Using I/O on Cray XT Systems

Lonnie D. Crosby
Computational Scientist

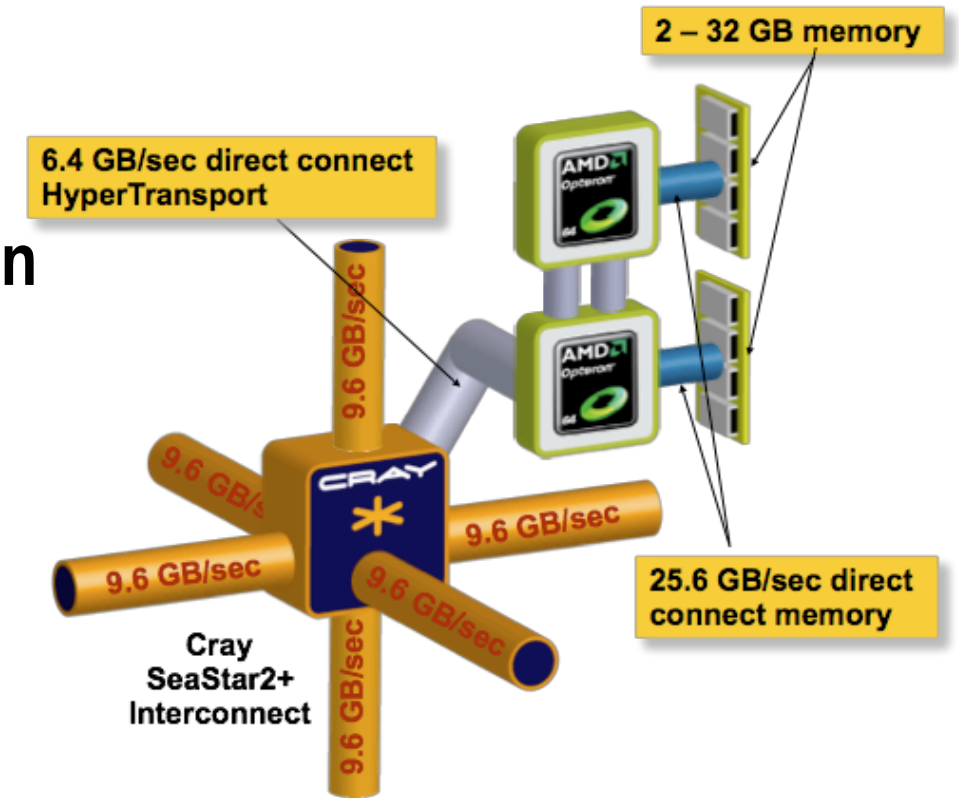
"NCCS/NICS Fall 2009 Cray XT5 Hex-core Workshop"

December 8, 2009

Application Performance

- Computation (FLOPs)
  - Processor
- Inter-process Communication
  - Interconnect
- Memory
  - Capacity and Speed
- I/O
  - File System

Cray XT5 Compute Node



Factors which affect I/O.

- I/O is simply data migration.
  - Memory ⟷ Disk
    - Layout
- Size of write/read operations
  - Bandwidth vs. Latency
- Data continuity and locality in memory and on disk
  - Bandwidth vs. Latency
- Number of processes performing I/O
  - I/O Pattern
- Characteristics of the file system
  - Distributed or Shared

Application I/O Patterns

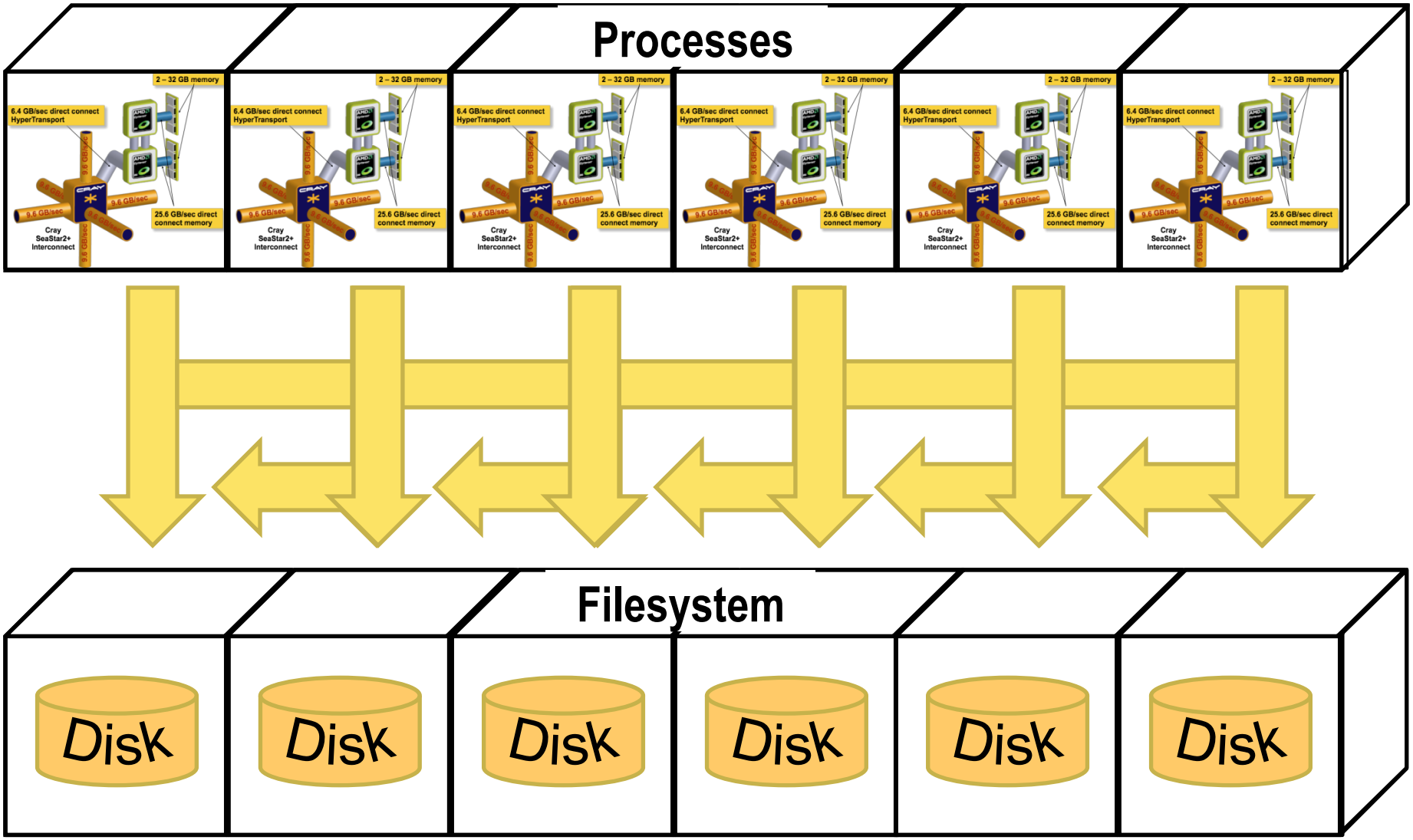
Serial I/O

- **Spokesperson**
  - One process performs I/O.

Parallel I/O

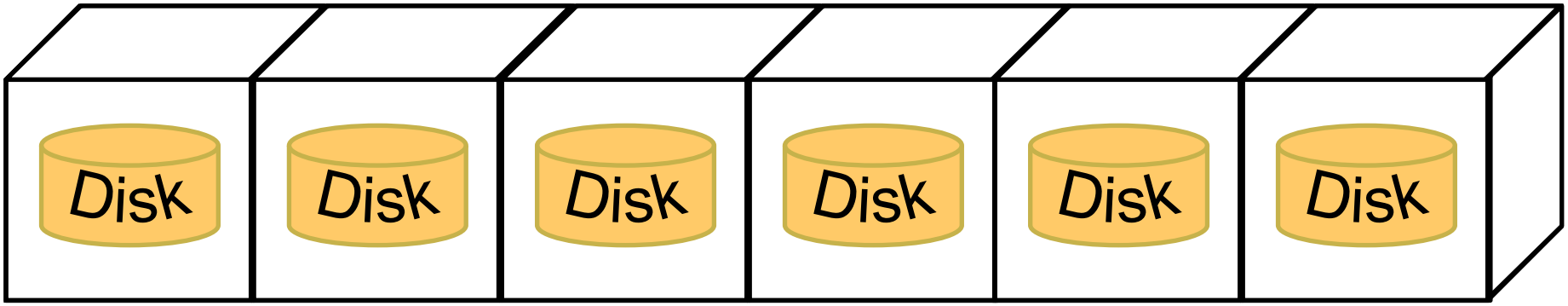
- **File per Process**
  - Each process performs I/O to a single file.
- **Single Shared File**
  - Each process collectively performs I/O to a single shared file.
- **Multiple Shared Files**
  - Groups of processes perform I/O to a single shared file.

Parallelism



Parallelism

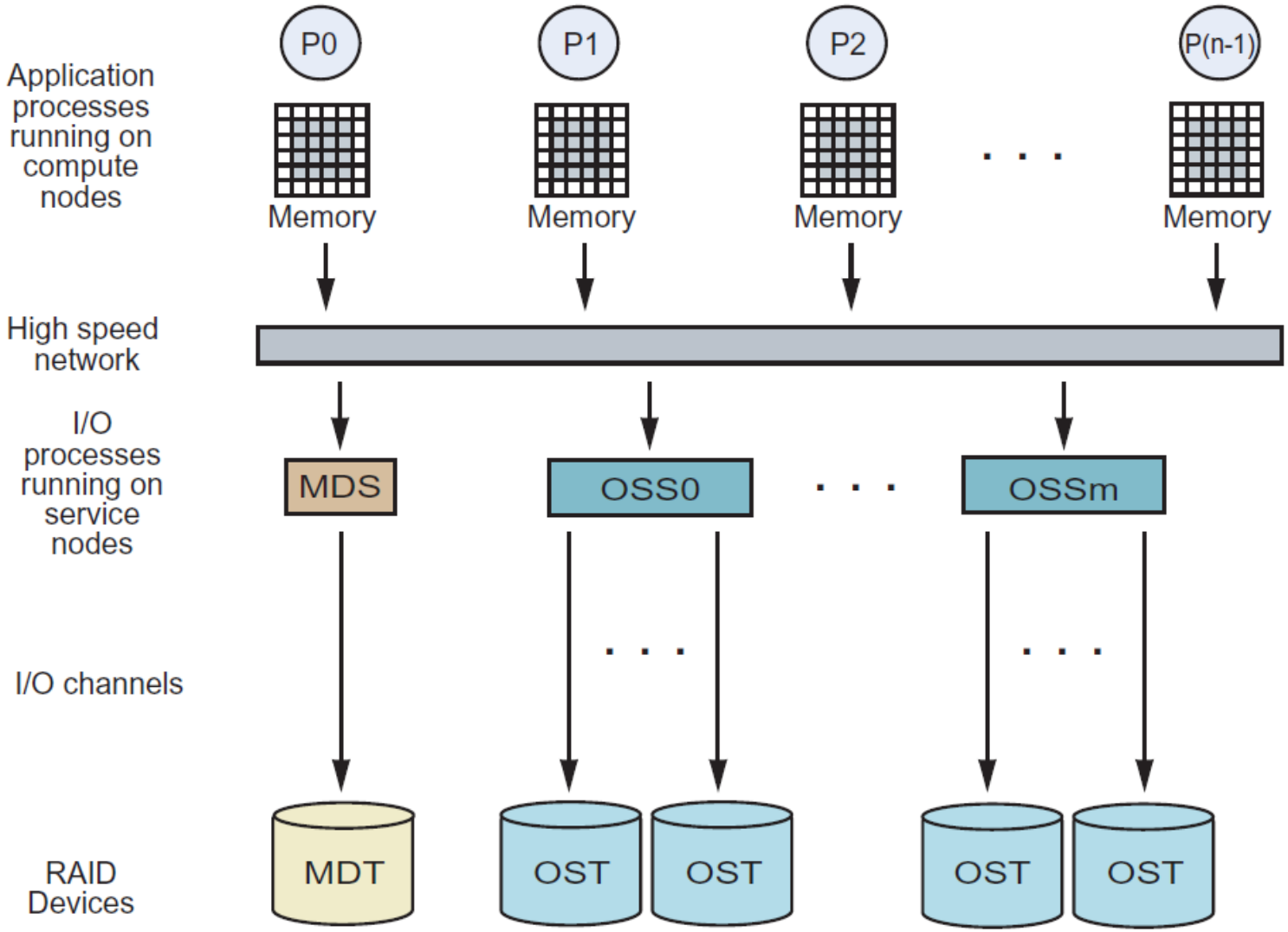
- Process level parallelism
  - MPI
  - IO Libraries (MPI-IO, HDF5, p-netCDF)
- File System parallelism
  - Distributed File System
  - Shared Parallel File System (Lustre)



Limits of I/O

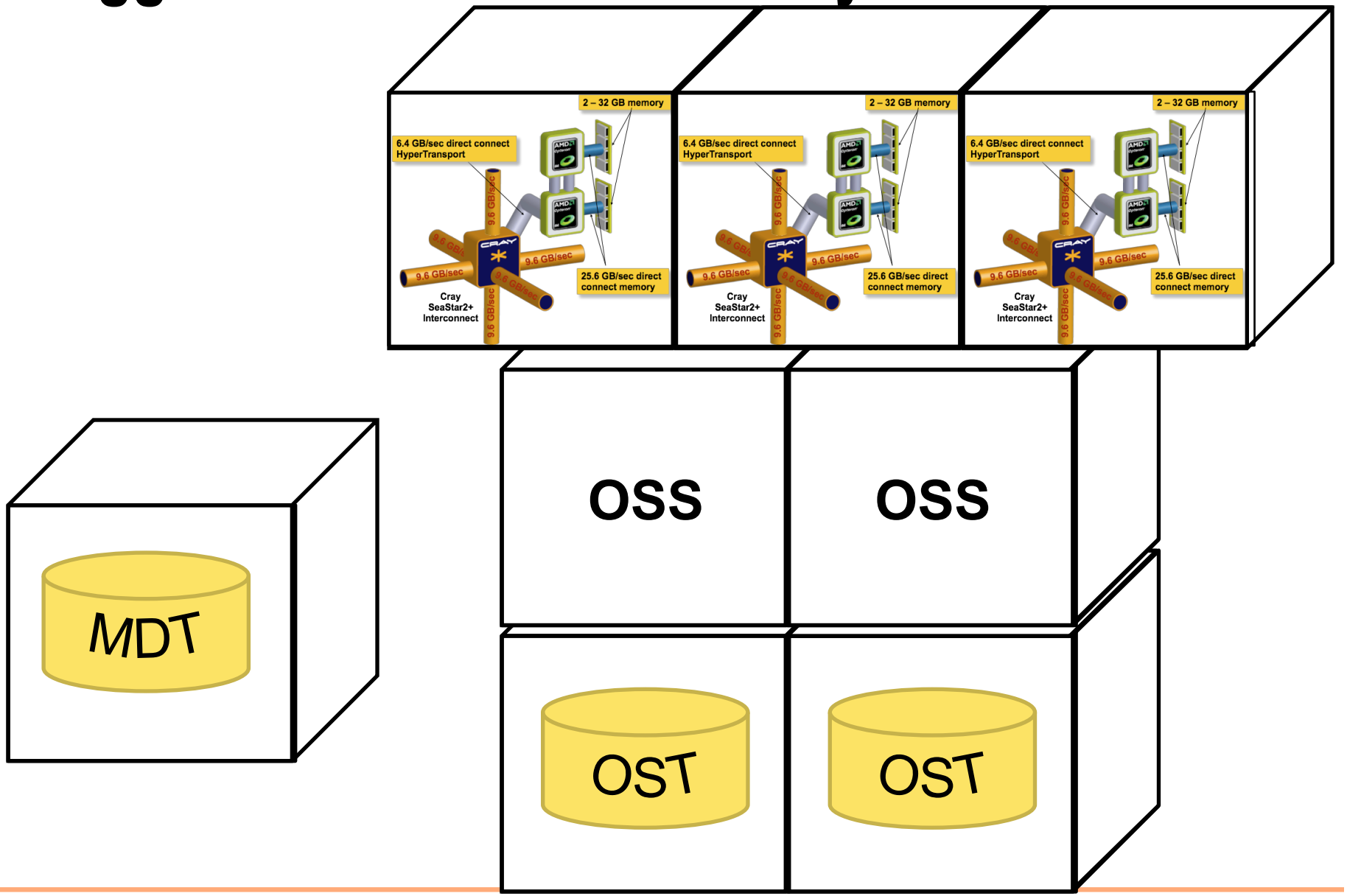
- Serial I/O
  - Is limited by the single process which performs I/O.
- Parallel I/O
  - Is limited by the number of disks which are concurrently utilized.
  - Contention for file system resources.
- Distributed File System
  - Files are localized on a single disk.
- Parallel File System
  - Files are localized on a single disk.
  - Files are striped across multiple disks.

A Bigger Picture: Lustre File System



©2009 Cray Inc.

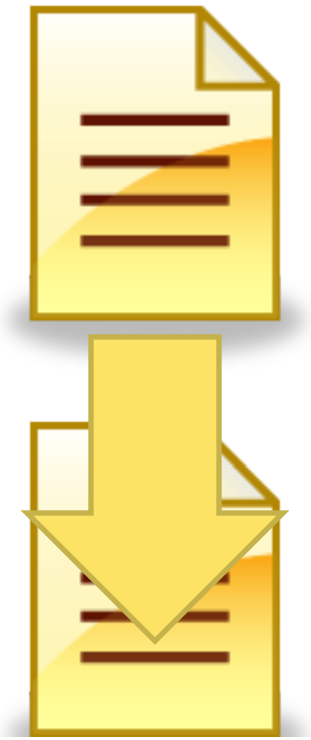
A Bigger Picture: Lustre File System



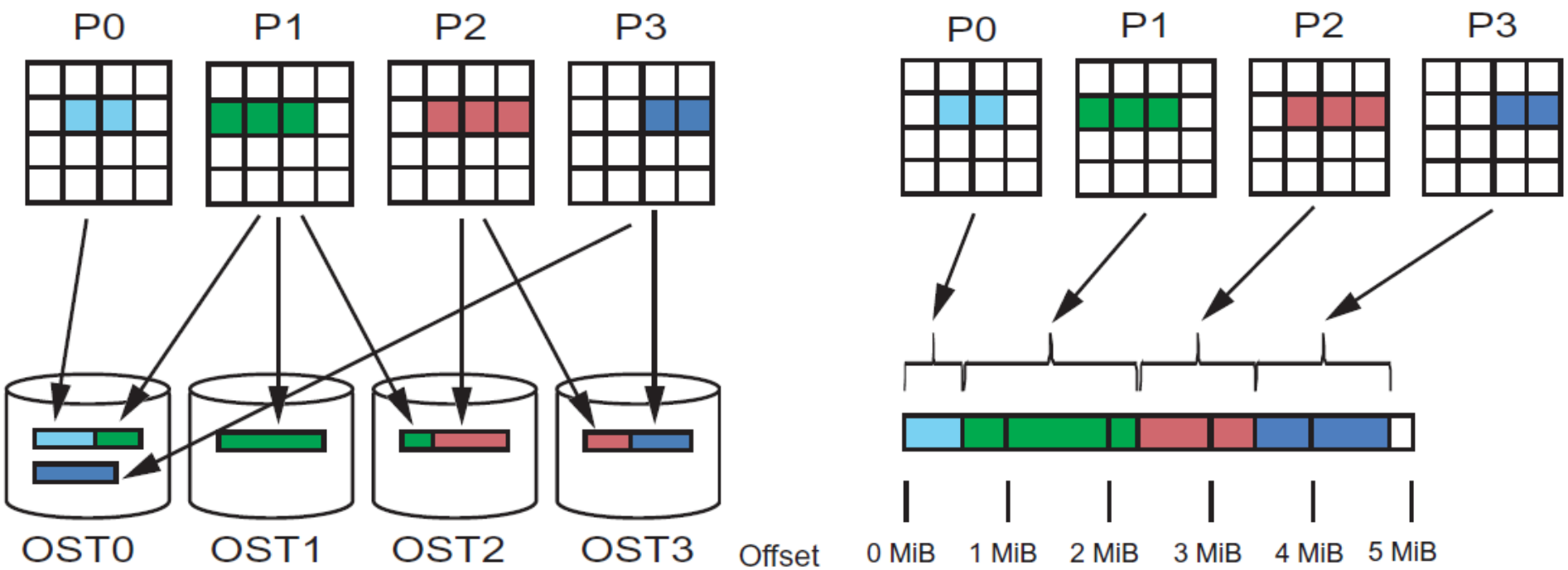
Lustre Striping: File Parallelism

- lfs setstripe
  - Stripe size -s (default: 1M)
  - Stripe count -c 5 (default 4, -1 All)
  - Stripe index -i 0 (default: -1 round robin)

  < file | directory >



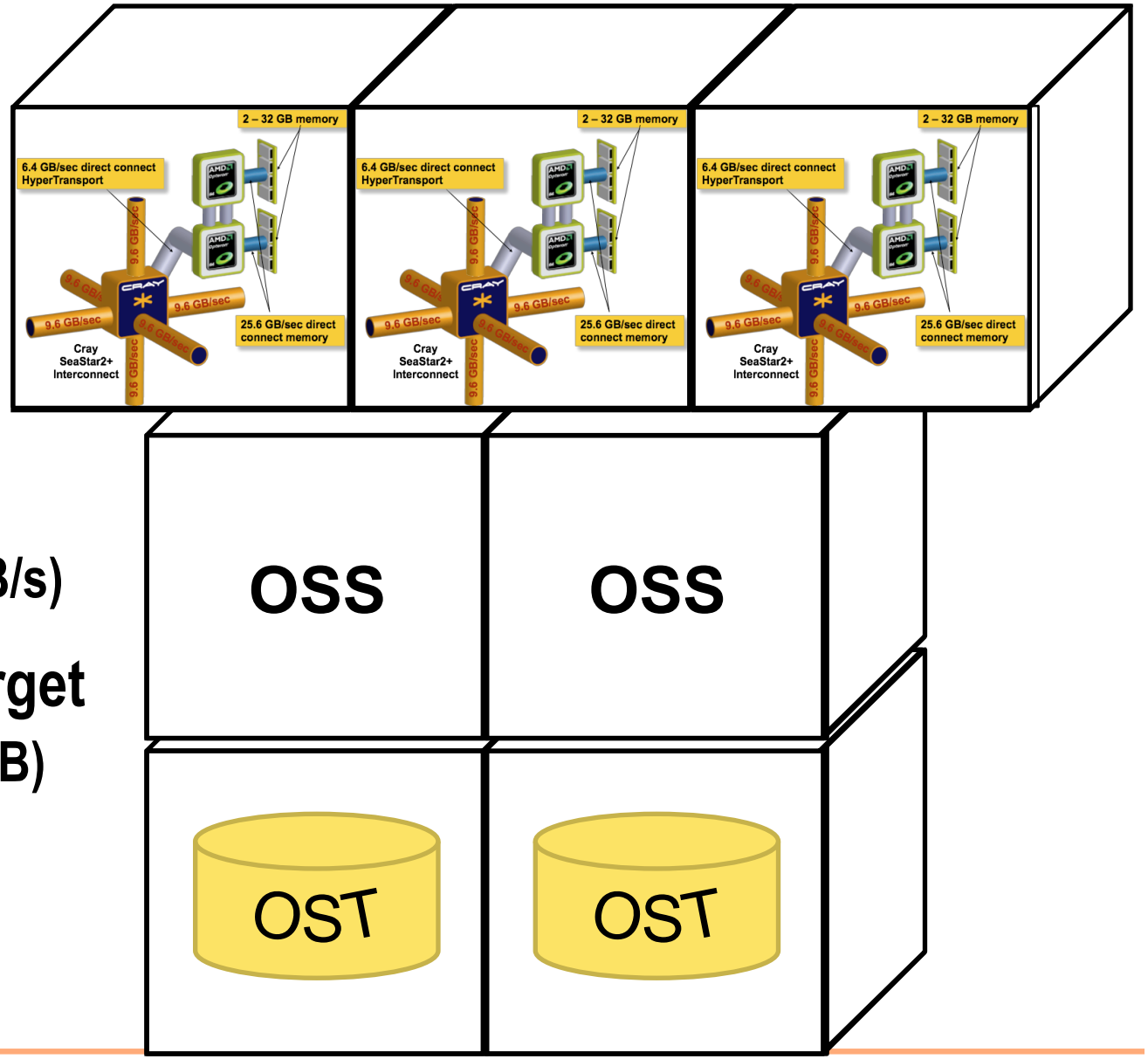
Lustre Striping: File Parallelism



©2009 Cray Inc.

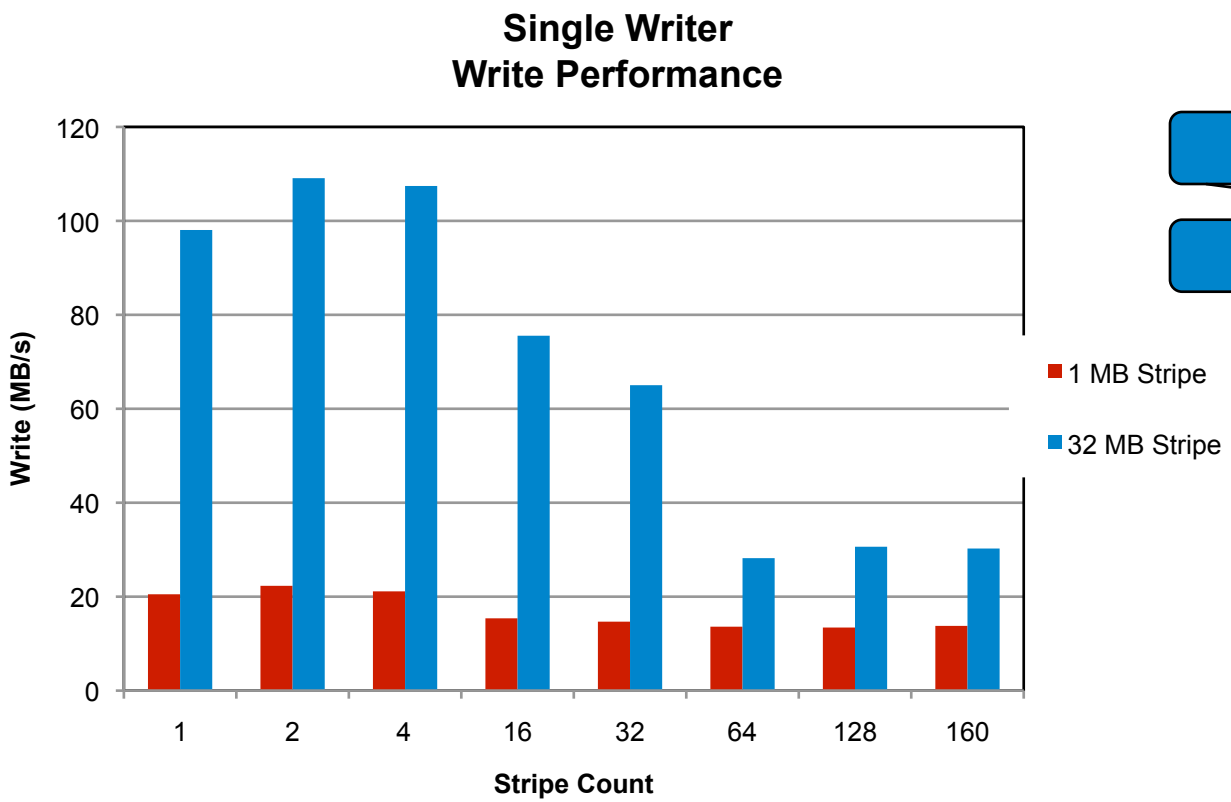
A Bigger Picture

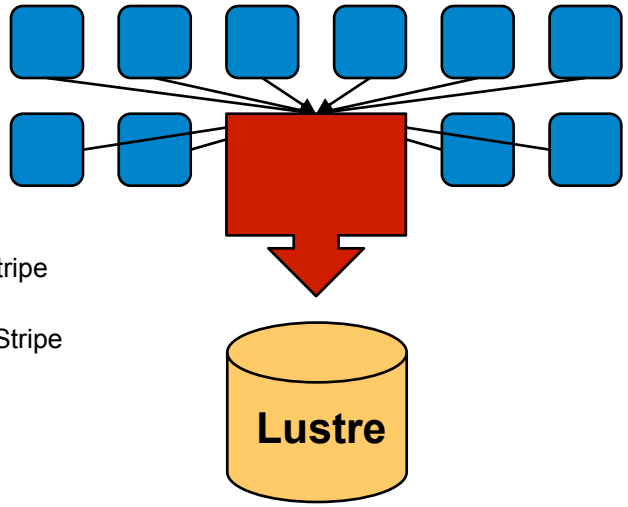
- Computational Nodes
  - Kraken: 8253
- Object Storage Server Nodes
  - Kraken: 48 (30 GB/s)
- Object Storage Target
  - Kraken: 336 (2.4 PB) [7.2 TB Disk]



Spokesperson – Serial I/O Importance of data locality

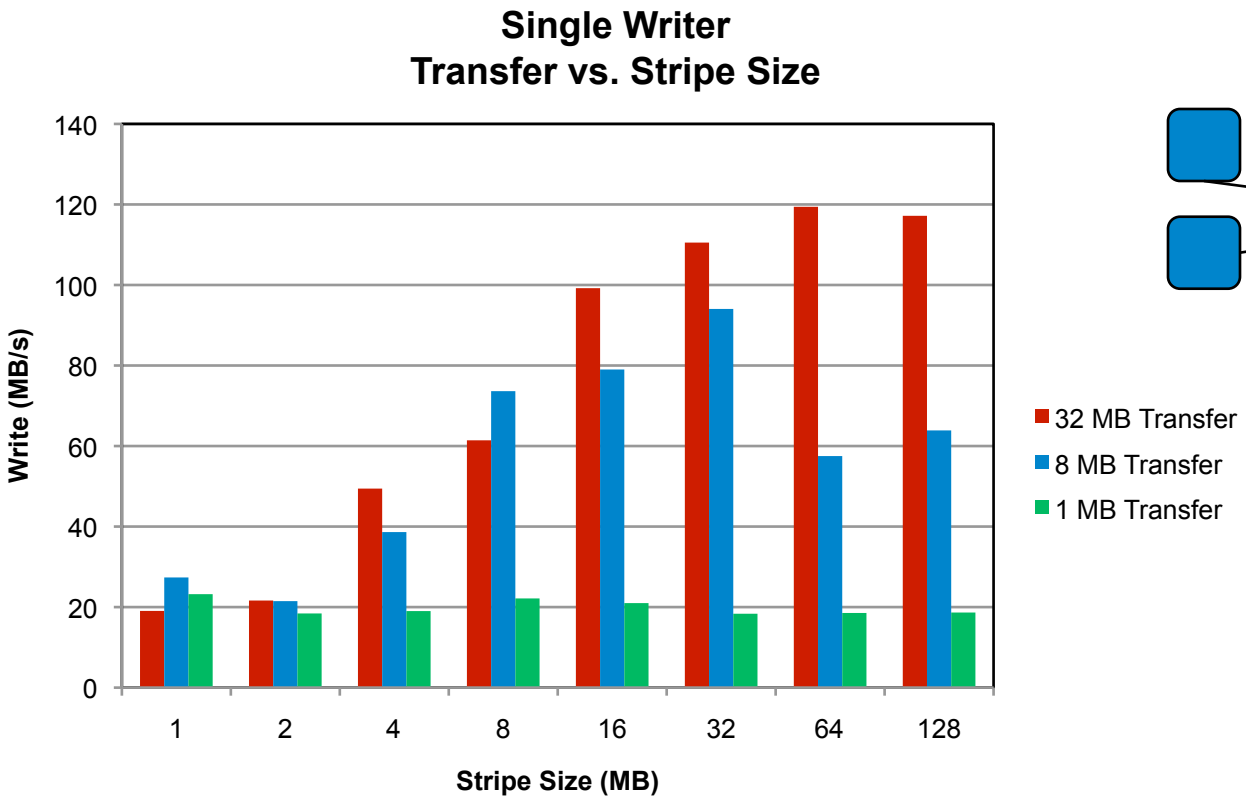
- 32 MB per OST (32 MB – 5 GB) and 32 MB Transfer Size

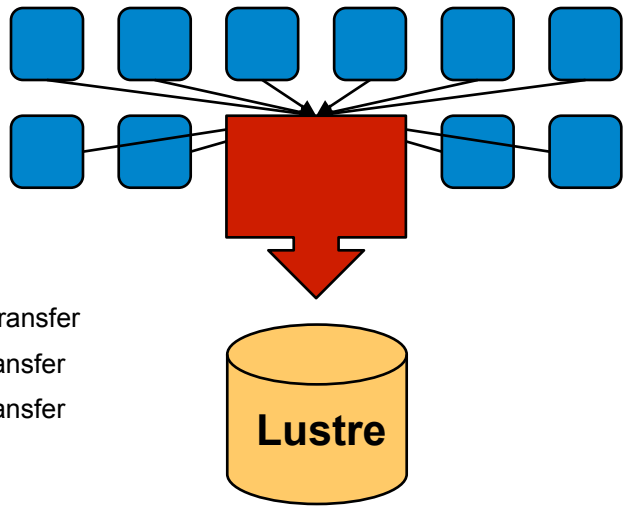




Spokesperson – Serial I/O Importance of data continuity

- Single OST, 256 MB File Size





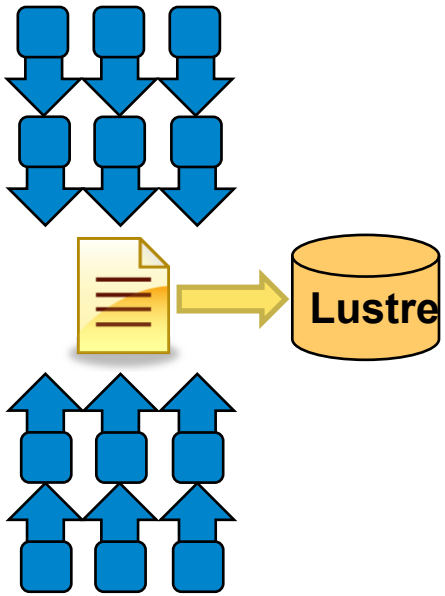
Data Locality and Continuity

- Data Locality
  - Performance is decreased when a single process accesses multiple disks.
  - Is limited by the single process which performs I/O.

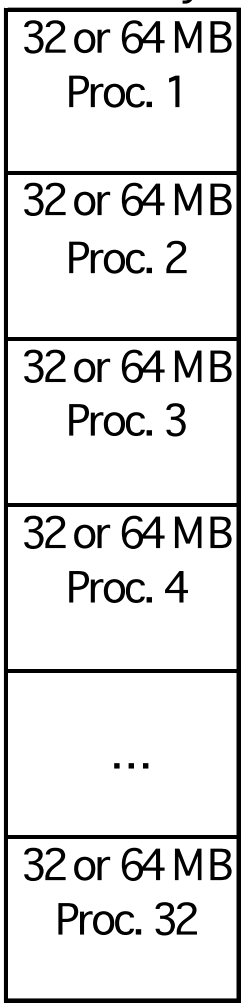
- Data Continuity
  - Larger read/write operations improve performance.
  - Larger stripe sizes improve performance (places data contiguously on disk).
  - Either may become a limiting factor.

Single Shared File

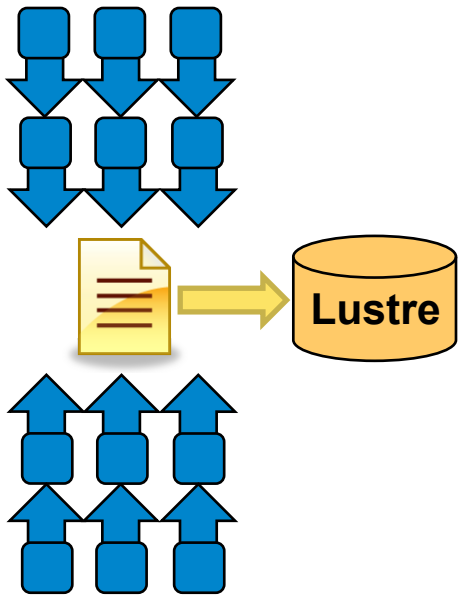
- Important Considerations
  - Data locality
  - Data continuity
- Parallel file Structure

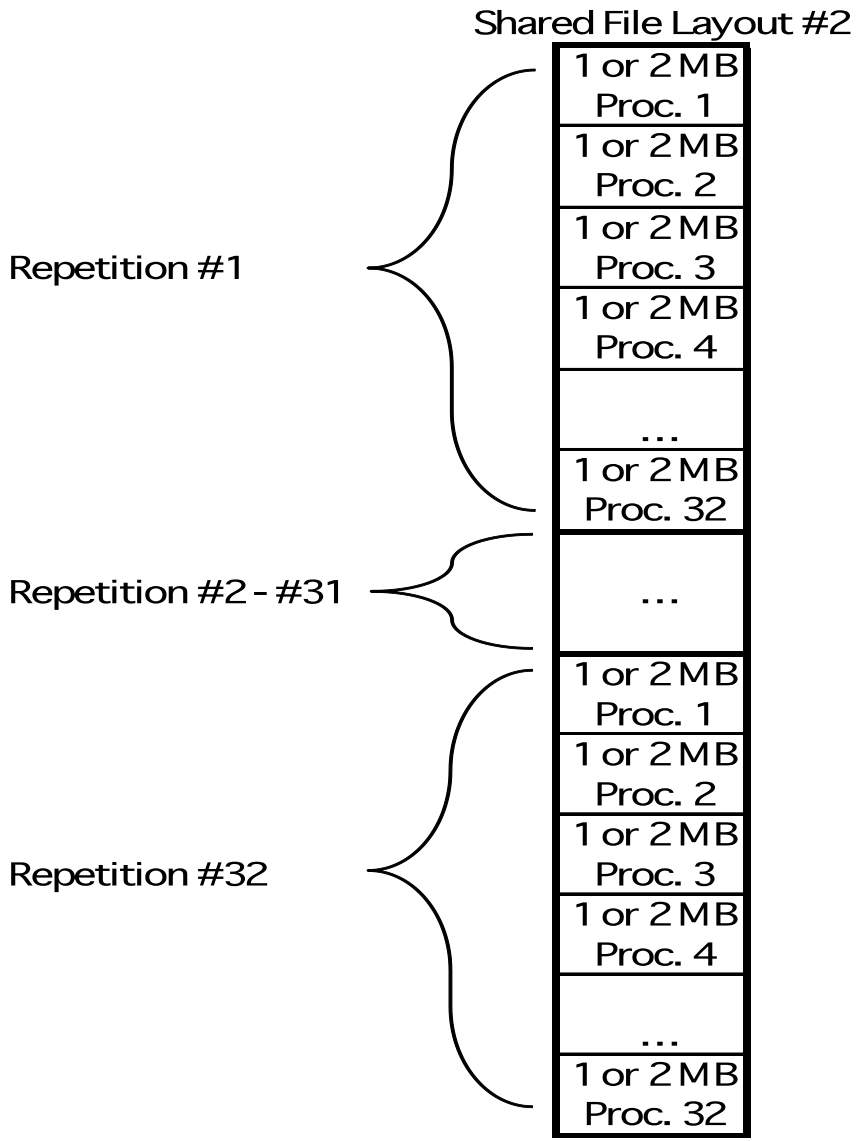


Shared File Layout #1

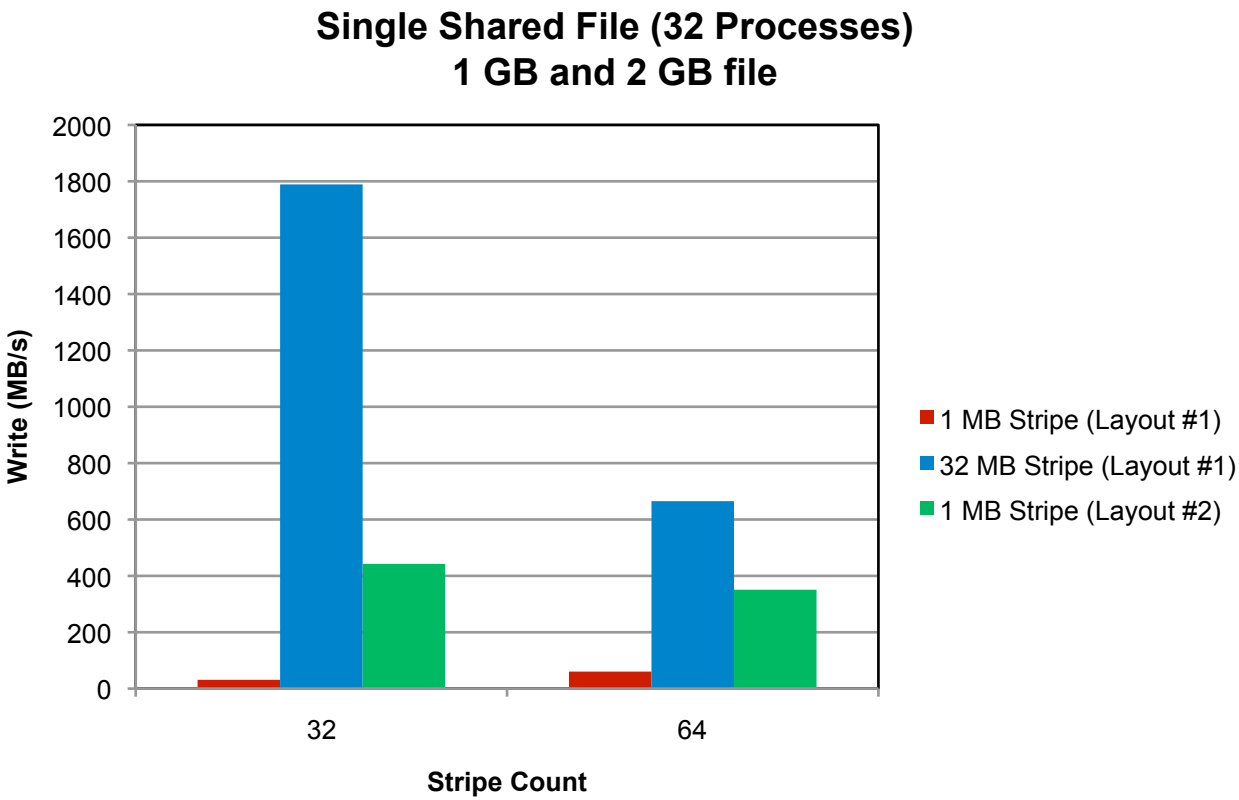


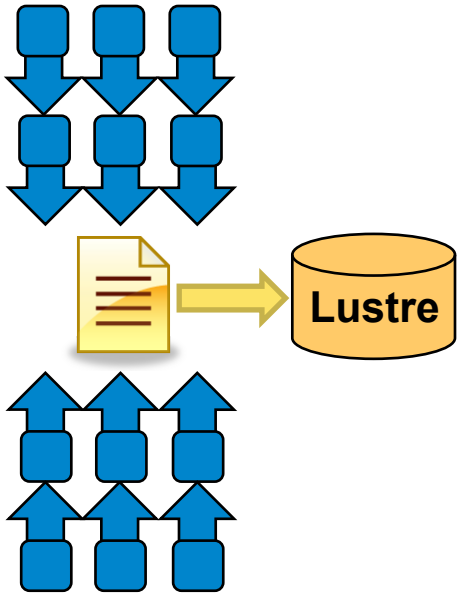
Single Shared File





Single Shared File





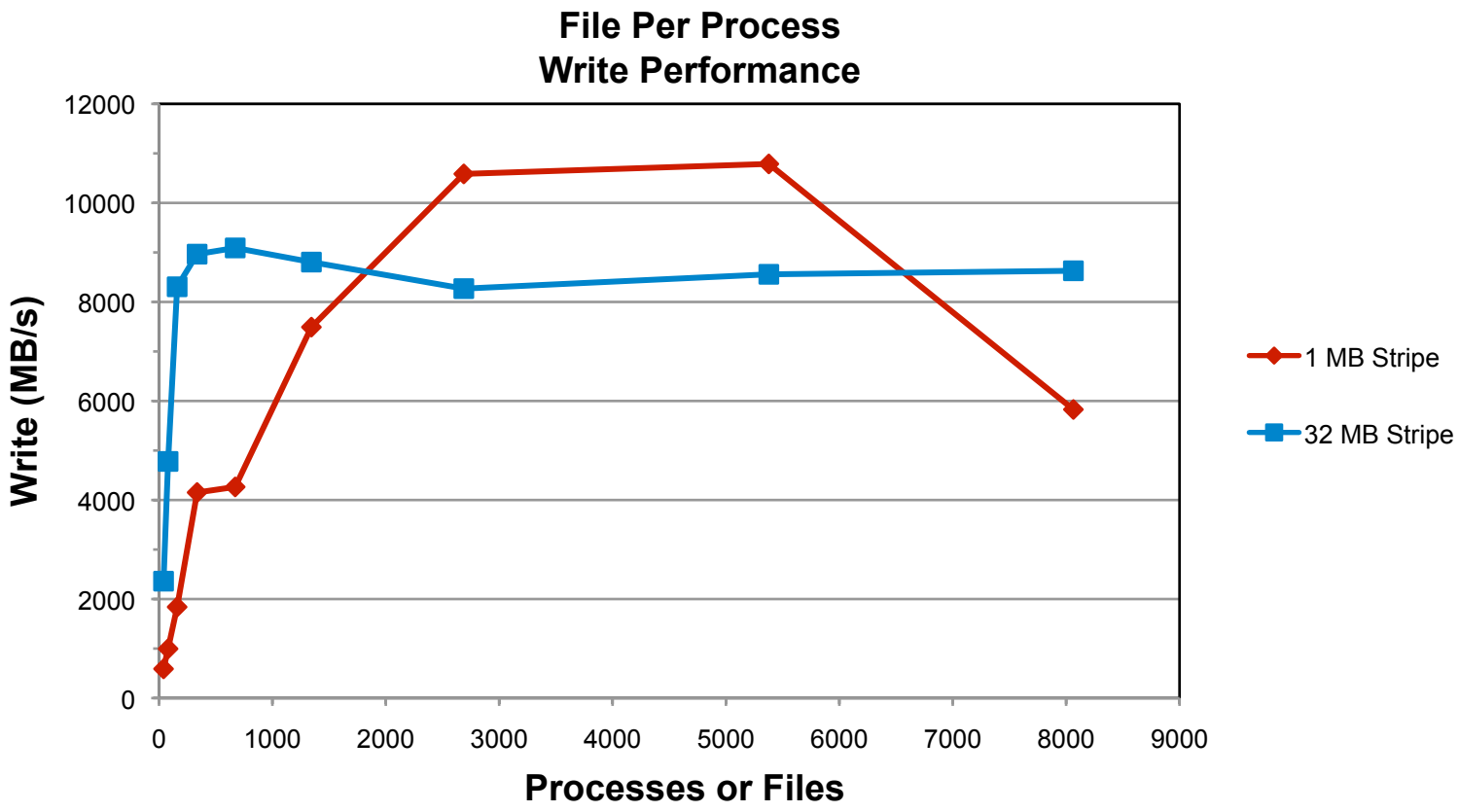
Data Locality and Continuity

- Data Locality
  - Performance is increased when portions of a shared file are localized on a single drive.
  - Contention is minimized.

- Data Continuity
  - Larger read/write operations improve performance.
  - Larger stripe sizes improve performance (places data contiguously on disk).
  - Either may become a limiting factor.

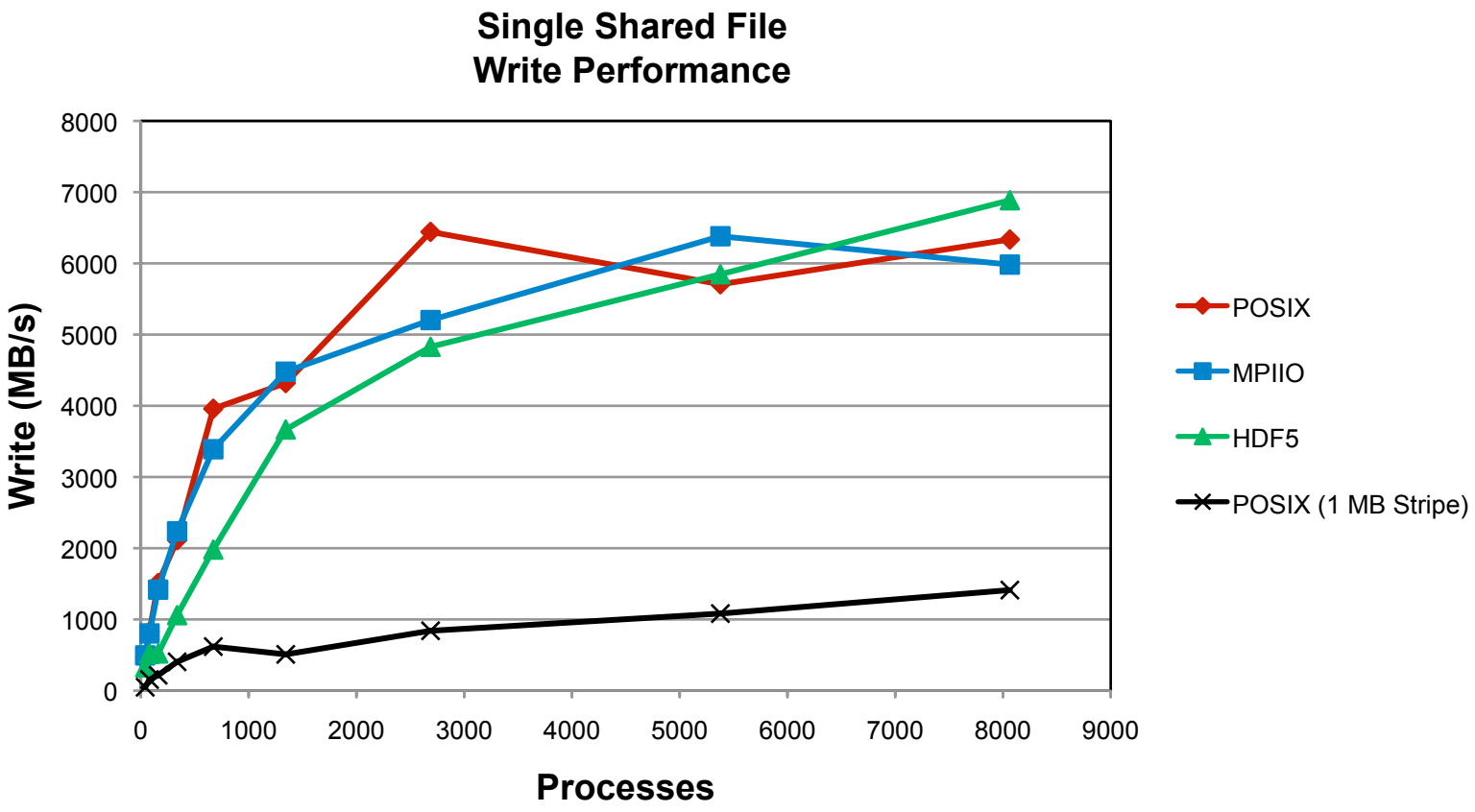
Scalability: File Per Process

- 128 MB per file and a 32 MB Transfer size



Scalability: Single Shared File

- 32 MB per process, 32 MB Transfer size and Stripe size

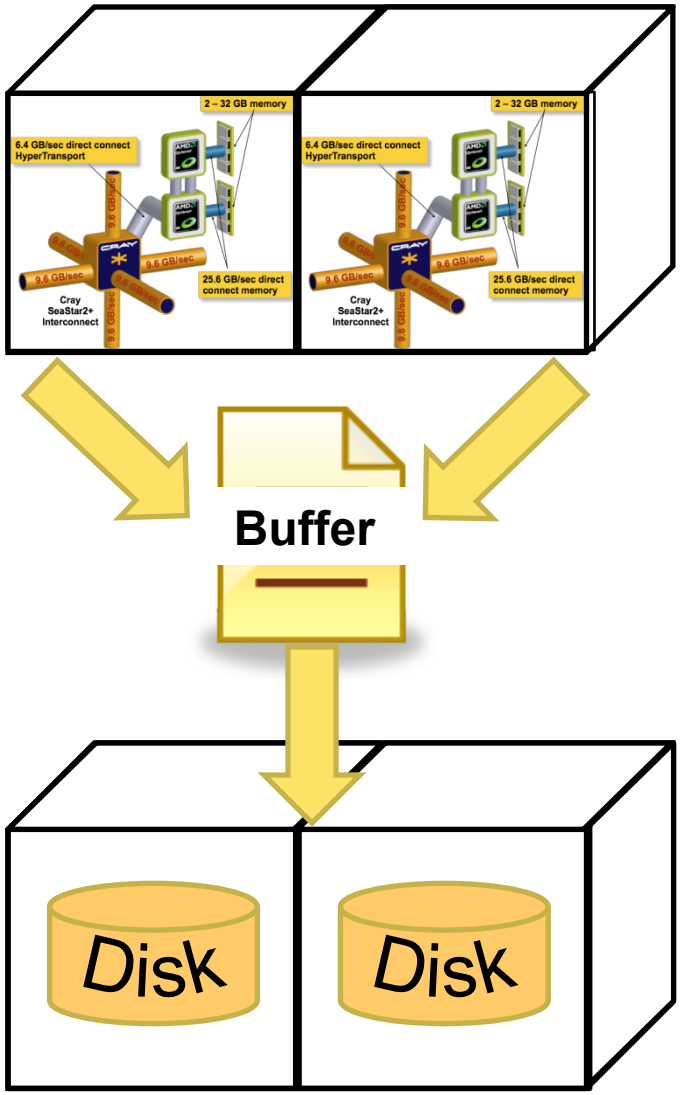


Scalability

- Serial I/O
  - Is not scalable. Limited by single process which performs I/O.
- File per Process
  - Limited at large process/file counts by:
    - Metadata Operations
    - FileSystem Contention
- Single Shared File
  - Limited at large process counts by filesystem contention.
  - File striping limitation of 160 OSTs in Lustre

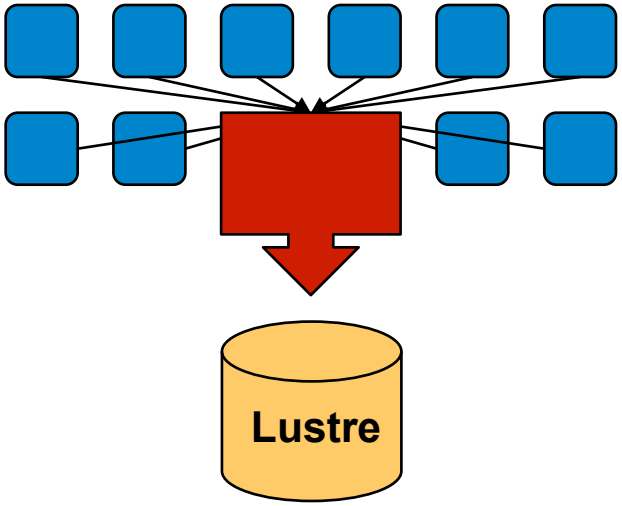
Buffered I/O

- Advantages
  - Aggregates smaller read/write operations into larger operations.
  - Examples: OS Kernel Buffer, MPI-IO Collective Buffering
- Disadvantages
  - Requires additional memory for the buffer.
  - Can tend to serialize I/O.
- Caution
  - Frequent buffer flushes can adversely affect performance.



Standard Output and Error

- Standard Ouput and Error streams are effectively serial I/O.
- Generally, the MPI launcher will aggregate these requests. (Example: mpirun, mpiexec, aprun, ibrun, etc..)
- Disable debugging messages when running in production mode.
  - "Hello, I'm task 32000!"
  - "Task 64000, made it through loop."



Binary Files and Endianess

- **Writing a big-endian binary file with compiler flag byteswapio**

```
File “XXXXXX"
                    Calls        Megabytes       Avg Size
  Open                  1
  Write           5918150      23071.28062           4088
  Close                 1
  Total           5918152      23071.28062           4088
```

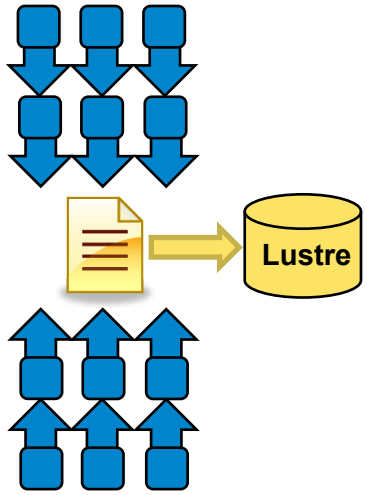
- **Writing a little-endian binary**

```
File “XXXXXX"
                    Calls        Megabytes       Avg Size
  Open                  1
  Write               350      23071.28062       69120000
  Close                 1
  Total               352      23071.28062       69120000
```

- **Can use more portable file formats such as HDF5, NetCDF, or MPI-IO.**

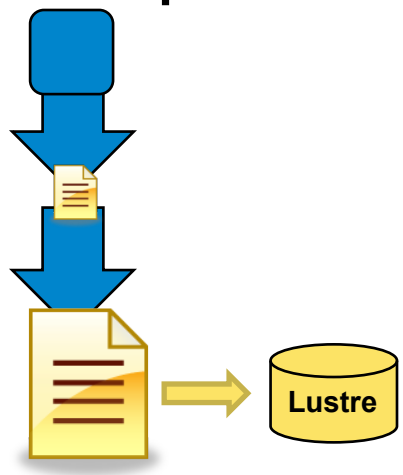
Case Study: Parallel I/O

- A particular code both reads and writes a 377 GB file. Runs on 6000 cores.
  - Total I/O volume (reads and writes) is 850 GB.
  - Utilizes parallel HDF5
- Default Stripe settings: count 4, size 1M, index -1.
  - 1800 s run time (~ 30 minutes)
- Stripe settings: count -1, size 1M, index -1.
  - 625 s run time (~ 10 minutes)
- Results
  - 66% decrease in run time.



Case Study: Buffered I/O

- A post processing application writes a 1GB file.
- This occurs from one writer, but occurs in many small write operations.
  - Takes 1080 s (~ 18 minutes) to complete.
- IOBUF was utilized to intercept these writes with 64 MB buffers.
  - Takes 4.5 s to complete. A 99.6% reduction in time.




```
File "ssef_cn_2008052600f000"
                  Calls          Seconds        Megabytes   Megabytes/sec   Avg Size
  Open                1         0.001119
  Read              217         0.247026         0.105957        0.428931        512
  Write         2083634         1.453222      1017.398927      700.098632        512
  Close               1         0.220755
  Total         2083853         1.922122      1017.504884      529.365466        512
  Sys Read            6         0.655251       384.000000      586.035160   67108864
  Sys Write          17         3.848807      1081.145508      280.904052   66686072
  Buffers used            4  (256 MB)
  Prefetches              6
  Preflushes             15
```

IOBUF -Beta Library

- module load iobuf/beta
- Relink application with the Cray wrappers (ftn, cc, CC)
- Controlled by environmental variable at runtime.
  - setenv IOBUF_PARAMS '*:verbose'
  - man iobuf for more information
- Intercepts standard I/O calls. May not operate with the use of I/O libraries such as netcdf.

MPI-IO Usage

- Included in the Cray MPT library.
- Environmental variable used to help MPI-IO optimize I/O performance.
  - setenv MPICH_MPIIO_HINTS
  - man mpi for more information
- If given appropriate information (stripe count, size) can choose aggregators in collective operations that are Lustre stripe aligned. (collective buffering).

Conclusions

- Serial I/O
  - For a single process is limited by the single I/O stream.
  - For a file-per-process pattern limitation is due to simultaneous metadata operations (file open) at large core counts. Additionally, increasing contention for file system resources can adversely affect performance.
- Parallel I/O
  - For a single shared file limitation is due to file system contention at large core counts. Lustre limitation of 160 OSTs per file.
  - MPI-IO can be utilized to minimize file system contention at large core counts by utilizing collective buffering and appropriate hints.
- Lustre
  - Appropriate stripe settings should be utilized to minimize file system contention.

References

- Lustre File System – White Paper October 2008
  - http://www.sun.com/software/products/lustre/docs/lustrefilesystem_wp.pdf
- Introduction to HDF5
  - http://www.hdfgroup.org/HDF5/doc/H5.intro.html
- The NetCDF Tutorial
  - http://www.unidata.ucar.edu/software/netcdf/docs/netcdf-tutorial.pdf