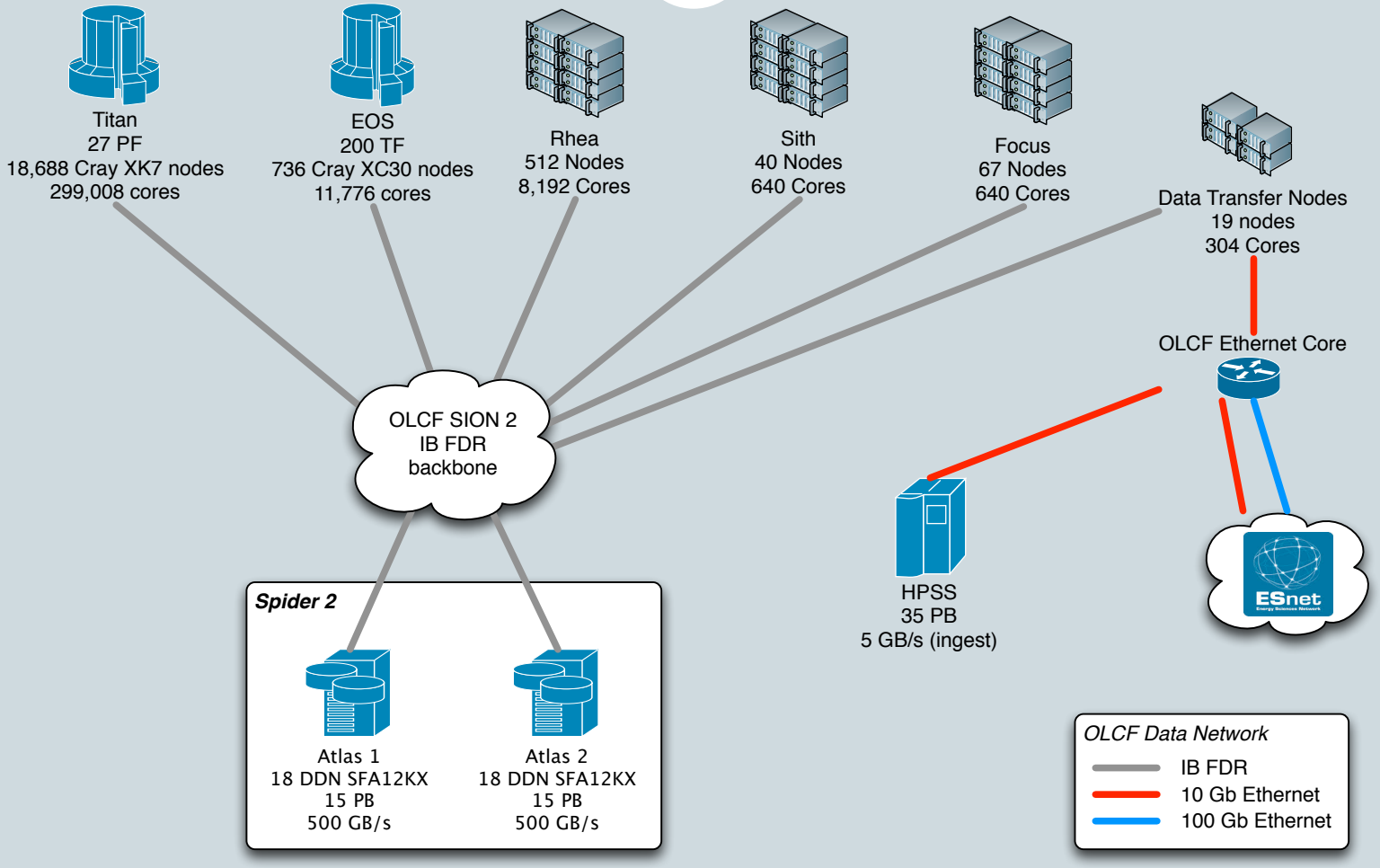
# OLCF I/O best practices

SARP ORAL, PHD

TECHNOLOGY INTEGRATION GROUP
OAK RIDGE LEADERSHIP COMPUTING FACILITY

# OLCF I/O Environment





# OLCF I/O Environment



- Mixed production I/O looks like …


… from the parallel file system point of view

# OLCF I/O Environment



- File system is the canary in a coal mine
  - First user interaction point
  - First point to reveal any system problem
    - Network, memory, storage

- A parallel file system is different than desktop/ laptop file systems
  - A contested and shared resource
  - Vastly more complex
  - Network attached

# OLCF I/O Environment



**A bad behaving application hurts**

**not only itself**

**but ALL running applications!**

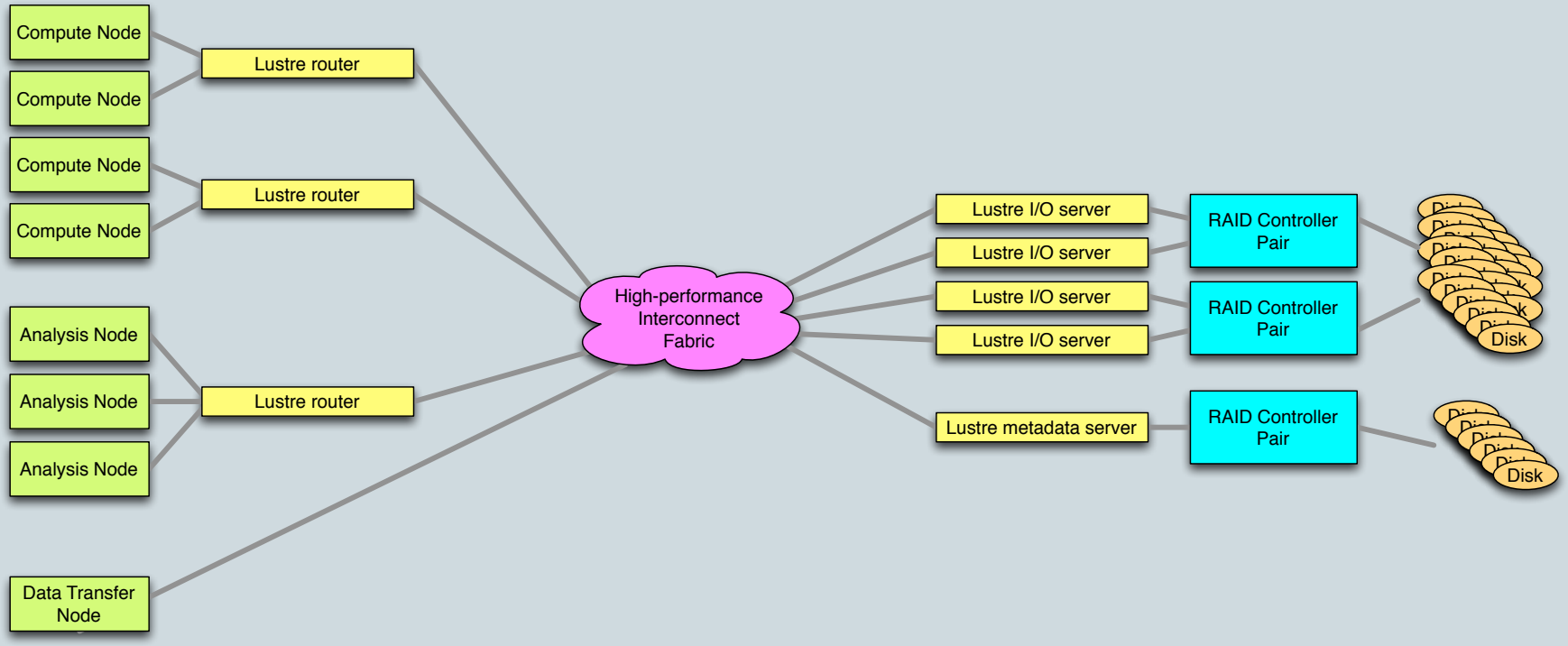
# Lustre Basics



- *Metadata Server (MDS)*
  - Presents the metadata stored in the MDT to Lustre clients
  - Manages the names and directories in a given Lustre file system
- *Metadata Target (MDT)*
  - Stores file system metadata
- *Object Storage Server (OSS)*
  - Provides file service for one or more local OSTs
- *Object Storage Target (OST)*
  - Stores file data (chunks of files) as data objects
  - A single file may be stripped across one or more OSTs
- *Lustre Clients*
  - Nodes which mount the Lustre file system

# Lustre Basics





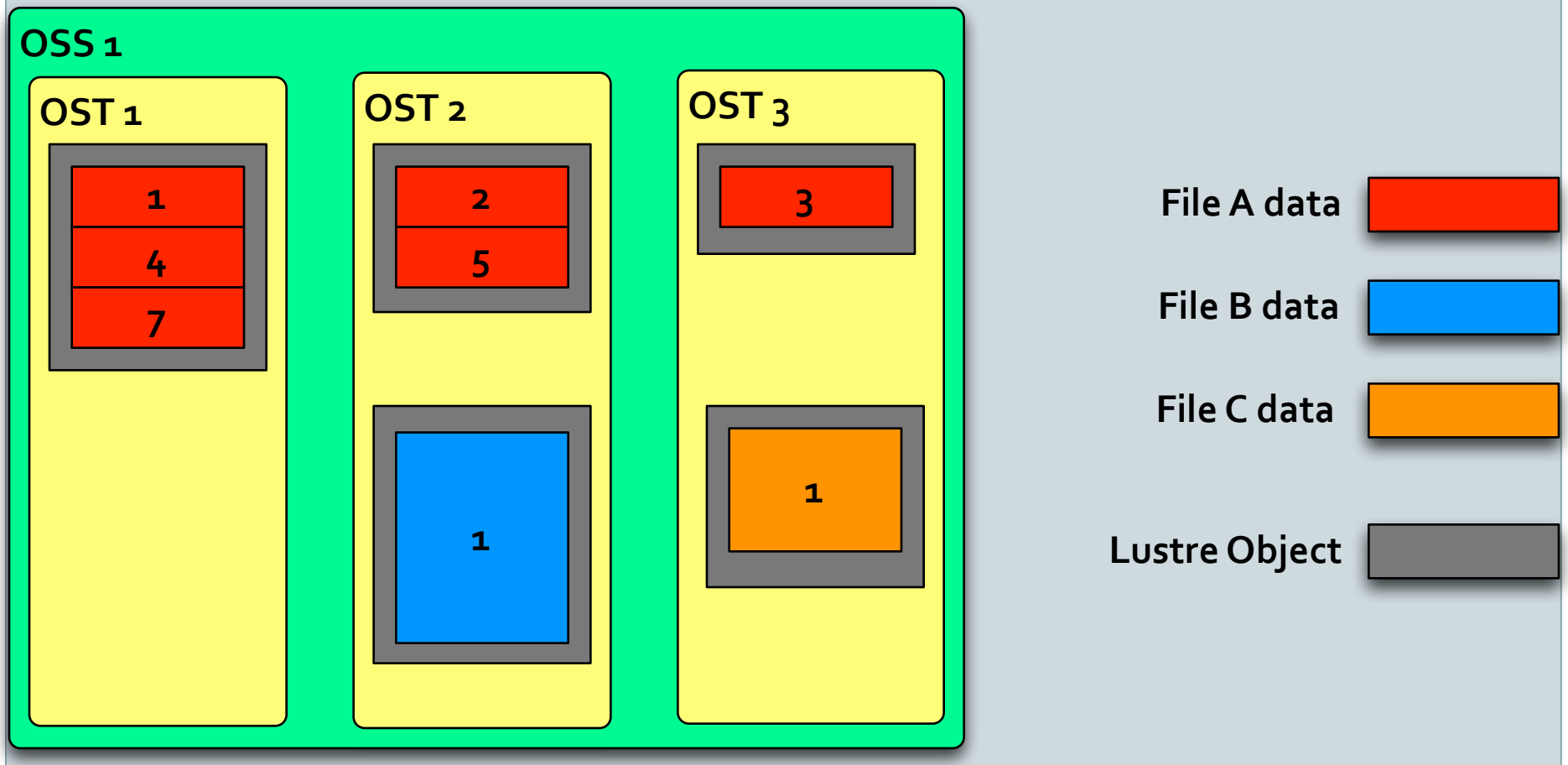
# Lustre Basics - Striping



- File striping
  - Chunks of a file exists on multiple OSTs
- File striping provides
  - *File Size*
    - By placing chucks of a file on multiple OSTs, the space required by the file can also be spread over the OSTs
    - Therefore, a file's size is not limited to the space available on a single OST
  - *Bandwidth*
    - By placing chunks of a file on multiple OSTs, the I/O bandwidth can also be spread over the OSTs
    - In this manner, a file's I/O bandwidth is not limited to a single OST

# Lustre Basics - Striping





# Lustre Basics - Striping



- File striping will most likely improve performance for applications which read or write to a single (or multiple) large shared files

- Striping will likely have little effect for
  - Serial I/O where a single processor performs all the I/O
  - Multiple node perform I/O, but access files at different times
  - Multiple nodes perform I/O simultaneously to different files that are small (each < 100 MB)
  - One file per processor

# Lustre Basics - Striping



- Striping can be set at a file or directory level
- Set striping on an directory then all files created in that directory with inherit striping level of the directory
- Moving a file into a directory with a set striping will NOT change the striping of that file

# Lustre Basics - Striping



- 3 basic Lustre striping controls
  - Stripe size
    - Number of bytes in each stripe (multiple of 64k block)
  - Stripe (OST) offset
    - Starting OST index to perform I/O
    - If not specified Lustre automatically chooses starting OST
  - Stripe count
    - Number of OSTs to stripe over
    - -1 to stripe over all OSTs
    - 1 to stripe over one OST
    - *n* to stripe over *n* OSTs

# Lustre Basics - Striping



- To set Lustre striping use
  - lfs setstripe
- To query Lustre striping use
  - lfs getstripe

# Lustre Basics - Striping



- *lfs setstripe* example
  - lfs setstripe DirPathName -s 1m -i -1 –c 1

- *lfs getstripe* example

```
$ lfs getstripe -q dir/file1
    19      28675008      0x1b58bc0      0
    59      28592466      0x1b44952      0
    70      28656421      0x1b54325      0
    54      28652653      0x1b5346d      0
```

# Lustre at OLCF



- Spider 2 provides two distinct name spaces (file systems)
  - Atlas 1 and 2
- Atlas 1 and Atlas 2 each provides
  - 500 GB/s peak performance
  - 15 PB of usable capacity
  - 1,008 Lustre OSTs
- Spider 2 are for scratch, they are not backed-up!

# Lustre at OLCF



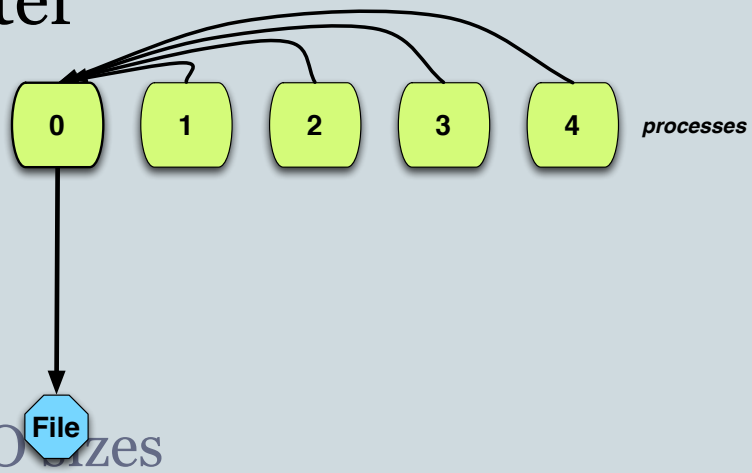
- Once a file is written to Atlas 1 or 2 from any resource (Titan, Eos, DTN, etc) it is accessible from any other resource (Titan, Eos, DTN, etc)

- OLCF default striping
  - 1 MB stripe size
  - Stripe over 4 OSTs (stripe count 4)

# I/O Methods - Serial I/O



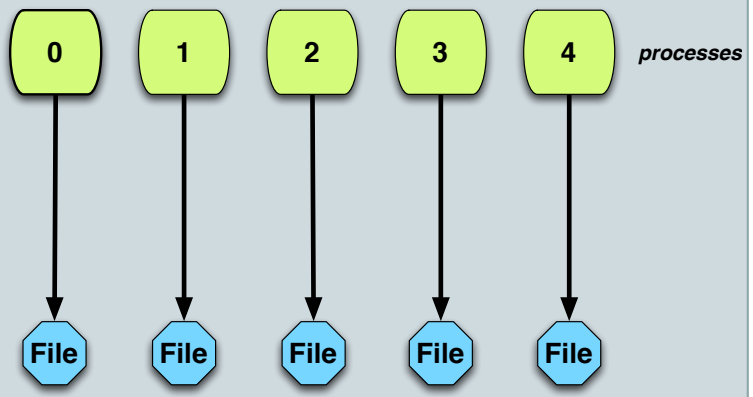
- Processes send data to the master
- Master writes the data to a file
- Read is in the reverse order
- Advantages
  - Simple
  - Good performance for very small I/O sizes
- Disadvantages
  - Not scalable
  - Not efficient, slow for any large number of processors or data sizes
  - May not be possible if memory constrained



# I/O Methods - Parallel I/O File Per Process



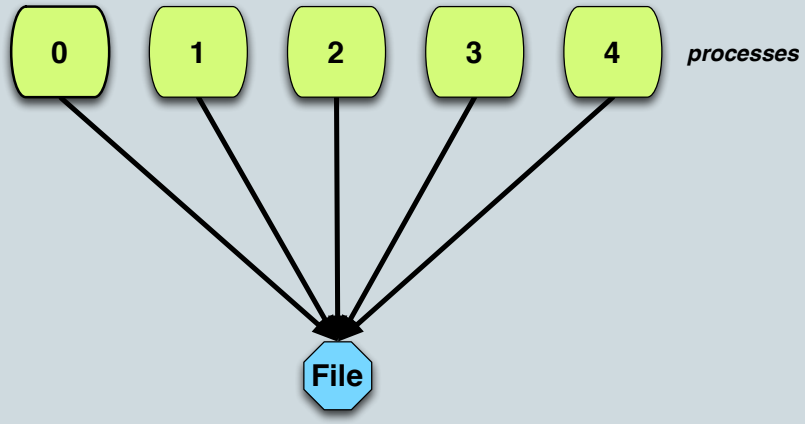
- Each processor writes its own data to a separate file
- Advantages
  - Simple to program
  - Can be fast -- (up to a point)
- Disadvantages
  - Can quickly accumulate many files
  - With Lustre, hit metadata server limit
  - Hard to manage
  - Requires post processing
  - Difficult for storage systems, HPSS, to handle many small files



# I/O methods - Parallel I/O Single Shared File



- Each processor writes its own data to the same file using MPI-IO mapping
- Advantages
  - Single file
  - Manageable data
- Disadvantages
  - Lower performance than one file per processor at some concurrencies



# OLCF Recommendations



- Think this as "preparing the ground"
  - The striping obtains the next time you write to the directory/write a file
  - If you change the settings for an existing directory, you will need to copy the files elsewhere and then copy them back to inherit the new settings.
- Striping is probably most beneficial when the application writes all the data to one file, either by collection or direct access

# OLCF Recommendations



- Many small files, one file per process
  - Use default striping
  - Or single striping (0, -1, 1)
- Large shared files
  - Stripe to some number larger than 4 (0, -1, *n*)

**Do not stripe over all available OSTs (0, -1, -1)**

**Do not stripe more than 512**

**Do not change the starting OST (OST offset -1)**

# OLCF Recommendations



- You can change the striping pattern across the OSTs on a *per directory* basis yourself
- You should have a good understanding of *how and how much* your application outputs before you attempt this!

**Do not fill up individual OSTs**

**Do not stripe your work directory wholesale!**

# OLCF Recommendations



- Edit/build code in user home and project home areas whenever possible
- Use `ls -l` only where absolutely necessary
- Open files as read-only whenever possible
- Read small, shared files from a single task

# OLCF Recommendations



- Limit the number of files in a single directory
- Place small files on a single OST
- Place directories containing many small files on a single OST
- `stat` files from a single task

# OLCF Recommendations



- If possible, use a high-level I/O library or middleware
  - ADIOS, HDF5, pNetCDF, etc
- Community developed parallel libraries are also good
- Use large and stripe-aligned I/O whenever possible

# Some Useful Links



- OLCF Lustre Basics
  - https://www.olcf.ornl.gov/kb_articles/lustre-basics/?nccssystems=Data%20Management
- OLCF Spider Center-wide Lustre File System
  - https://www.olcf.ornl.gov/kb_articles/spider-the-center-wide-lustre-file-system/
- OLCF Spider Best Practices
  - https://www.olcf.ornl.gov/kb_articles/spider-best-practices/

# Some Useful Links



- NERSC I/O case studies
  - *http://www.nersc.gov/users/software/debugging-and-profiling/darshan/i-o-case-studies/*
- Livermore Computing I/O guide
  - *https://computing.llnl.gov/LCdocs/ioguide/*
- Darshan HPC I/O Characterization Tool
  - *http://www.mcs.anl.gov/research/projects/darshan/*

# Useful I/O libraries and middleware



- ADIOS
  - https://www.olcf.ornl.gov/center-projects/adios/
- HDF5
  - https://www.hdfgroup.org/HDF5/
- NetCDF
  - http://www.unidata.ucar.edu/software/netcdf/
- Parallel NetCDF
  - https://trac.mcs.anl.gov/projects/parallel-netcdf

Questions?

help@olcf.ornl.gov

(865) 241-6536

oralhs@ornl.gov