Data Management and File Systems on Expanse

Mahidhar Tatineni

SDSC HPC/CI Training Series - Session I

Jan 27, 2022

Ref: Manu Shantharam's Expanse webinar

Overview

- Example Workflow
- Data Management
  - What and why?
  - Tools / frameworks
- File Systems - Expanse
  - Pros and cons of different file systems
  - Lustre File System
- Example scenarios
- Summary

Example Workflow

- Copy (scp) the source code to $HOME
- Login (ssh) to the supercomputer and compile the source code
- Preprocessing step
  - Use Globus OR rsync (pull) *.tar.gz files from a client to project space
  - Split the tar.gz files into groups to fit into the local SSD
- Main processing steps
  - 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) to external storage

What is Data Management?

- Managing data before / after computation
  - data collection, copy, sync
- Managing data during computation
  - staging, generation
- Data provenance and integrity

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…
- Performance and scalability of applications

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/
- Useful resource related to data management
  - https://portal.xsede.org/web/xup/data-management

Data Management Tools

- scp, sftp: good for small number of files, small files

```
SCP(1)

NAME
     scp – secure copy (remote file copy program)

SYNOPSIS
     scp [-346BCpqrTv] [-c cipher] [-F ssh_config] [-i
         [-S program] source ... target

DESCRIPTION
     scp copies files between hosts on a network.  It
     the same security as ssh(1).  scp will ask for pa

     The source and target may be specified as a local
     a URI in the form scp://[user@]host[:port][/path]
```

```
SFTP(1)

NAME
     sftp – secure file transfer program

SYNOPSIS
     sftp [-46aCfpqrv] [-B buffer_size] [
          [-J destination] [-l limit] [-o
          destination

DESCRIPTION
     sftp is a file transfer program, sim
     also use many features of ssh, such
```

- rsync

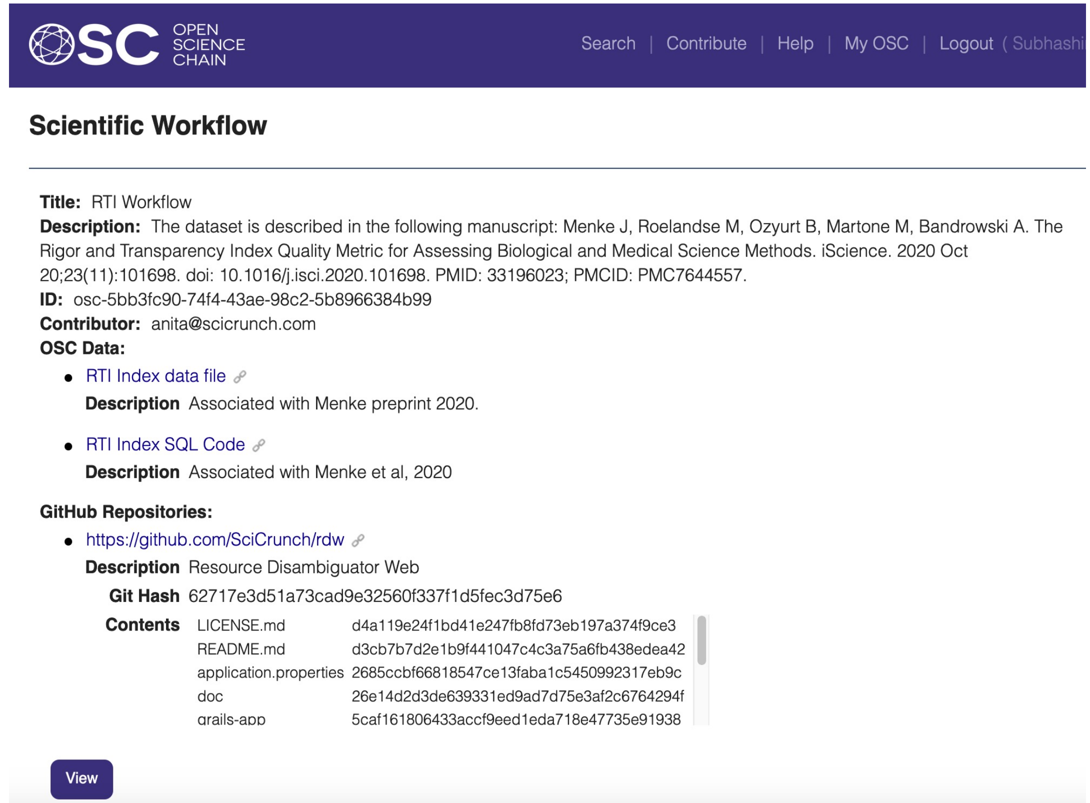
```
rsync(1)

NAME
       rsync - a fast, versatile, remote (and local) file-copying tool
```

Open Science Chain

https://www.opensciencechain.org

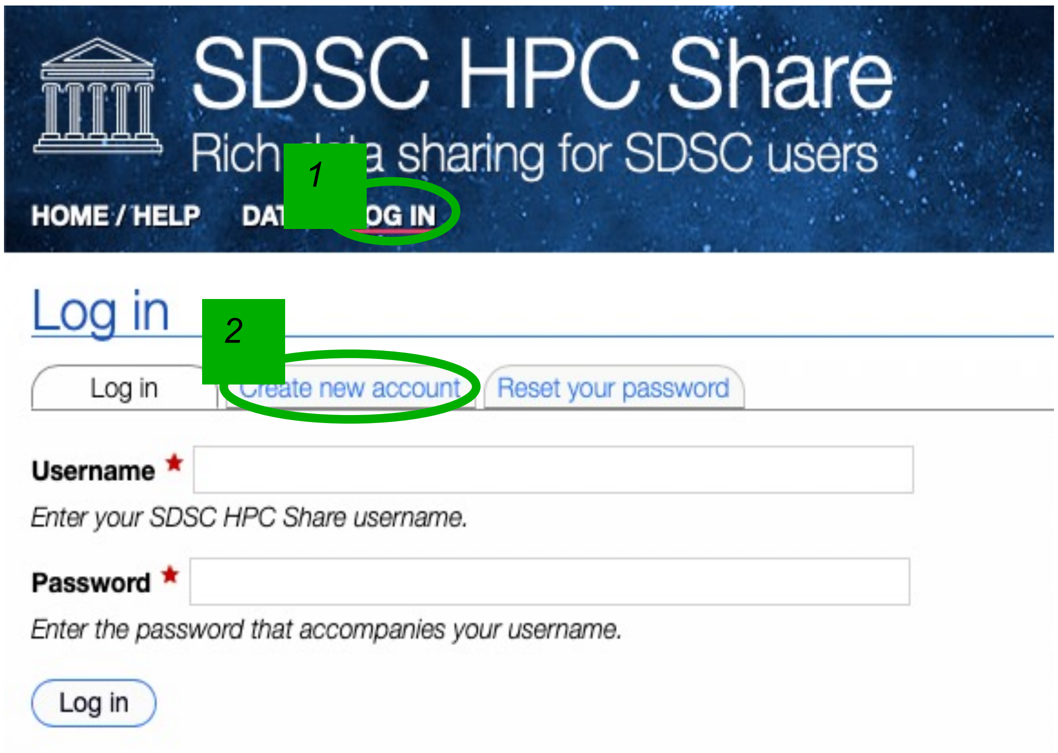
- Open Science Chain utilizes blockchain technologies to securely provide traceability of research artifacts
- Cryptographic hash of datasets and the metadata information stored in consortium blockchain
- Interact with blockchain using easy to use web interface (OSC portal) or Python API (e.g., from Expanse).
- Ability to search, verify and validate the research artifacts that are contributed by other researchers.
- Create a detailed workflow linking multiple sources of data and computational/analysis code (e.g., GitHub)
- Track provenance of metadata as the dataset evolves/changes



Contact: info@opensciencechain.org
sivagnan@sdsc.edu

HPCShare SeedMeLab

Sign up at hpcshare.sdsc.edu & view usage videos



Benefits

- *Collaborate*
  Share data and comments on any file or folder.
- *Upload/download (<2 GB/per file) from Expanse cluster or elsewhere to Web*
  Via command line, API or web browser.
- *Annotate data with rich text*
  Such as arbitrary context, metadata, equations, etc.
- *Create preview visualizations of small tabular data*
  CSV and JSON formats.

HPCShare is powered by SeedMeLab – an open source data management system


FILE SYSTEMS
FILES & FILE SYSTEMS

Why File Systems

- Place to store data / files – manage data

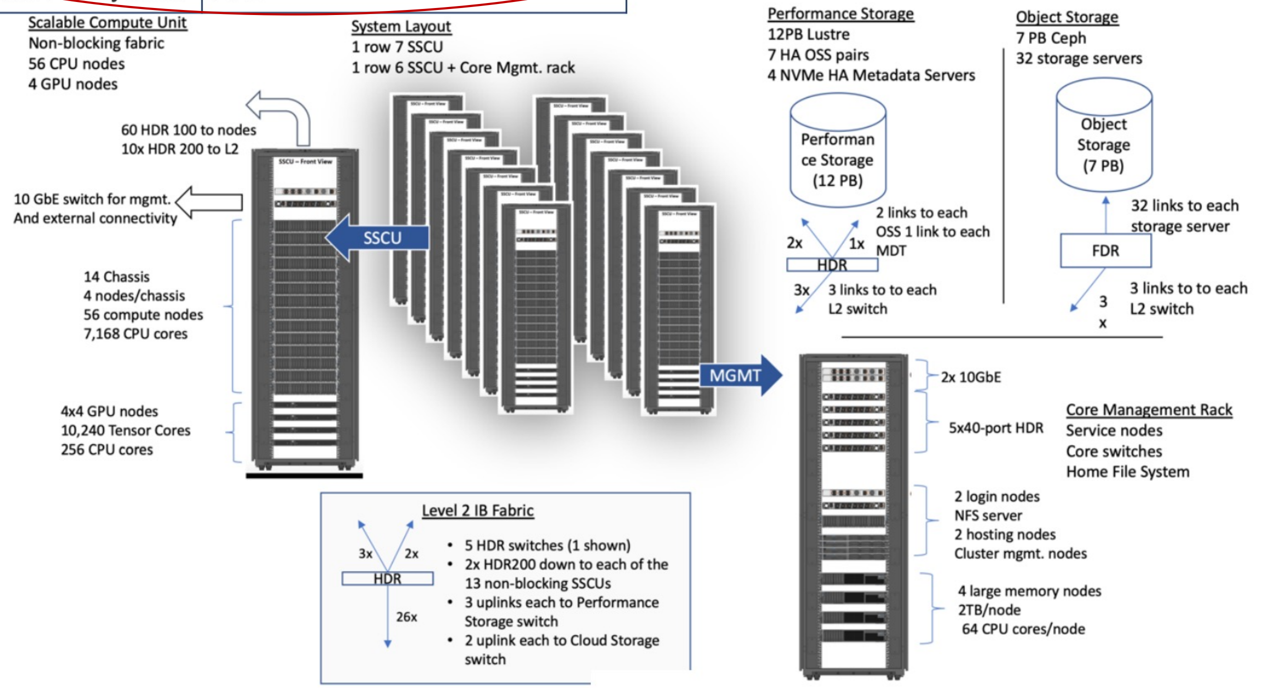
- Computations involving 1000s of files – temporary files during genome sequencing, images…

- Large shared files due to checkpointing – weather forecasting, long running machine learning jobs

Expanse System

System Component	Configuration
AMD EPYC (Rome) 7742 Compute Nodes	
Node count	728
Clock speed	2.25 GHz
Cores/node	128
Total # cores	93,184
DRAM/node	256 GB
NVMe/node	1 TB
NVIDIA V100 GPU Nodes	
Node count	52
Total # GPUs	208
GPUs/node	4
GPU Type	V100 SMX2
Memory/GPU	32 GB
CPU cores; DRAM; clock (per node)	40; 384 GB; 2.5 GHz;
CPU	6248 Xeon
NVMe/node	1.6TB
Large Memory Nodes	
Number of nodes	4
Memory per node	2 TB
CPUs	2x AMD 7742/node;

Storage	
Lustre file system	12 PB (split between scratch & allocable projects)
Ceph file system	7 PB Coming soon
Home File system	1 PB



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

Applications and Performance

```
[manu1729@exp-2-09 src]$ mpirun -np 4 ./ior --posix.od
IOR-3.4.0+dev: MPI Coordinated Test of Parallel I/O
Began               : Thu Oct 21 05:47:11 2021
Command line        : ./ior --posix.odirect -F -b 5m -
Machine             : Linux exp-2-09
TestID              : 0
StartTime           : Thu Oct 21 05:47:11 2021
Path                : testFile.00000000
FS                  : 10842.9 TiB   Used FS: 14.0%   I

Options:
api                 : POSIX
apiVersion          :
test filename       : testFile
access              : file-per-process
type                : independent
segments            : 1
ordering in a file  : sequential
ordering inter file : no tasks offsets
nodes               : 1
tasks               : 4
clients per node    : 4
repetitions         : 1
xfersize            : 131072 bytes
blocksize           : 5 MiB
aggregate filesize  : 20 MiB

Results:

access    bw(MiB/s)  IOPS       Latency(s)  block(KiB)
------    ---------  ----       ----------  ----------
write     3.39       27.09      0.135989    5120
read      1249.61    10220      0.000339    5120
```

```
[manu1729@exp-2-09 job_6589251]$ mpirun -np 4 ./ior --posix.o
IOR-3.4.0+dev: MPI Coordinated Test of Parallel I/O
Began               : Thu Oct 21 05:48:56 2021
Command line        : ./ior --posix.odirect -F -b 5m -t 128k
Machine             : Linux exp-2-09
TestID              : 0
StartTime           : Thu Oct 21 05:48:56 2021
Path                : testFile.00000000
FS                  : 915.9 GiB   Used FS: 0.0%   Inodes: 58.

Options:
api                 : POSIX
apiVersion          :
test filename       : testFile
access              : file-per-process
type                : independent
segments            : 1
ordering in a file  : sequential
ordering inter file : no tasks offsets
nodes               : 1
tasks               : 4
clients per node    : 4
repetitions         : 1
xfersize            : 131072 bytes
blocksize           : 5 MiB
aggregate filesize  : 20 MiB

Results:

access    bw(MiB/s)  IOPS       Latency(s)  block(KiB) xfer(K
------    ---------  ----       ----------  ---------- -------
write     1319.96    10606      0.000367    5120       128.00
read      752.81     6024       0.000631    5120       128.00
```

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.

Expanse File Systems: $HOME

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

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

- Limited number of snapshots (~1-2 months) available. Make offsite copies of anything critical.

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 NVMe 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 ends

Expanse File Systems Summary

Path	Purpose	User Access Limits	Lifetime
$HOME	NFS storage; Source code, important files	100 GB	Limited number of snapshots (~1-2 month)
/expanse/lustre/scratch/$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 NVMe on batch job node fast per-node access	More than 1 TB	Purged after job ends

File Systems Guidelines

(a) Lustre scratch space: /expanse/lustre/scratch/$USER/temp_project

- *Meant for storing data required for active simulations*
- *Not backed up and should not be used for storing data long term*
- *90-day purge policy (based on *create* date)*
- *Large block scalable IO*

(b) Compute/GPU node local NVMe storage: /scratch/$USER/job_$SLURM_JOBID

- *Meta-data intensive jobs, high IOPs*
- *File per core type I/O with continuous writes*
- *Purged at end of job*

(c) Lustre projects space: /expanse/lustre/projects/GROUP/$USER

- *Not backed up*
- *Meant for storing data needed for duration of project - an example is reference datasets*

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

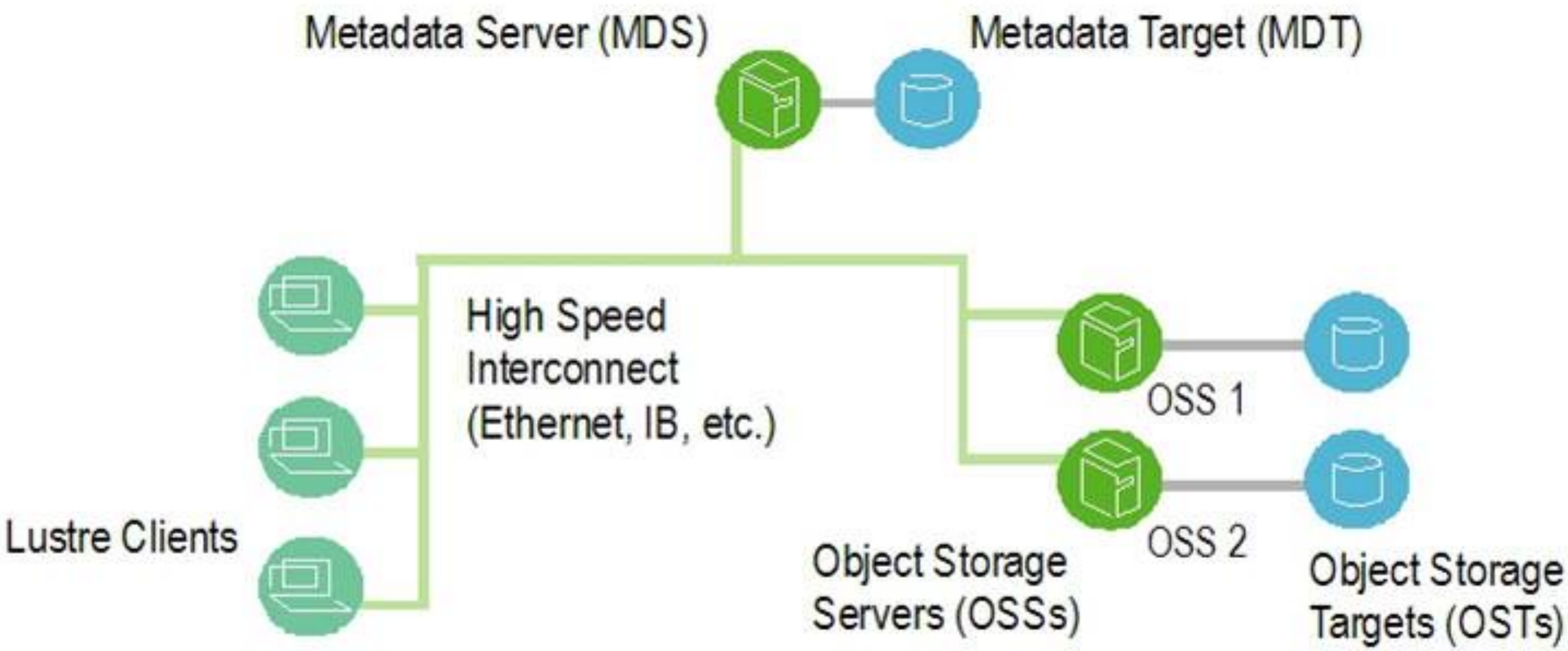
Order of Magnitude Guide

Storage	file/directory	file sizes	BW
Local HDD	1000s	GB	100 MB/s
Local NVMe	1000s	GB	1000 MB/s
RAM FS	10000s	GB	Several 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)

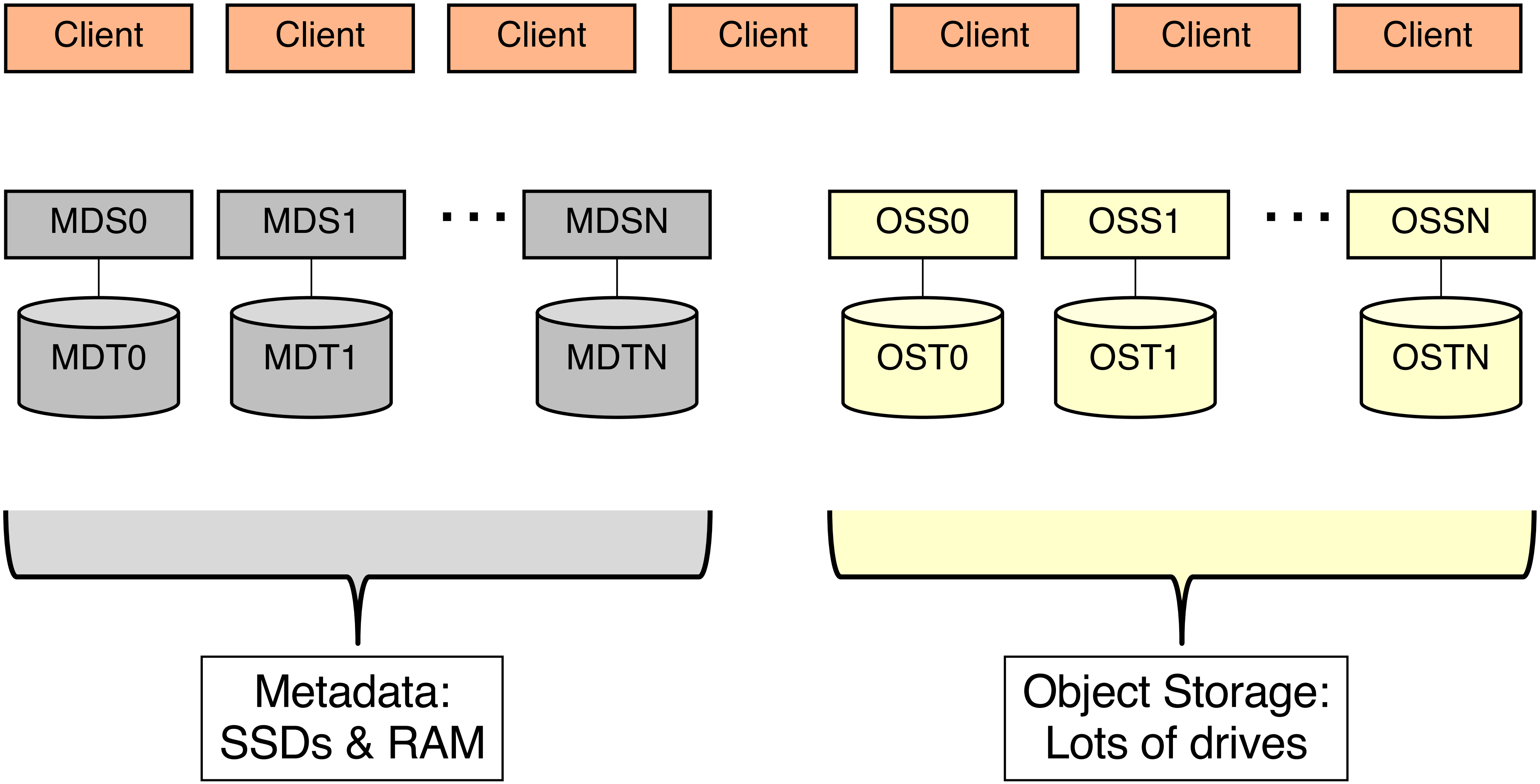
Parallel file systems very convenient for sharing data between the nodes (high latency, high bandwidth)

Lustre File System



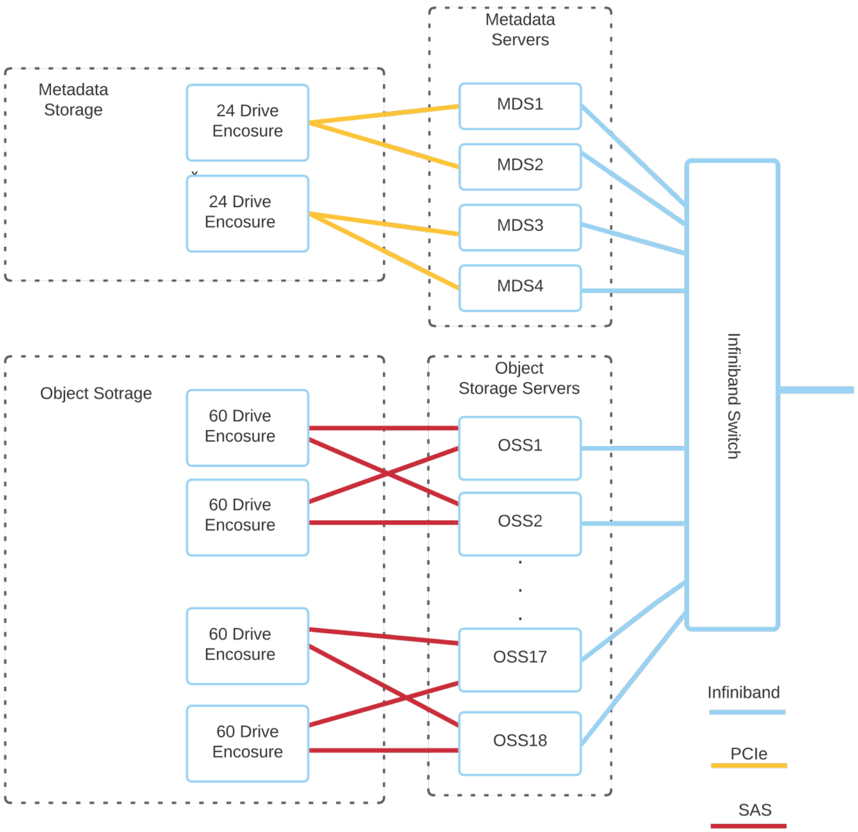
Ref: Cornell Virtual Workshop

A Typical LFS



Expanse Lustre File System Architecture

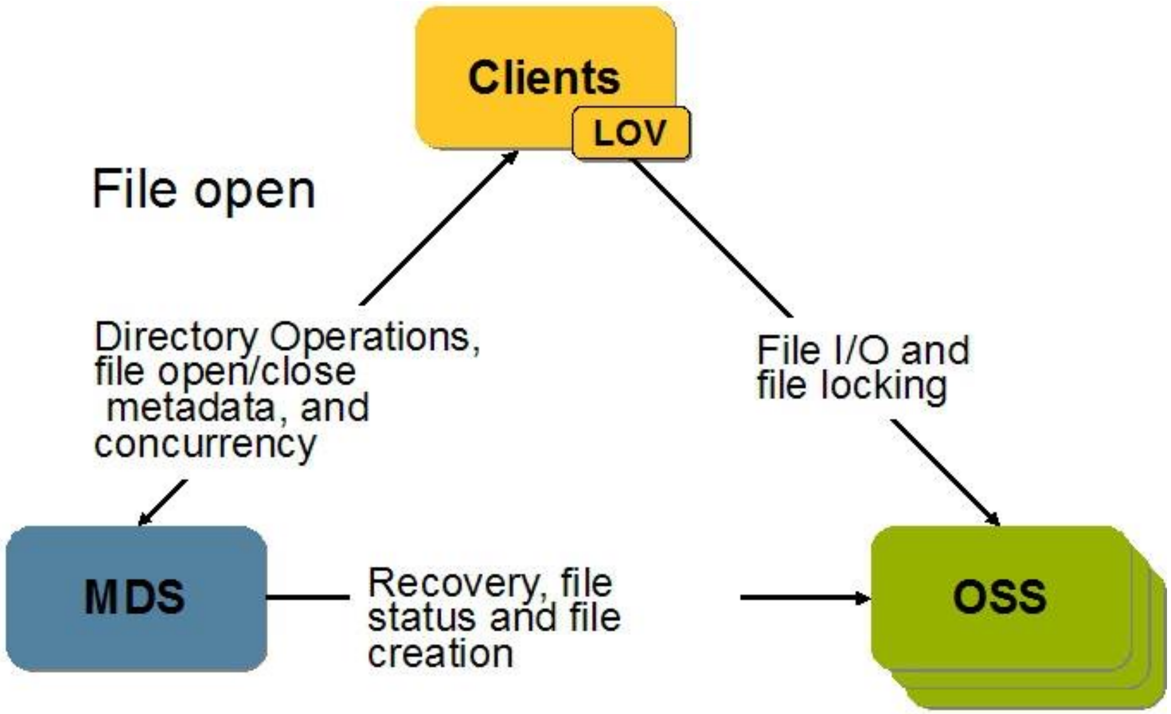
- 12 Peta Bytes of RAW capacity, approx.
  11 PB formatted
- File Capacity of approx. 3 billion files.
- 140 GB/s Filesystem Bandwidth
- 200K IOPS
- Data on MDT (DoM) for small file performance



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

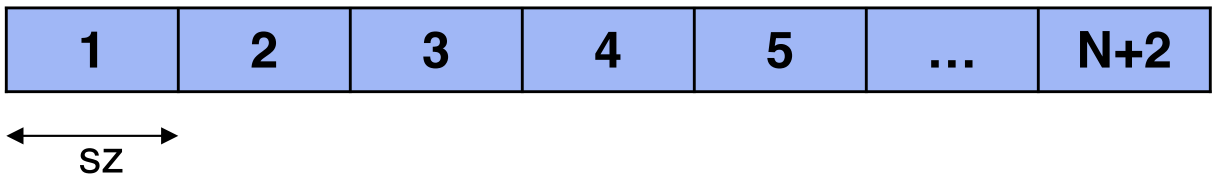
LFS Interactions



Ref: *Cornell Virtual Workshop*

File View

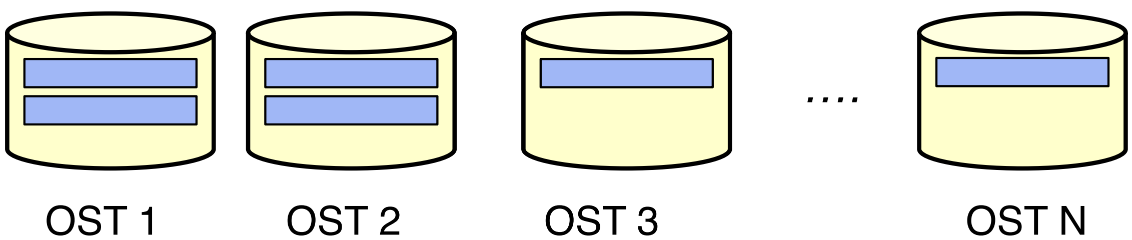
Logical view of a file with N+2 segments



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

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

Demo

IOR Benchmark

- IOR (Interleaved Or Random) developed at LLNL to benchmark/test parallel filesystems.
- Current version is very versatile (beyond what the name suggests).
- Test aggregate I/O rates using several I/O options including **POSIX, MPIIO, HDF5, and NCMPI.**
- Control several aspects of I/O to help mimic real applications:
  - Overall I/O Size
  - Transfer size
  - File access mode – single or file/task
  - Random/Sequential I/O
  - Lustre specific options
  - GPFS hints

IOR Options

IOR -h gives you all the options. Some important ones are:

- -F : *write one file per task*
- *(without -F a single file is written)*
- -b : *blockSize – contiguous bytes to write per task*
- -t : *size of transfer in bytes (e.g. 8, 4k, 2m, 1g)*
- -w : *Only write a file (default is to write and read)*
- -r : *Only read an existing file*
- -i : *number of iterations*
- --posix.odirect: *uses O_DIRECT for POSIX, bypassing I/O buffers*

IOR Example Script
/cm/shared/examples/sdsc/localscratch

Localscratch-slurm.sb

```
#!/bin/bash
#SBATCH --job-name="localscratch"
#SBATCH --output="localscratch.%j.%N.out"
#SBATCH --partition=shared
#SBATCH --account=XYZ123
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
#SBATCH --export=ALL
#SBATCH -t 01:30:00


#Copy binary to SSD
#Can also copy inputs there if needed
cp IOR-mpiio.exe /scratch/$USER/job_$SLURM_JOBID


#Change to local scratch (SSD) and run IOR benchmark
cd /scratch/$USER/job_$SLURM_JOBID

#Run IO benchmark
module reset
module load gcc/10.2.0
module load openmpi/4.0.4
module load sdsc
ibrun -np 16 ./IOR-mpiio.exe -F -t 1m -b 1g -v -v >
IOR.out.$SLURM_JOBID


#Copy output file back
cp IOR.out.$SLURM_JOBID $SLURM_SUBMIT_DIR
```

Sample IOR Output

```
Summary:
        api                 = POSIX
        test filename       = testFile
        access              = file-per-process
        pattern             = segmented (1 segment)
        ordering in a file = sequential offsets
        ordering inter file= no tasks offsets
        clients             = 16 (16 per node)
        repetitions         = 1
        xfersize            = 1 MiB
        blocksize           = 1 GiB
        aggregate filesize = 16 GiB

Using Time Stamp 1635794780 (0x61803f5c) for Data Signature
Commencing write performance test.
Mon Nov  1 12:26:20 2021

access    bw(MiB/s)  block(KiB) xfer(KiB)  open(s)    wr/rd(s)   close(s) total(s)  iter
------    ---------  ---------- ---------  --------   --------   --------  --------  ----
write     4147       1048576    1024.00    0.002891   3.95       0.974847   3.95       0    XXCEL
[RANK 000] open for reading file testFile.00000000 XXCEL
Commencing read performance test.
Mon Nov  1 12:26:24 2021

read      3988       1048576    1024.00    0.002485   4.11       3.68       4.11       0    XXCEL
Operation  Max (MiB)  Min (MiB)  Mean (MiB)   Std Dev  Max (OPs)  Min (OPs)  Mean (OPs)   Std Dev  Mean (s)  Op grep #Tasks tPN
 reps  fPP reord reordoff reordrand seed segcnt blksiz xsize aggsize

---------  ---------  ---------  ----------   -------  ---------  ---------  ----------   -------  --------
write        4147.15    4147.15     4147.15      0.00    4147.15    4147.15     4147.15      0.00   3.95066   16 16 1 1 0 1 0 0
 1 1073741824 1048576 17179869184 -1 POSIX EXCEL
read         3988.35    3988.35     3988.35      0.00    3988.35    3988.35     3988.35      0.00   4.10796   16 16 1 1 0 1 0 0
 1 1073741824 1048576 17179869184 -1 POSIX EXCEL

Max Write: 4147.15 MiB/sec (4348.60 MB/sec)
Max Read:  3988.35 MiB/sec (4182.09 MB/sec)

Run finished: Mon Nov  1 12:26:29 2021
```

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!

Quiz: Workflow requirement

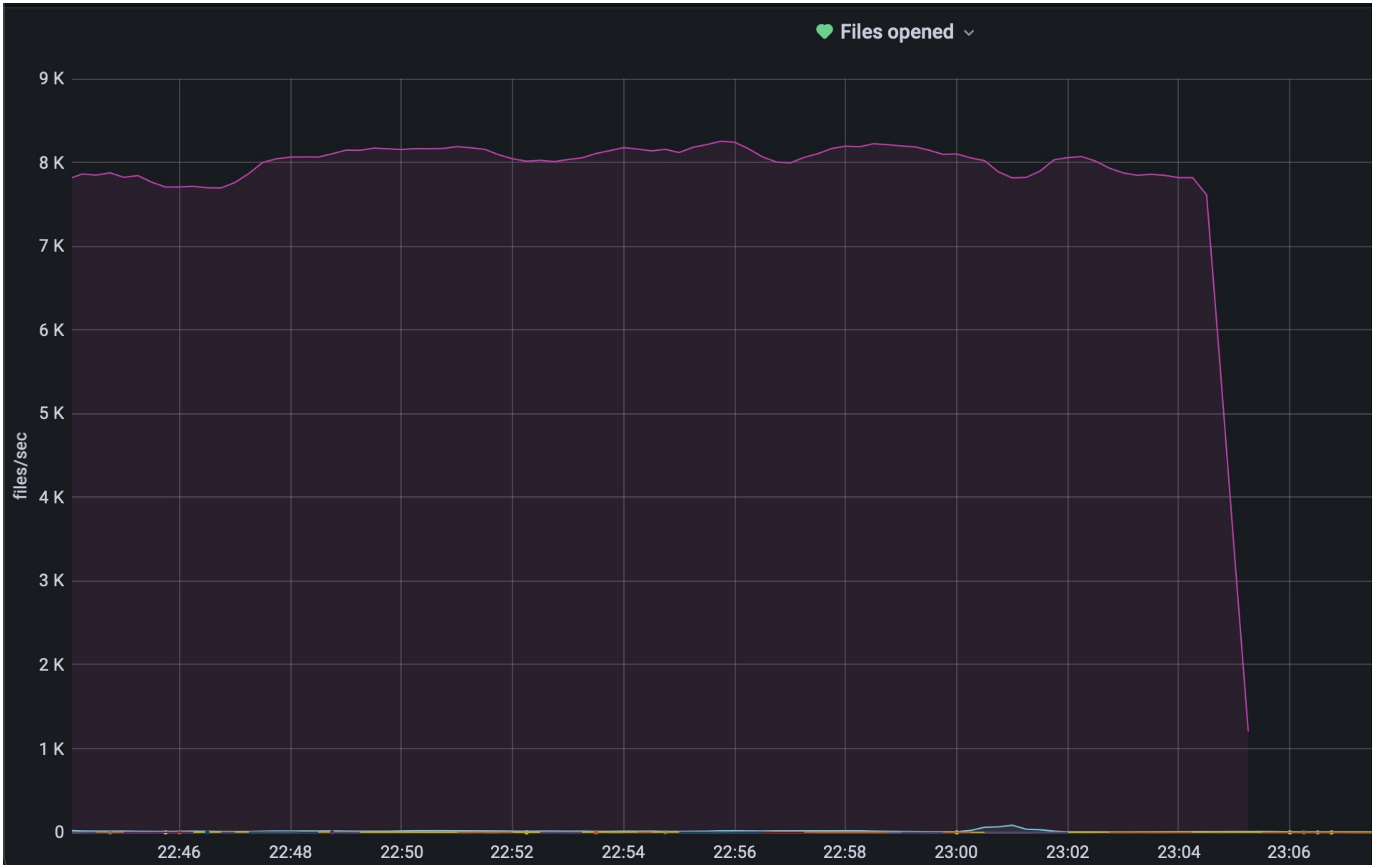
My workflow needs to:

Run 1000s of jobs each of which writes and reads O(10) small files continuously. The jobs need very few cores so most of them may run simultaneously.

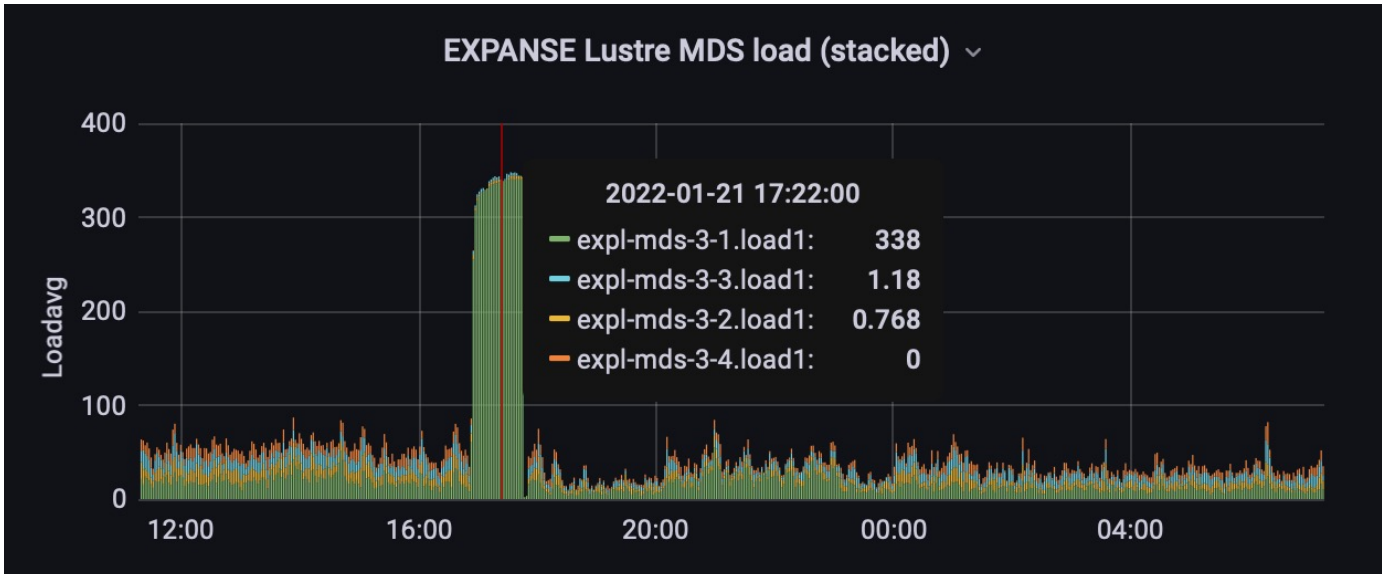
I should consider:

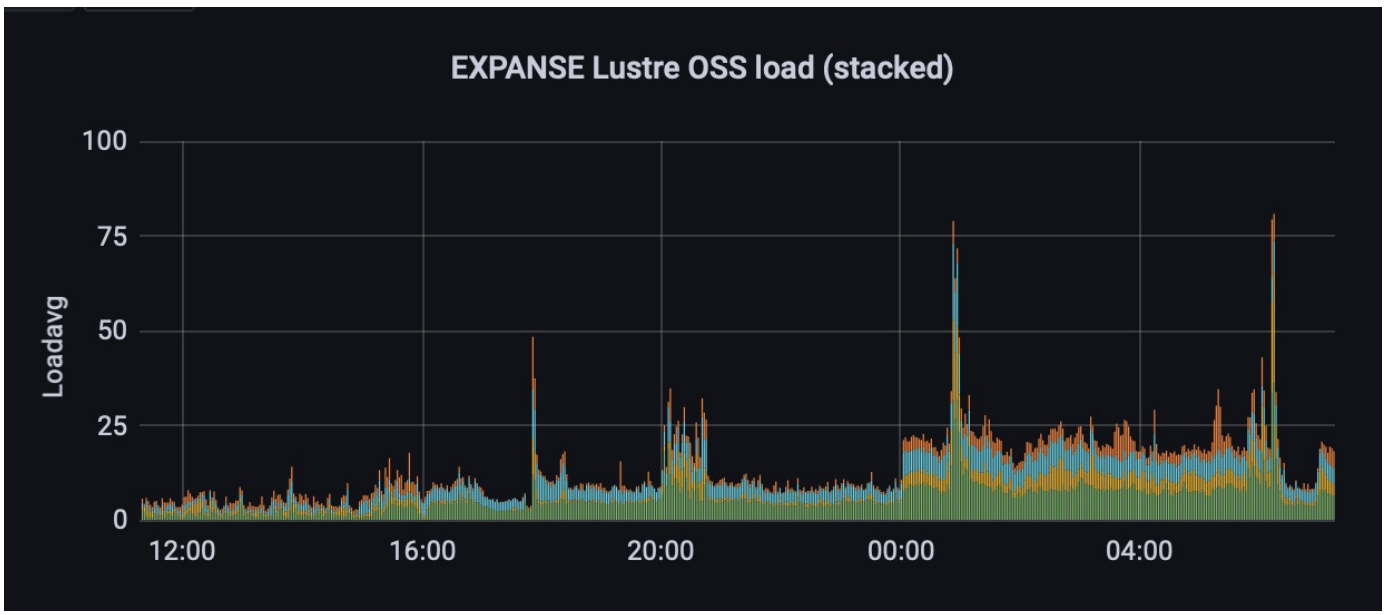
Writing small files to node local NVMe for each job and only copy out results to Lustre

Example of IOPs load from ill-designed IO workflow

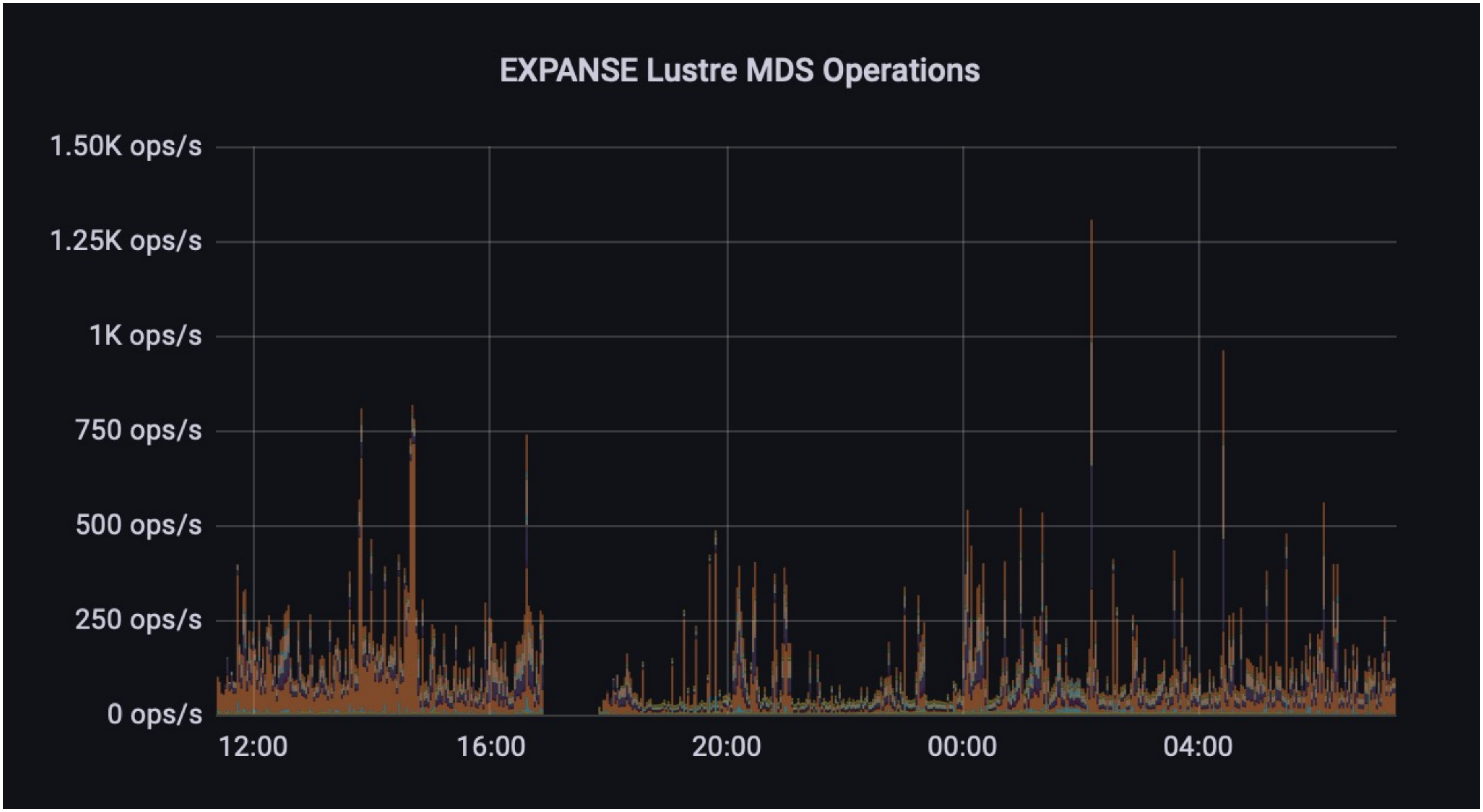


Impact on Filesystem Servers





Impact on Filesystem Servers



Summary

- Data Management is important in all stages of your computational workflow - before, during, and after simulations are done.

- Important for performance and scalability of your computations and workflow. ***Poor IO choices can lead to systemwide issues impacting *all* other users.***

- Data provenance and integrity are important considerations.

- Several filesystem options available on Expanse, ranging from node local NVMe that can handle high IOPs workloads to the large Lustre parallel filesystem that can handle large scale large block IO. ***Choose the right filesystem for your workload depending on IOPs (metadata load) and performance considerations.***

- Always have offsite (multiple) backups of anything critical. Expanse Lustre scratch is on a 90-day purge policy => ***backup anything important.***

Thank You!

Questions?