Questions: mshantharam@sdsc.edu

LFS Commands: getstripe

```
-bash-4.1$ lfs getstripe testout
testout
lmm_stripe_count:   1
lmm_stripe_size:    1048576
lmm_pattern:        1
lmm_layout_gen:     0
lmm_stripe_offset:  43
        obdidx           objid          objid          group
            43         8979631       0x8904af              0

-bash-4.1$ lfs getstripe --stripe-count testout
1
-bash-4.1$ lfs getstripe --stripe-size testout
1048576
```

LFS Commands: setstripe

lfs setstripe -c 16 testout

```
-bash-4.1$ lfs getstripe testout
testout
lmm_stripe_count:   16
lmm_stripe_size:    1048576
lmm_pattern:        1
lmm_layout_gen:     0
lmm_stripe_offset:  89
        obdidx          objid           objid           group
            89        9202813        0x8c6c7d              0
            45        9819070        0x95d3be              0
                  ................................
```

LFS Commands: setstripe

```
bash-4.1$ lfs setstripe -c -1 test1
bash-4.1$ lfs getstripe test1
test1
lmm_stripe_count:   96
lmm_stripe_size:    1048576
lmm_pattern:        1
lmm_layout_gen:     0
lmm_stripe_offset:  65
        obdidx       objid        objid        group
            65     9738084     0x949764            0
            41     9153699     0x8baca3            0
                   ..............................
```

LFS Commands: setstripe

```
-bash-4.1$ mkdir dir
-bash-4.1$ lfs setstripe -c 4 dir
-bash-4.1$ vi dir/test
-bash-4.1$ lfs getstripe dir/test
dir/test
lmm_stripe_count:   4
lmm_stripe_size:    1048576
lmm_pattern:        1
lmm_layout_gen:     0
lmm_stripe_offset:  43
        obdidx           objid           objid           group
            43         8979901        0x8905bd               0
            25        10609192        0xa1e228               0
```

LFS Usage Guidelines

- Avoid certain operations
  - ls –l, ls with color, frequent file opens/closes
  - find, du, wildcards (ls *.out)
  - Why??
  - Try /bin/ls -U instead of ls -l
- Select appropriate stripe count / size
  - Best case selection is complicated
- Do not store too many files in one directory