BLUE WATERS

SUSTAINED PETASCALE COMPUTING

# Parallel IO best practices

Andriy Kot

NCSA

NSF

GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION

CRAY

# File systems

- Home
  - Small, global, private, backed up, not purged
  - Store you code
- Project
  - Larger, global, shared among project users, backed up, not purged
  - Store your shared data, store small files
- Scratch
  - Large, global, private, not backed up, purged every 30 days
  - Run your jobs here, store large files temporarily
- /tmp
  - Very small, local, private, not backed up, purged frequently
  - Good for storing intermediate files during compiling/linking

# Improve IO performance: Lustre

- Don't keep to many files in one directory
  - Don't go over 10,000, 1000 or fewer is better
  - Use parallel IO to/from single/few files
  - Hierarchical directory structure: directory is locked for metadata operation
- lfs find
  - Only talks to metadata server as opposed to all servers that hold files (i.e. regular find)
- ~~ls with options~~
  - Don't do it in folder with many files

# Improve IO performance: Striping

lfs getstripe <file/dir>

lfs setstripe –s <size> -c <count> <file/dir>

- Pick appropriate size and count
- Can both increase and **decrease** performance
- Ideally, match stripping configuration to application's IO pattern
- Files inherit striping pattern of directories
- Moving files does not change stripping pattern

# Improve IO performance: Moving data

- Use Globus Online (GO) to move large files to/from BW, to/from Nearline, even between filesystems
- Web interface is easy for limited transfers
- CLI is useful for batch data movement
  - Use the same user id for BW and GO, if not – need to activate endpoints
  - Check the status of transfers with status command, uses transfer id returned by transfer command
- Don't store small files to tape – tar them, avoid storing directories

# Improve IO performance: Parallel access

- Use one file per MPI rank if feasible, when the number of ranks is small
- Avoid reading shared data in its entirety concurrently
  - If small, read by one rank then broadcast
  - If large, read in chunks by many/all ranks then exchange
- Minimize metadata (file system) operations
- Store metadata (application) separately from data

# Improve IO performance: Parallel access

If the number of ranks is high

- Parallel read/write to a single/few large striped files
- Don't use all ranks for IO if the number of ranks is very high -- allocate a subset for ranks for IO
- Make IO process independent of computing process and number of computing ranks
- Try to align reads and writes to striping, aggregate
- Tune IO to access pattern

# Improve IO performance: Parallel access

Use collective IO, parallel IO libraries:

- NetCDF
- HDF5
- NetCDF with HDF5
- IOBUF
- MPI-IO

https://bluewaters.ncsa.illinois.edu/io-libraries

# Use Darshan

Lightweight and scalable I/O profiling tool

Works with POSIX, HDF5, NetCDF and MPIIO

Can be used to investigate and tune I/O behavior of MPI applications.

Darshan is enabled by default

Must be MPI application

https://bluewaters.ncsa.illinois.edu/darshan

# Recent changes

- Number of OSTs decreased while the number of disks per OST increased (1440 → 360)
  - Fewer max stripe count
  - Higher bandwidth to individual OSTs
  - Same aggregate bandwidth