

Cray XT4 File Systems & IO

Richard Gerber

NERSC User Services

RAGerber@lbl.gov

Thanks to Katie Antypas & Andrew Uselton



**SciDAC 2008 Tutorials
Redmond, WA
July 18, 2008**



Outline

- **File Systems**
- **System Layout**
- **Best Practices**
- **Details**
- **Reference**
 - **NERSC:** www.nersc.gov
 - **NCCS:** www.nccs.gov



File Systems



What is a File System?

- **A special-purpose database for the storage, hierarchical organization, manipulation, navigation, access, and retrieval of data.**
 - **This is a layer that mediates between the Operating System and the Storage Device.**
- **We often refer to a “file system name” as the root of a hierarchical directory tree, e.g. “the /home file system.”**
 - **We can treat this as “one big disk,” but it may actually be a complex collection of disk arrays, IO servers, and networks.**
- **A file system deals with “data” and “metadata” (data about the data, e.g. file name, physical location on disk, file size, timestamps)**



File Systems on the XT4

- **“Scratch” or “Work”**
 - Large temporary high-performance file systems
 - To be used for parallel job IO
 - Not backed up
 - Each user has a unique directory
 - ✓ **NERSC: \$SCRATCH**
 - ✓ **NCCS: /tmp/work/\$USER**
 - **Quotas at NERSC; 14-day purge at NCCS**
- **Home**
 - You are in your “home” directory when you log in
 - Permanent storage for source code, binaries, scripts, ...
 - Small(ish) quota; not intended for data
 - Use \$HOME to reference your home directory
 - Site-dependent details



File Systems on the XT4 (2)

- **Project**
 - Use to share files among group members
 - Not high performance
 - Quotas
 - **NERSC: by request, /project/projectdirs/proj/**
 - **NCCS: all users, /ccs/proj/**
- **/tmp**
 - Reserved for system use; **DO NOT USE!!**
- **Archival Storage**
 - HPSS mass storage systems
 - Extremely high capacity
 - Tape storage with disk cache
 - “hsi” is the shell interface utility (ftp-like)



Implications for Jobs

- Your parallel application (launched with `aprun`) can only access `/scratch` or `/tmp/work`.
- Serial (shell) script commands can access all file systems.
- Use `cd $PBS_O_WORKDIR` in script to change to submission directory.
- `STDERR` and `STDOUT` are buffered and returned at job completion.



System Layout



System IO Architecture

- **Should an application scientist/programmer care about these details?**
 - **Yes! It would be nice not to need to, but performance and perhaps functionality depend on it.**
 - **You may be able to make simple changes to the code or runtime environment that make a big difference.**



Network File Systems

- **All disk storage on the XT4 is accessed “externally” as a network file system.**
- **What is a “network file system?”**
 - **A file system that supports sharing of files as persistent storage over a network.**
 - **Network File System (protocol) (NFS)**
 - ✓ NFS is a standard protocol
 - ✓ Widely used and available, but not developed as a standard for high-performance parallel computing
 - **Lustre**
 - ✓ High-performance file systems on the XT4 are Lustre file systems
 - **Other examples: AFS, NetWare Core Protocol, Server Message Block (SMB).**



XT4 IO Network Fundamentals

- **The XT4 has two types of nodes: compute (CNL) and service (login, IO, network; full Linux)**
- **All nodes are connected by a high-speed Seastar 2 torus network (aka, “The torus”).**
- **IO service nodes are also connected to large, high-performance disk servers by a fast “Fibre Channel” network.**
- **Login and batch service nodes are further connected to HPSS and other disk servers via a gigabit ethernet network.**



Terminology: Fibre Channel

- **Fibre Channel**
 - **Gigabit network technology primarily used for storage networking.**
 - **Fibre Channel Protocol (FCP) is similar to TCP for FC networks**
 - **Can run over copper or fibre-optic cables.**
 - **Typically, you have a FC card on a node, similar to a giga-bit ethernet card.**



Terminology: Lustre

- **Lustre (name derived from “Linux Cluster”)**
- **A clustered, shared file system**
- **Open software, available under GNU GPL**
- **Designed, developed, and maintained by Sun Microsystems, Inc., which acquired it from Cluster File Systems, Inc. in Oct. 2007**
- **Two types of Lustre servers (on IO service nodes)**
 - **Object Storage Servers (OSS)**
 - **Metadata Servers (MDS)**



Terminology: Metadata

- **File systems store information about files externally to those files.**
- **Linux uses an inode, which stores information about files and directories: size in bytes, device id, user id, group id, mode, timestamps, link info, pointers to disk blocks, ...**
- **Any time a file's attributes change or info is desired (e.g., `ls -l`) metadata has to be retrieved or written.**
- **Metadata operations are IO operations and inodes use disk space.**



Types of File Systems on the XT4

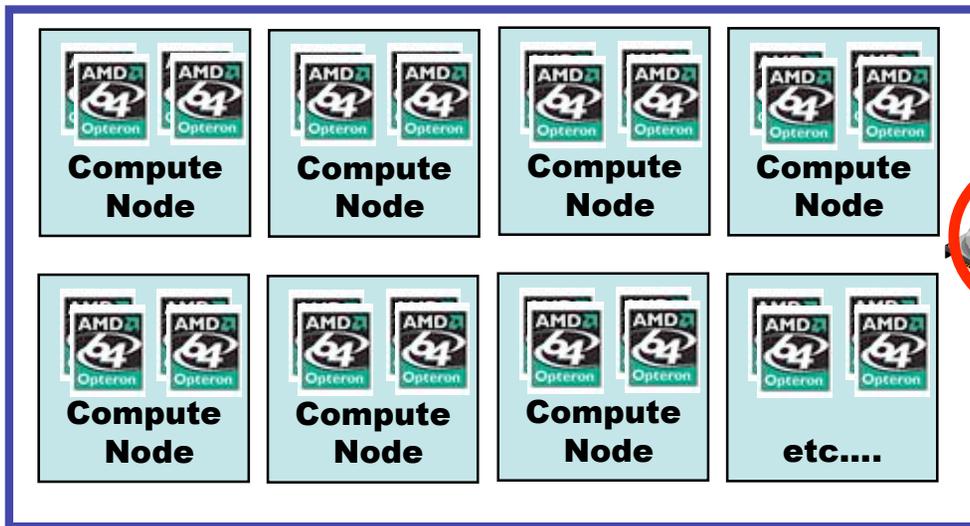
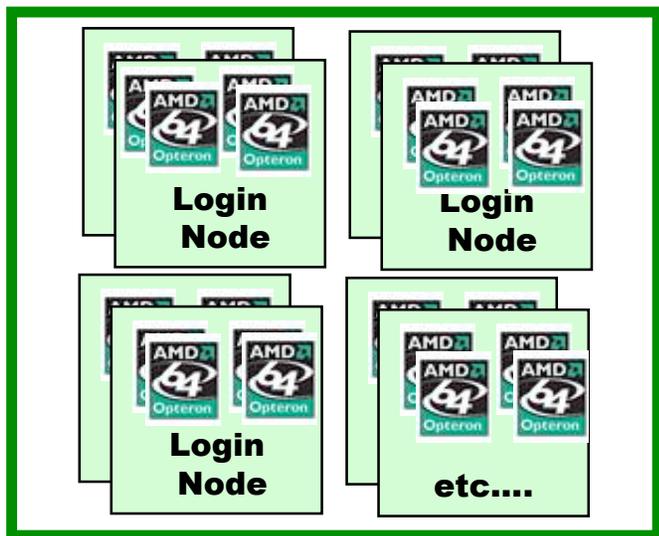
- **Lustre**
 - **Scratch or Work directories are Lustre file systems.**
 - **The libluster library is available for CNL and Linux, thus Lustre file systems are available from all nodes.**
 - **At NERSC, \$HOME is a Lustre file system.**
- **NFS or similar protocol (may be proprietary)**
 - **Project directories**
 - **\$HOME at NCCS**
 - **No client or library support within CNL, thus no access from compute nodes.**



Cray XT4 File System Visibility

Full Linux OS

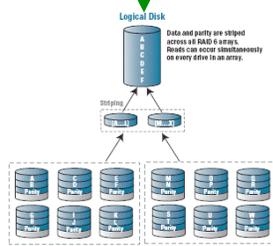
CNL (no logins)



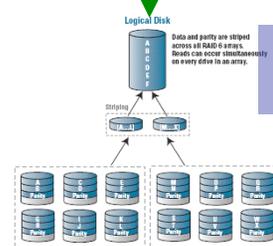
No local disk



HPSS

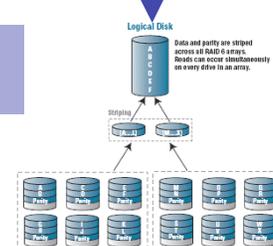


project



home

NERSC only



scratch work



Application IO

- All IO performed by your job should use the file system designed for HPC applications.
- **\$HOME at NERSC is not configured for good application IO performance.**
- Lustre is currently the only file system that can be used by parallel applications, so we'll concentrate on it.



Some XT4 Lustre Terminology

- **Object Storage Server (OSS)**
 - Some number of XT4 service nodes are dedicated to IO and serve as OSSs.
 - A file system partition (e.g. `/scratch`) can be served by multiple OSSs.
 - OSSs are connected to the high-speed torus and also via a fibre-channel network to physical disks.
- **Object Storage Target (OST)**
 - OSTs are the fundamental unit of disk as seen by the OS.
 - Each OSS hosts 4 OSTs
 - OSTs are combined into a file system partition that is presented to users.
 - A partition (e.g. `/tmp/work`) can be viewed as being built from a number of independent OSTs.
- **Metadata Server (MDS)**
 - An XT4 IO service node can also be configured as an MDS.
 - The MDS deals with all information about individual files.
 - One MDS per file system partition.
 - Any time you request info about or modify a file, a request has to be made to an MDS.

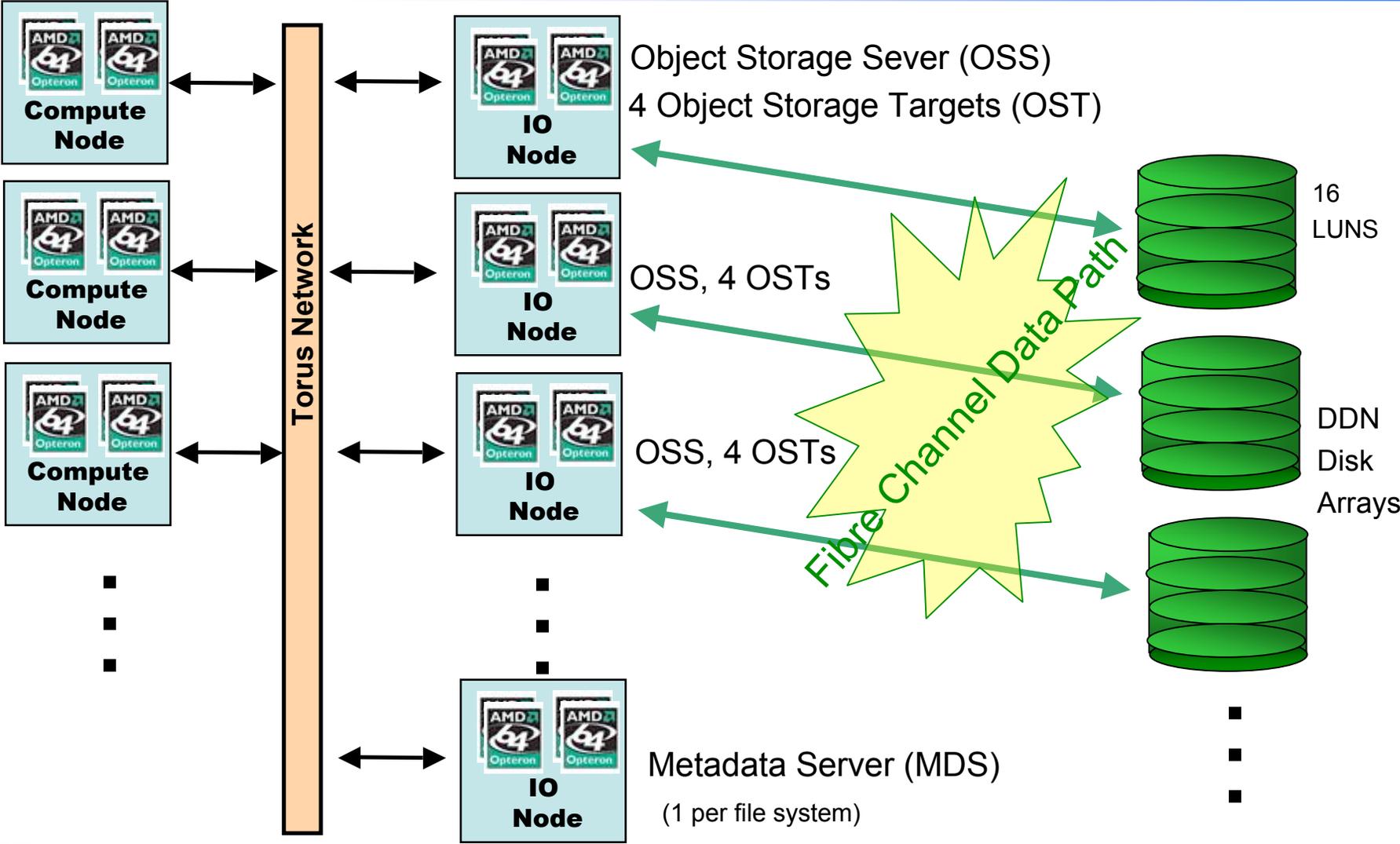


Physical Disks

- **Physical disk storage resides on a DDN (Direct Data Networks) Storage Appliance**
- **The “DDN Disk Arrays” include supporting software and connectivity.**
- **The DDN server presents collections of hard disks as a Logical Unit Number (LUN) to the file system.**
- **One Lustre OST maps to one 2TB or 4TB LUN.**
- **DDN appliance aggregates 16 LUNS; aggregate bandwidth to DDNs may be limiting factor on performance.**



XT4 Lustre Connectivity





Best Practices

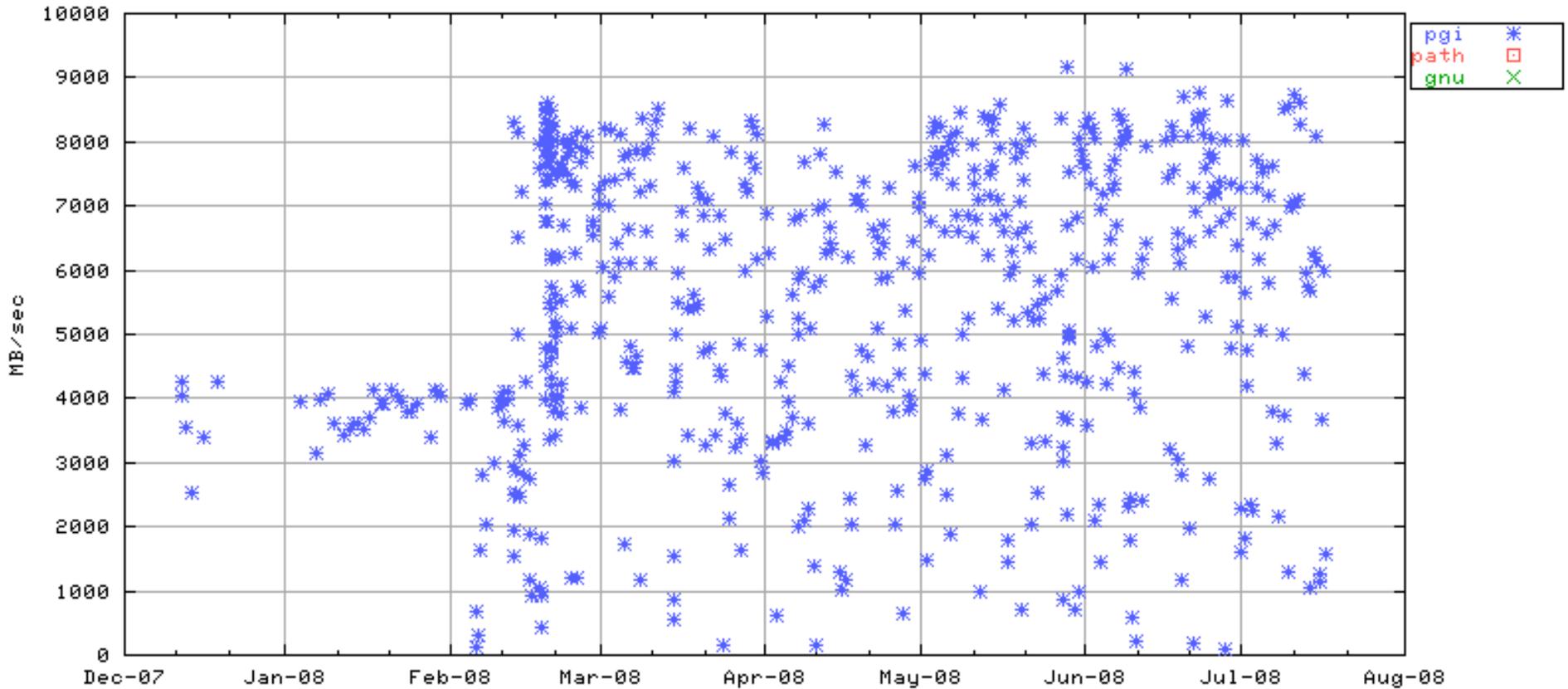


Performance

- **IO performance is complicated to predict.**
- **Other users impact your job because IO uses a shared resource.**
- **Buffer caches exist throughout the system (on disks, in LUNs, on OSS, ...), adding to the unpredictability.**
- **Data paths into single elements (e.g., a node, OST) are limiting for large IO.**
- **Accessing many components (e.g. multiple data paths, OSTs and beating on the MDS) for small IO requests has a high overhead.**



IOR POSIX Write Rate on Franklin



64 MPI Tasks, 1 file per task



Best Practices

- **Do large IO: write lots of data (1MB+) at once.**
 - Best for network performance and disk operations.
 - Also reduces MDS accesses.
- **Do parallel IO.**
 - Serial IO (single writer) can not take advantage of the system's parallel capabilities.
- **Stripe large files over many OSTs.**
- **Use a single, shared file instead of 1 file per writer, esp. at high parallel concurrency.**
- **If job uses > 2,000 tasks, reduce the number of tasks performing IO (experiment with this number)**
- **Use an IO library API and write flexible, portable programs.**



Lustre File Striping

- Files are broken into chunks that are stored on OSTs in a round-robin fashion.
- The size of the chunks and number of OSTs can be set by the user
 - `lstripe <filename|dirname> <size><start><count>`
 - Can set for a file or directory. If directory, new files in directory will inherit setting.
 - size = size of chunk, 0 signifies default of 1 MB
 - start = starting OST; you should use -1 to let the system decide
 - count = number of OSTs; 0 means use default, -1 means use all



File Striping

- `lfs df` will show you the OSTs in each file system and how full they are.
- `lfs getstripe <file|dir>` will show you striping info for a file or directory.
- The default stripe count is **4 on NERSC's /scratch**. Stripe large files over many OSTs.
- Small files should not be striped.

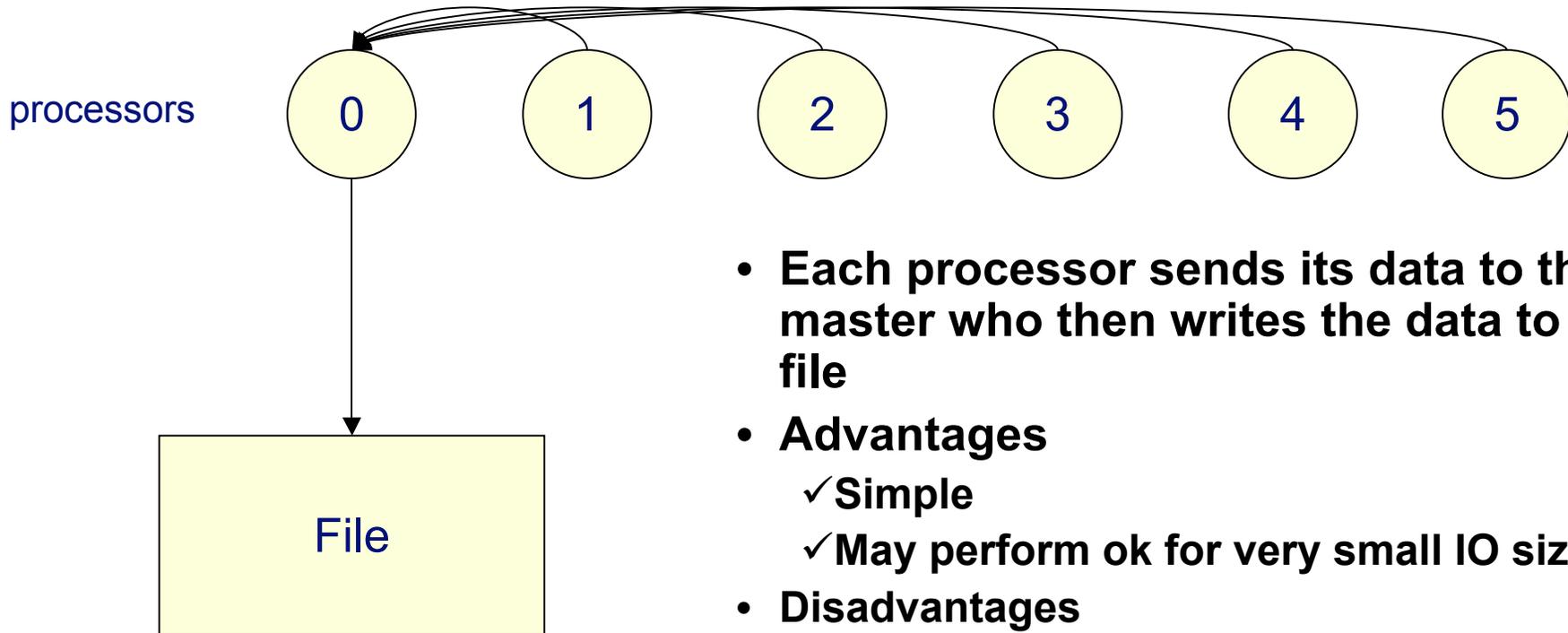


High Level IO Strategies

- **Single task**
- **One file per task**
- **Single file using all tasks**
- **$n < N$ tasks write to a single file**
- **$n_1 < N$ tasks write to $n_2 < N$ files**



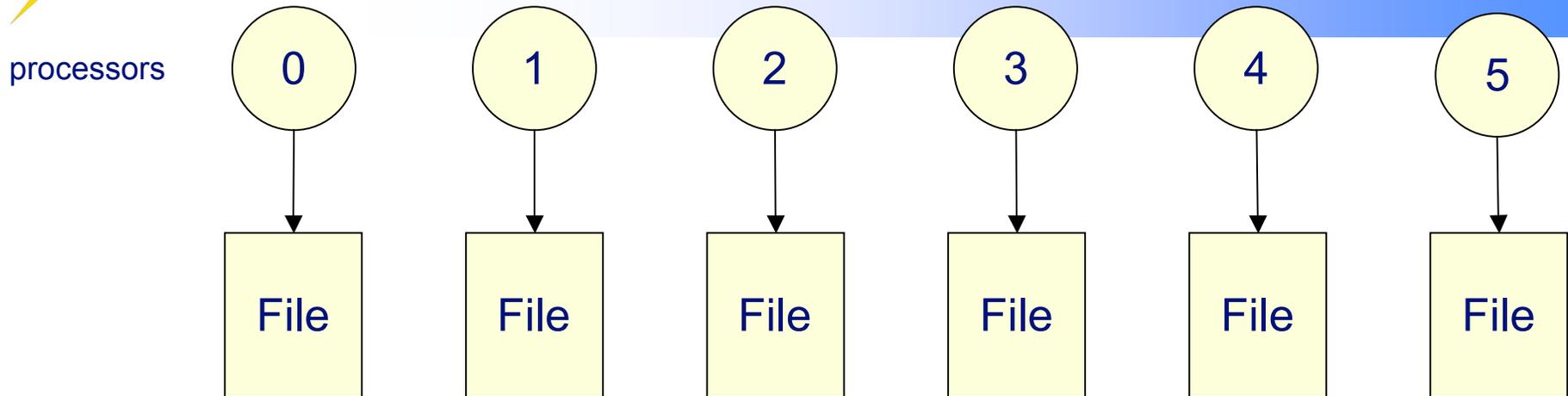
Serial I/O



- **Each processor sends its data to the master who then writes the data to a file**
- **Advantages**
 - ✓ Simple
 - ✓ May perform ok for very small IO sizes
- **Disadvantages**
 - ✓ Not scalable
 - ✓ Not efficient, slow for any large number of processors or data sizes
 - ✓ May not be possible if memory constrained



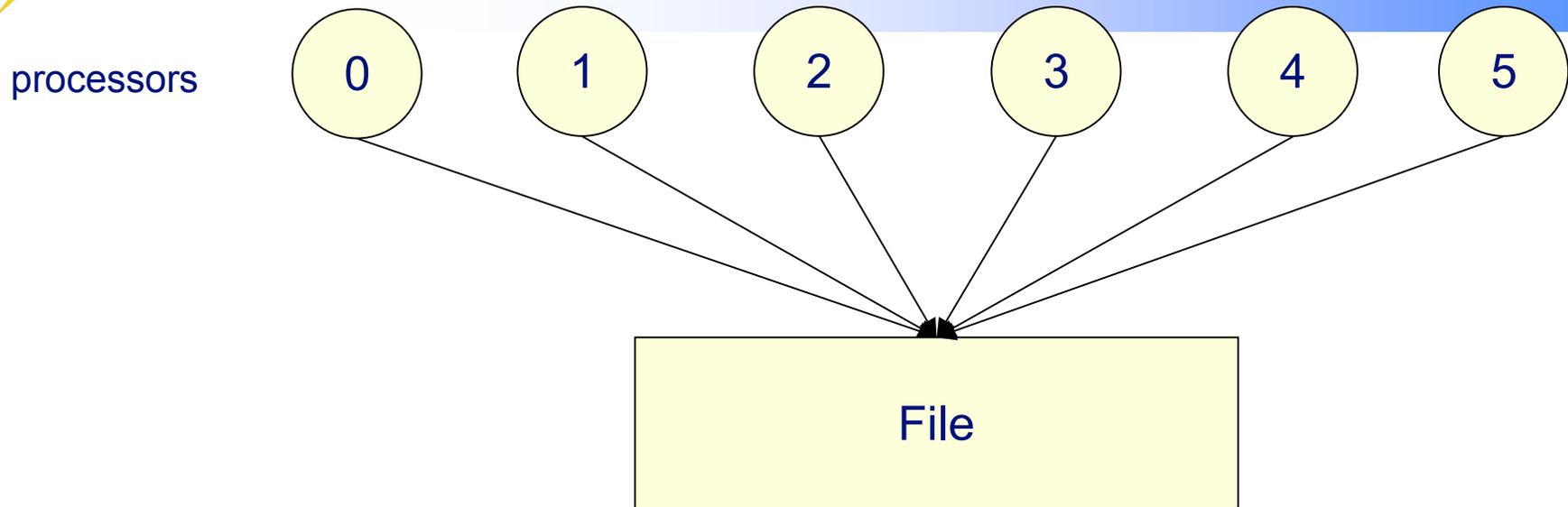
Parallel I/O Multi-file



- **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 many (10-100K) files
 - ✓ Requires post processing
 - ✓ Difficult for storage systems, HPSS, to handle many small files



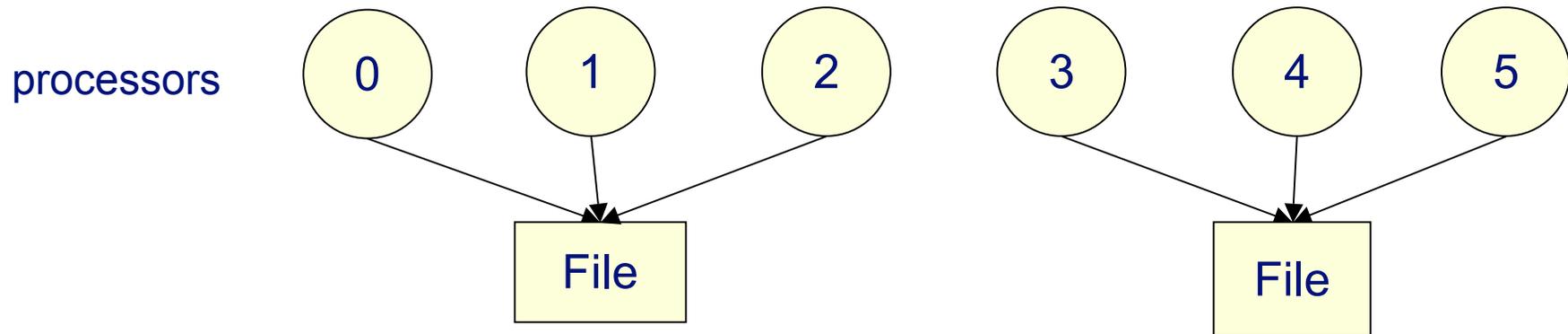
Parallel I/O Single-file



- **Each processor writes its own data to the same file using parallel library API**
- **Advantages**
 - ✓ Single file
 - ✓ Manageable data
- **Disadvantages**
 - ✓ Lower performance than one file per processor at some concurrencies



Hybrid Model I

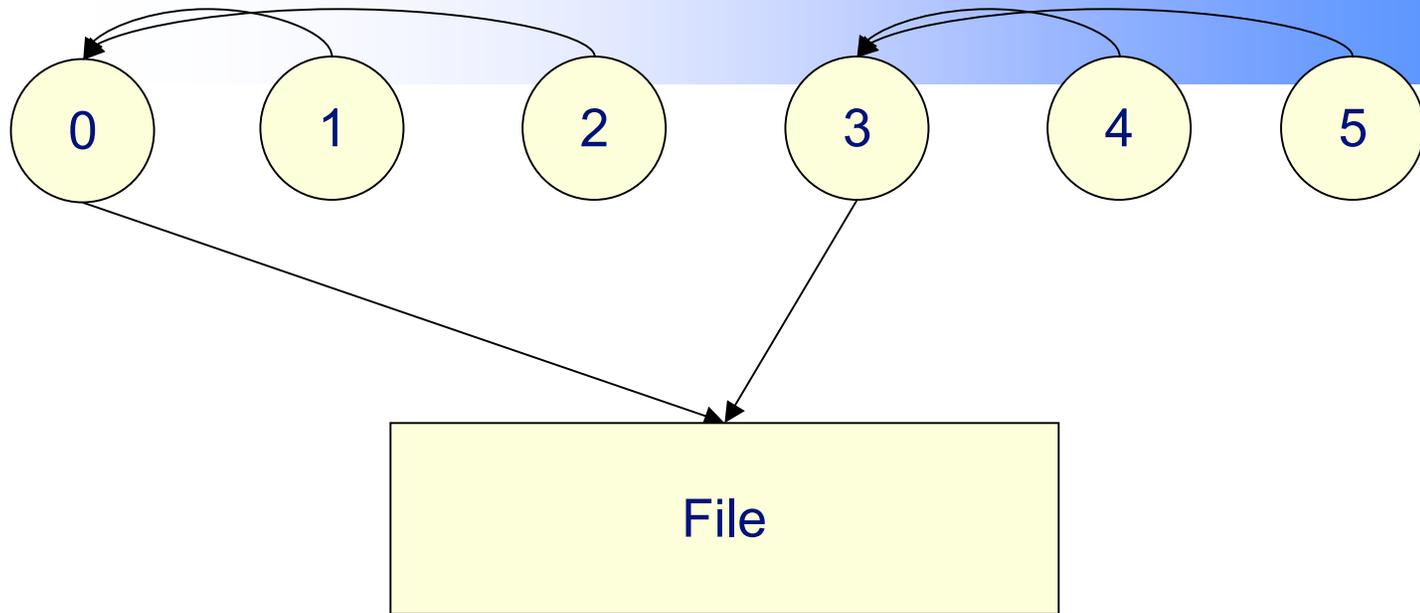


- **Intermediate between file per processor and single shared file.**
- **Advantages of both models**
 - **Reduced number of files**
 - **Better performance and scaling than single shared file**
- **Adds algorithmic complexity**



Reduced Writers to Single-file

processors

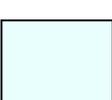
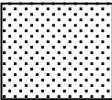


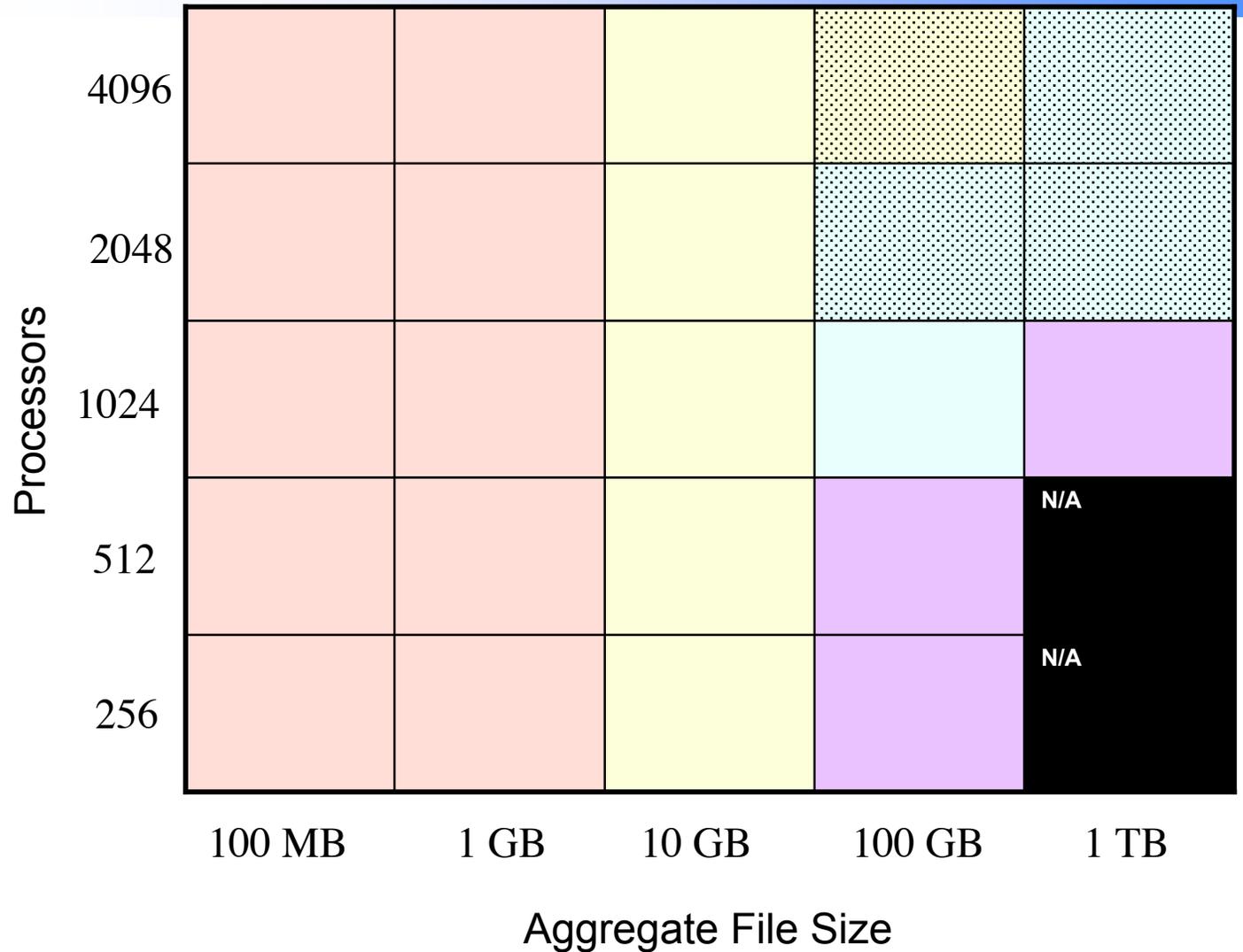
- **A subset of processors writes its own data to the same file using a parallel library API.**
- **Advantages**
 - ✓ Single file; manageable data
 - ✓ Better performance than all tasks writing for high concurrency jobs
- **Disadvantages**
 - ✓ Lower performance than one file per processor at some concurrencies



Recommendations

Legend

	Single Shared File, Default or No Striping
	Single Shared File, Stripe over some OSTs (~10)
	Single Shared File, Stripe over many OSTs
	Single Shared File, Stripe over many OSTs OR File per processor with default striping
	Benefits to mod n shared files





Common Storage Formats

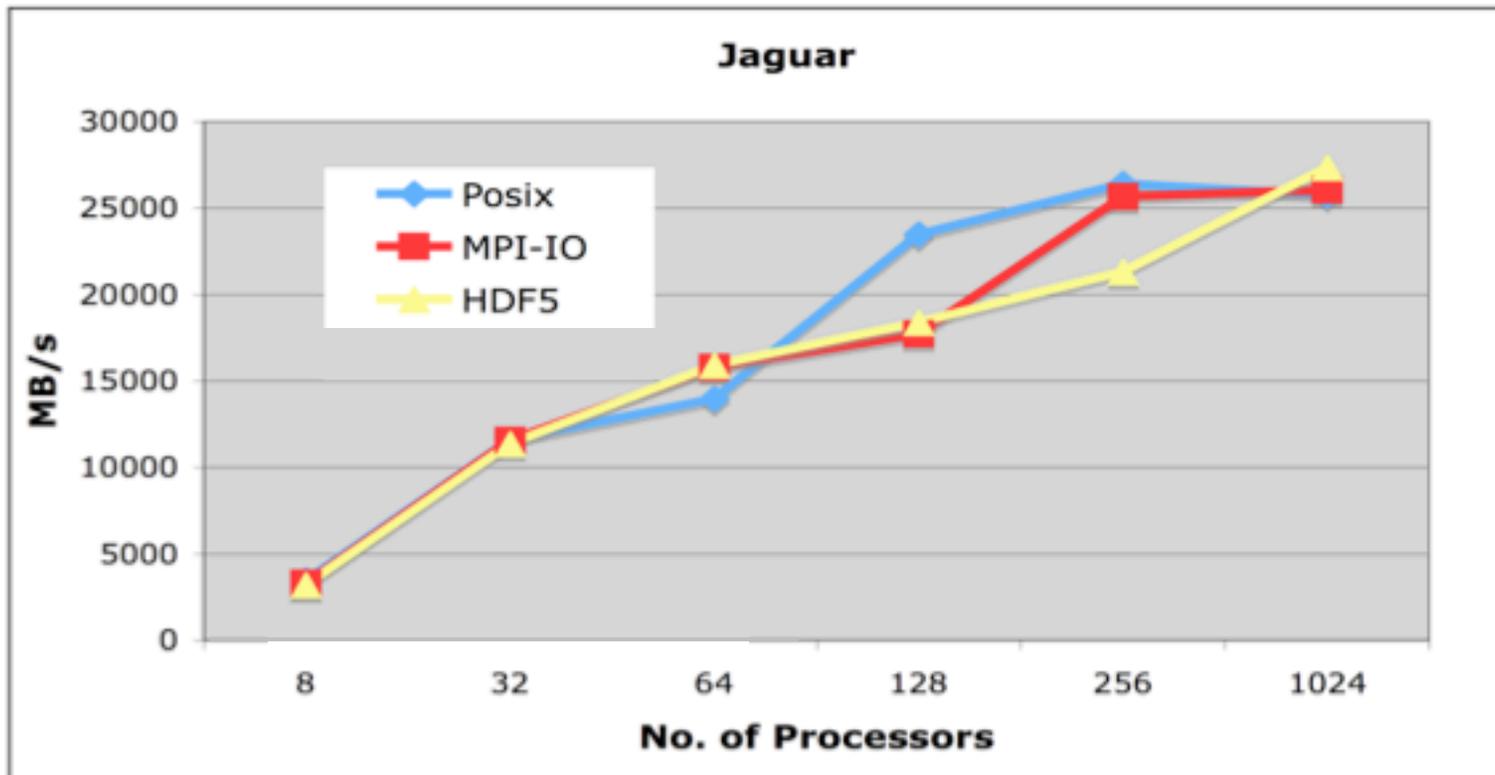
- **ASCII:**
 - **Slow**
 - **Takes more space!**
 - **Inaccurate**
- **Binary**
 - **Non-portable (eg. byte ordering and types sizes)**
 - **May not be able to read using file systems of the future**
 - **May be fastest format and can use parallel MPI-IO library**
- **Self-Describing formats**
 - **NetCDF, HDF5, Parallel NetCDF, Parallel HDF5**
 - **Portable and accessible as long as libraries are supported**
 - **Performance now rivals native formats**
- **Community File Formats**
 - **FITS, HDF-EOS, SAF, PDB, Plot3D**
 - **Modern Implementations built on top of HDF, NetCDF, or other self-describing object-model API**



A Plug for Self Describing Formats ...

- **Application developers shouldn't care about about physical layout of data**
- **Using own binary file format forces user to understand layers below the application to get optimal IO performance**
- **Every time code is ported to a new machine or underlying file system is changed or upgraded, user is required to make changes to improve IO performance**
- **Let other people do the work**
 - **HDF5 can be optimized for given platforms and file systems by HDF5 developers**
 - **User can stay with the high level**
- **But what about performance?**

IO Library Overhead

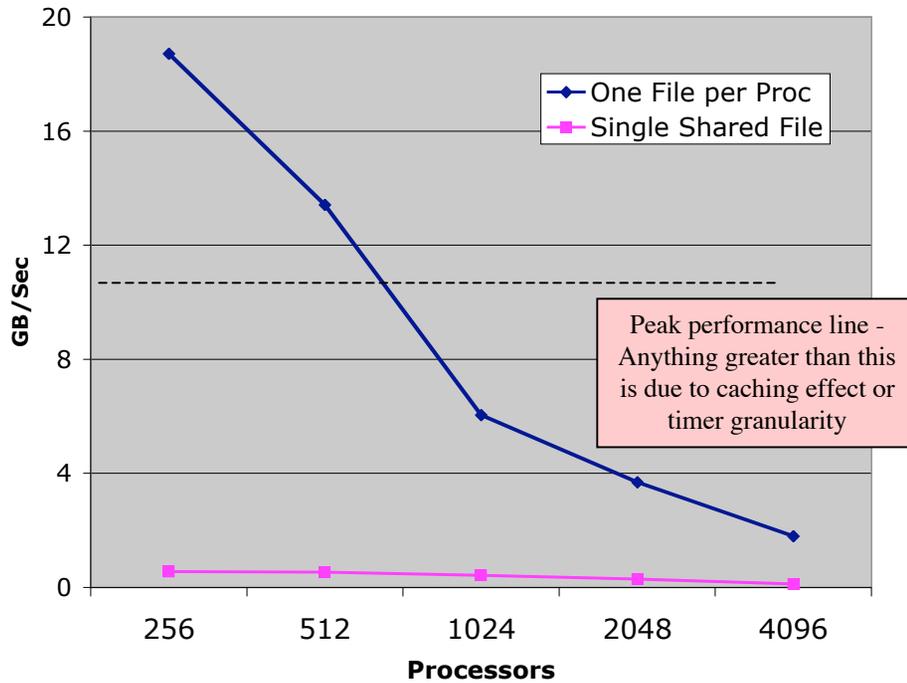


Very little, if any overhead from HDF5 for one file per processor IO compared to Posix and MPI-IO

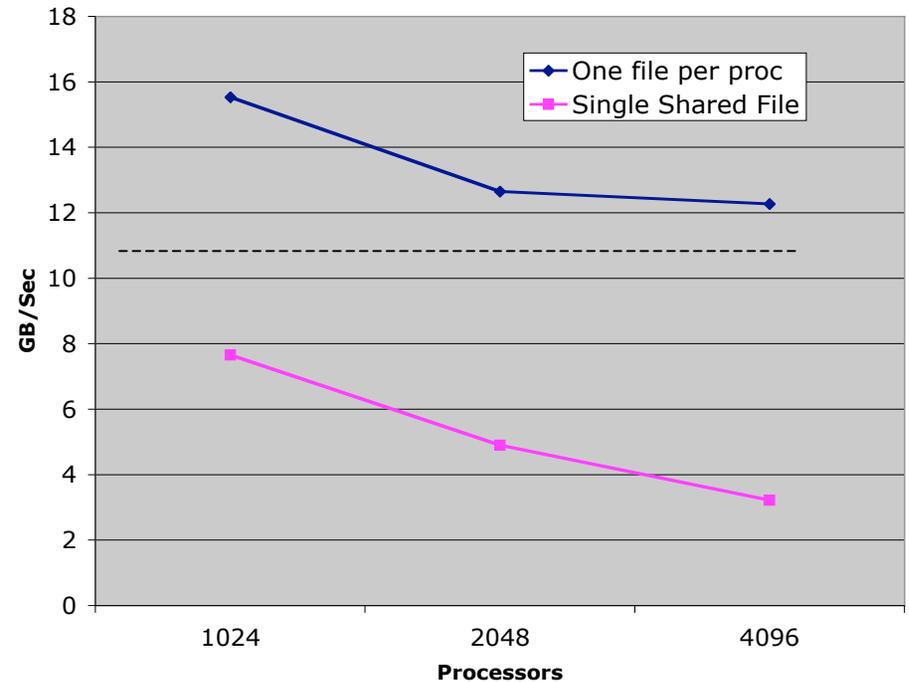


File Per Processor vs Shared File : Rate GB/Sec

Aggregate File Size 1 GB



Aggregate File Size 1 TB

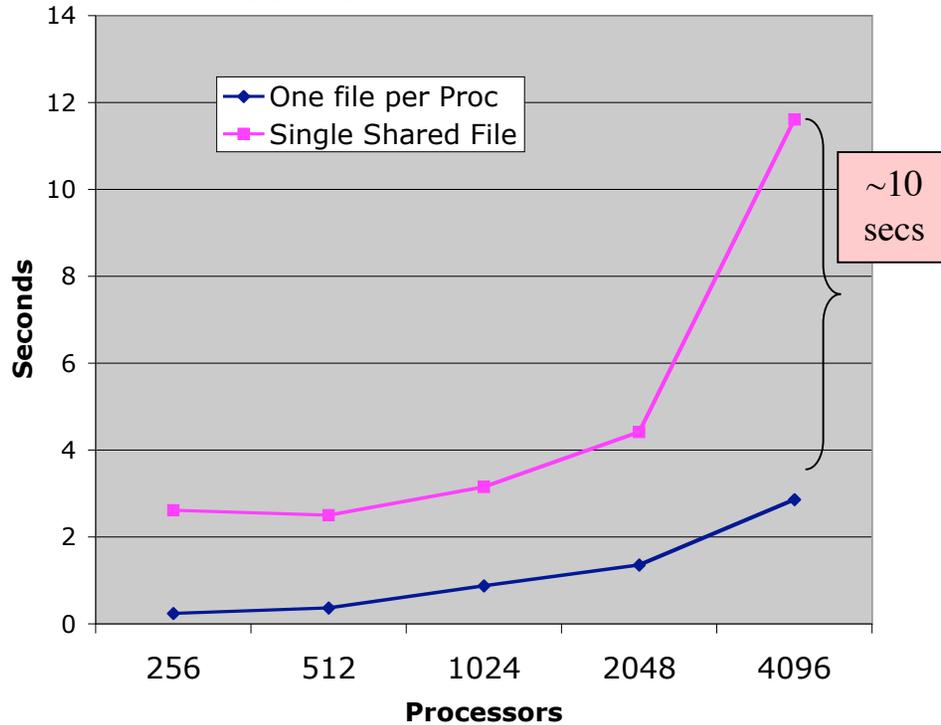


Difference in rates between one-file-per processor and shared files appears to be large

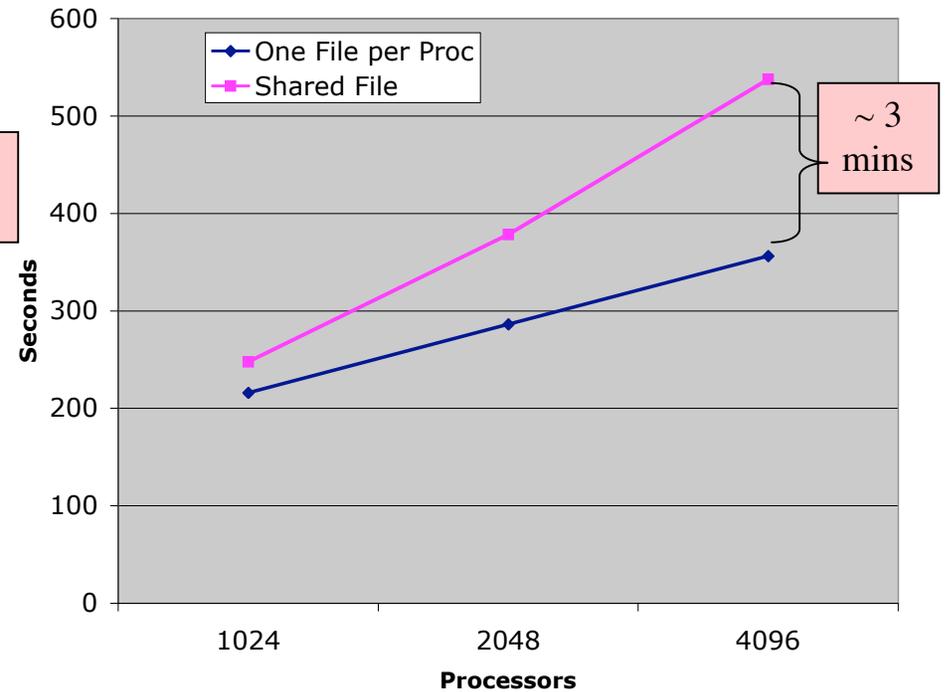


File Per Processor vs Shared File: Time

Aggregate File Size 1 GB



Aggregate File Size 1 TB



Looking closer at the difference in time results in a more complicated story. How much IO overhead can you afford?



Best Practices

- **Do large IO: write lots of data (1MB+) at once.**
 - Best for network performance and disk operations.
 - Also reduces MDS accesses.
- **Do parallel IO.**
 - Serial IO (single writer) can not take advantage of the system's parallel capabilities.
- **Stripe large files over many OSTs.**
- **Use a single, shared file instead of 1 file per writer, esp. at high parallel concurrency.**
- **If job uses > 2,000 tasks, reduce the number of tasks performing IO (experiment with this number)**
- **Use an IO library API and write flexible, portable programs.**