HPC.NRW

aiXcelerate 2021: Part I – File I/O

HPC.NRW Competence Network

DAS KOMPETENZNETZWERK FÜR HOCHLEISTUNGSRECHNEN in NRW.

Best Practices in I/O

Radita Liem (RWTH)

aiXcelerate 2021: Part I – File I/O

Performance Analysis Factors

HPC.NRW

- **Access pattern**
  - Happens within the application
  - Serial I/O, file-per-process, shared-file, collective I/O with MPI-IO or other libraries
- **File system & its configuration**
  - Parallel FS is a shared resource, other applications that are running concurrently might affect the I/O performance
  - Some parallel FS configuration can be adjusted to the application (e.g. file striping)
- **Disk Type and Speed**
  - Hardware dependent, cluster users doesn't have decision on it
- **Network**
  - Shared resource that might affect the performance especially in large scale runs

** This slides are adapted from Introduction to Parallel I/O and Distributed File Systems by Sebastian Oeste - https://tu-dresden.de/zih/hochleistungsrechnen/nhr-training/innovative-storage-architecture/io?set_language=en*

I/O Best Practices – Read/Write

HPC.NRW

- Write/Read only if necessary
  - The best I/O practice is no I/O ☺
  - Switch off debug output for production runs
  - Convert to target/visualization format in memory if possible.
- Perform I/O in few and large chunks.
  - In parallel file systems, the chunk size should be a multiple of the block size or stripe size.
- Read small shared files from a single task.
  - Instead of reading a small file from every task, it is advisable to read the entire file from one task and broadcast the contents to all other task.
- Write/read arrays data structures in one call rather than per element

** This slides are adapted from Introduction to Parallel I/O and Distributed File Systems by Sebastian Oeste - https://tu-dresden.de/zih/hochleistungsrechnen/nhr-training/innovative-storage-architecture/io?set_language=en*

I/O Best Practices – Metadata

HPC.NRW

- Avoid opening and closing files frequently
  - It creates excessive metadata overhead
  - Metadata operations are latency bound
- Open files with correct mode
  - It allows the system to do automatic optimization
- Create files independent from number of processes
  - Easier for post process & future scaling strategy

** This slides are adapted from Introduction to Parallel I/O and Distributed File Systems by Sebastian Oeste - https://tu-dresden.de/zih/hochleistungsrechnen/nhr-training/innovative-storage-architecture/io?set_language=en*

I/O Best Practices – Lustre Glossaries

HPC.NRW

- **Metadata Servers (MDS)**
  - The MDS makes metadata stored in one or more MDTs available to Lustre clients
  - Lustre clients are nodes running Lustre client software
- **Metadata Targets (MDT)**
  - The MDT stores metadata (such as filenames, directories, permissions and file layout) on storage attached to an MDS.
- **Object Storage Servers (OSS)**
  - The OSS provides file I/O service and network request handling for one or more local OSTs
- **Object Storage Target (OST):**
  - User file data is stored in one or more objects, each object on a separate OST in a Lustre.
  - OST can be considered as an equivalent to a disk, although in practice it may comprise multiple disks, e.g. in a RAID array.
  - An individual file can be stored across multiple OSTs; **this is called striping**

I/O Best Practices – File systems

HPC.NRW

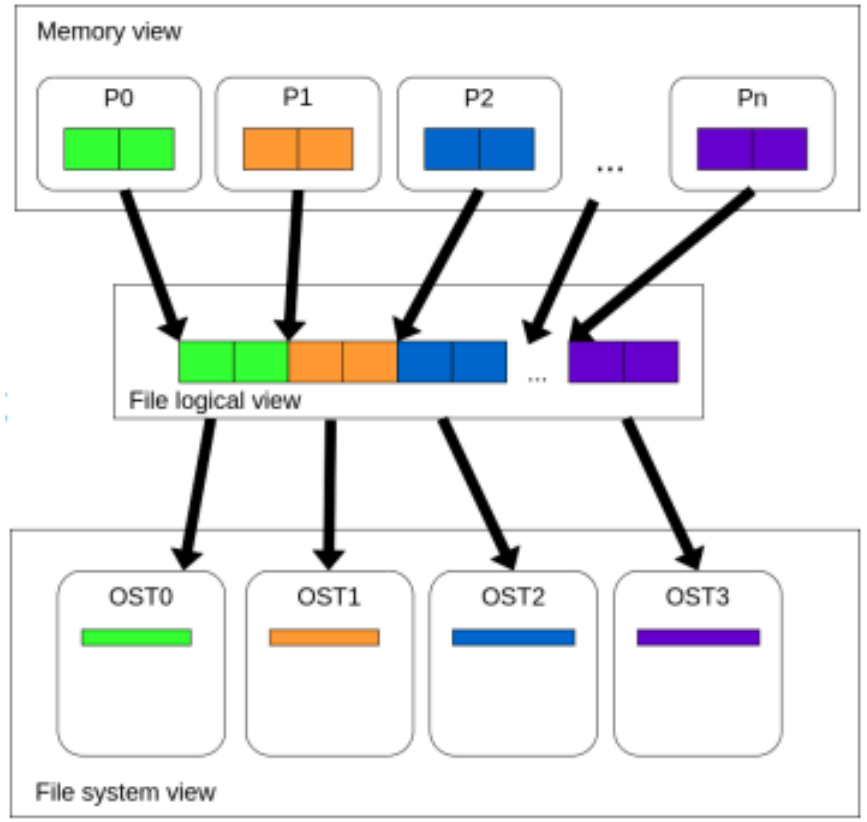
- Use an appropriate file system.
  - Parallel file systems may not scale well for metadata operations, but provide high bandwidth
  - NFS-based file systems may show the reversed behavior.
- Use ‘ls-l’ only when absolutely necessary
  - ‘ls –l’ must communicate with every OST that is assigned to a file being listed and for every file listed. In Lustre, use ‘lfs find’
- Place small files on single OSTs
  - If only one process will read/write the file and the amount of data in the file is small (1GB), performance will be improved by limiting the file to single OST on creation
- Place directories containing many small files on single OSTs
  - If you are going to create many small files in a single directory, greater efficiency will be achieved if you have the directory default to 1 OST on creation

** This slides are adapted from Introduction to Parallel I/O and Distributed File Systems by Sebastian Oeste - https://tu-dresden.de/zih/hochleistungsrechnen/nhr-training/innovative-storage-architecture/io?set_language=en*

I/O Best Practices – Striping Overview

HPC.NRW

- A single file can be split into multiple chunks. A chunk then striped on one or one OSTs.
- Advantages:
  - An increase in the bandwidth available when accessing the file.
  - Increase in the available disk space for storing the file.
- Disadvantages:
  - Increased overhead due to network operations and server contention.
- Most parallel file systems allows user to specify the striping policy for each file or directory of files



** This slides are adapted from Introduction to Parallel I/O and Distributed File Systems by Sebastian Oeste - https://tu-dresden.de/zih/hochleistungsrechnen/nhr-training/innovative-storage-architecture/io?set_language=en*

I/O Best Practices – Striping Considerations

HPC.NRW

- Using more OSTs **does not** increase write performance
- Single writer:
  - Unable to take advantage of file parallelism
  - Access to multiple disks adds overhead which hurts performance
- File per process
  - Performance increases as the number of processes/files until OST and Metadata contention hinder performance improvements.
- Best performance when the I/O operation and stripe size are similar.
- Larger I/O and matching stipe sizes may improve performance (reduces I/O latency)
- OST Contention
  - If each OST is accessed by every process
  - Advised to have each OST only accessed only by one process

I/O Best Practices – Striping Advices

HPC.NRW

- Small file (<1GB) accessed by a single process
  - set stripe count of 1
- Medium sized files (> 1GB) accessed by a single process
  - set to utilize a stripe count of no more than 4
- Large files (10GB)
  - Stripe count should be adjusted to value larger than 4
  - Such files should never be accessed by serial I/O or file-per-process I/O pattern.
- Limit the number of files within a single directory
  - Incorporate additional directory structure
  - Set stripe count of directories that contain many small files to 1

** This slides are adapted from Introduction to Parallel I/O and Distributed File Systems by Sebastian Oeste - https://tu-dresden.de/zih/hochleistungsrechnen/nhr-training/innovative-storage-architecture/io?set_language=en*

I/O Best Practices – I/O Libraries

HPC.NRW

- Consider I/O middleware libraries like HDF5, MPI-IO, pnetCDF, SIONLib, ADIOS
  - But it does not mean using these libraries will magically fix everything ☺
- Specialized I/O libraries may provide more a portable way of writing data and may reduce metadata load when properly used
- For parallel programs the output to separate files for each process can provide high throughput
  - It usually needs additional post-processing.
- Binary files may need to use library/compiler support for conversion
  - If binary files are transferred between different architectures (little vs big-endian byte order) then the limitations may apply on file sizes and data types.
- Utilize MPI-IO Hints to improve performance

** This slides are adapted from Introduction to Parallel I/O and Distributed File Systems by Sebastian Oeste - https://tu-dresden.de/zih/hochleistungsrechnen/nhr-training/innovative-storage-architecture/io?set_language=en*

I/O Best Practices – I/O Libraries: Using Binary File

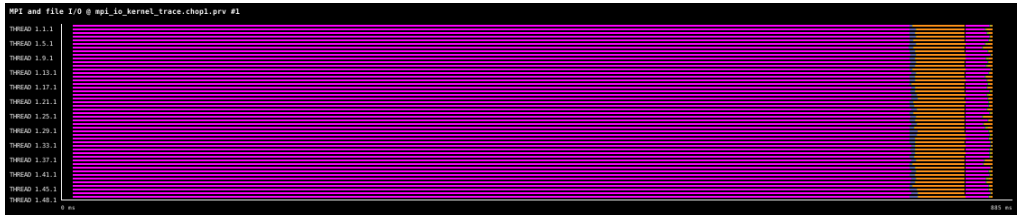
HPC.NRW

- A parallel library can make optimal use of any underlying parallel file system, and will give better performance than serial file I/O.
  - Additionally, reading and writing binary is more efficiency than writing the equivalent ASCII data.

MPI-IO Collective Access

NetCDF Collective Access

MPI-IO Independent Access



NetCDF Independent Access

* Taken from https://co-design.pop-coe.eu/best-practices/parallel-library-file-io.html