Best Practices for Reading and Writing Data on HPC Systems

Katie Antypas
NERSC User Services
Lawrence Berkeley National Lab
NUG Meeting
15 February 2013
In this tutorial you will learn about I/O on HPC systems from the storage layer to the application.

Storage devices and parallel file systems

Use cases and best practices

Application I/O profiling
Layers between application and physical disk must translate molecules, atoms, particles and grid cells into bits of data.
Despite limitations, magnetic hard drives remain the storage media of choice for scientific applications.

Source: Popular Science, Wikimedia Commons
The delay before the first byte is read from a disk is called “access time” and is a significant barrier to performance.

- $T(\text{access}) = T(\text{seek}) + T(\text{latency})$
- $T(\text{seek}) = \text{move head to correct track}$
- $T(\text{latency}) = \text{rotate to correct sector}$
- $T(\text{seek}) = 10 \text{ milli-sec}$
- $T(\text{latency}) = 4.2 \text{ milli-sec}$
- $T(\text{access}) = 14 \text{ milli-sec}$

\[\text{~100 Million flops in the time it takes to access disk}\]
Clearly a single magnetic hard drive cannot support a supercomputer, so we put many of them together.

Disks are added in parallel in a format called “RAID”.

Redundant Array of Independent Disks
Flash or NVRAM could provide a better balance between bandwidth and storage that could benefit HPC applications

**Questions**

- What are the use cases for NVRAM in a supercomputer?
- How much of the NERSC workload could benefit from NVRAM in an HPC system?
- What are the cost/benefit trade-offs in terms of disk capacity, I/O bandwidth and compute?
- At what level should NVRAM sit? (compute node? I/O node? Disk?)
- What software development would be needed to allow productive use of NVRAM on NERSC-8?
A high performing parallel file system efficiently manages concurrent file access and scales to support huge HPC systems.
What’s the best file system for your application to use on Hopper?

<table>
<thead>
<tr>
<th></th>
<th>PEAK</th>
<th>PURPOSE</th>
<th>PROS</th>
<th>CONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$HOME</td>
<td>Low</td>
<td>Store application code, compile files</td>
<td>Backed up, not purged</td>
<td>Low performing; Low quota</td>
</tr>
<tr>
<td>$SCRATCH/</td>
<td>35 GB/sec</td>
<td>Large temporary files, checkpoints</td>
<td>Highest performing</td>
<td>Data not available on other NERSC systems</td>
</tr>
<tr>
<td>$SCRATCH2</td>
<td></td>
<td></td>
<td></td>
<td>Purged</td>
</tr>
<tr>
<td>$PROJECT</td>
<td>15GB/sec</td>
<td>For groups needing shared data access</td>
<td>Data available on all NERSC systems</td>
<td>Shared file performance</td>
</tr>
<tr>
<td>$GSCRATCH</td>
<td>12 GB/sec</td>
<td>Alternative scratch space</td>
<td>Data available on almost all NERSC systems</td>
<td>Shared file performance Purged</td>
</tr>
</tbody>
</table>
Serial I/O and file-per-processor I/O are used by many NERSC users
Shared file IO leads to better data management and more natural data layout for scientific applications.
MPI/IO and high level parallel I/O libraries like HDF5 allow users to write shared files with a simple interface.

This talk doesn’t give details on using MPI-IO or HDF5. See online tutorials.
Shared file I/O depends on the file system, MPI-IO layer and the data access pattern

IO Test using IOR benchmark on 576 cores on Hopper with Lustre file system

ior write performance mpi-io vs posix file per processor

ior read performance mpi-io vs posix file per processor

Hard sell to users
We initiated a collaboration with Cray to improve MPI-IO performance on Hopper to /project and /gscratch.

Consider combining smaller write requests into larger ones and limiting the number of writers per node.
On Hopper, read and write performance do not scale linearly with the number of tasks per node.

Consider combining smaller write requests into larger ones and limiting the number of writers per node.

Progress over time ...

- DVS_Max Node=14
- Match DVS Block size 4MB
- Turn Off CB
- Best we get so far

Baseline (Jan) 0.8 1.8
Jan 26 5 0.05
Feb 8 5 0.5
Feb 10 5 1.5
Feb 22 6 3.5
Mar 7 7.8 5

Wrong DVS Block size
New MPIIO Setting Correct Defaults

File/Process 12 11
On Hopper, read and write performance do not scale linearly with the number of tasks per node.

Consider combining smaller write requests into larger ones and limiting the number of writers per node.

- 10X performance improvement on read, 3X write, after changing both run setup and MPIIO library.
- Setting DEFAULT values for users so that they can get best performance (in most cases) automatically.
Files are broken up into lock units, which the file system uses to manage concurrent access to a region in a file.

Processor A

Can I write?

File

Lock unit, typically 1MB

Processor B

Processes request access to a file region

Can I write?

Can I write?
Files are broken up into lock units, which the file system uses to manage concurrent access to a region in a file.

Processors request access to a file region.

Can I write?

Yes!

No!
Users can (and should) adjust striping parameters on Lustre file systems.

User controlled striping on Lustre filesystems, $SCRATCH, $SCRATCH2

NO User controlled striping on GPFS filesystems, $GSCRATCH, $HOME, $PROJECT

No user controlled striping, (ALL GPFS filesystems)
There are three parameters that characterize striping on a Lustre file system, the stripe count, stripe size and the offset.

- **Stripe count**: Number of OSTs a file is split across: Default 2
- **Stripe size**: Number of bytes to write on each OST before cycling to the next OST: Default 1MB
- **OST offset**: Indicates the starting OST: Default round robin
Striping can be set at the file or directory level.

When striping set on a directory: all files created in that directory will inherit striping set on the directory.

```
    lfs setstripe <directory|file> -c stripe-count
```

Stripe count - # of OSTs file is split across

Example: change stripe count to 10
```
    lfs setstripe mydirectory -c 10
```
For one-file-per-processor workloads set the stripe count to 1 for maximum bandwidth and minimal contention.
Striping guidelines for Hopper

• One File-Per-Processor I/O or shared files < 10 GB
  – Keep default or stripe count 1

• Medium shared files: 10GB – 100sGB
  – Set stripe count ~4-20

• Large shared files > 1TB
  – Set stripe count to 20 or higher, maybe all OSTs?

• You’ll have to experiment a little
I/O resources are shared between all users on the system so you may often see some variability in performance.
You can learn more about your application’s I/O pattern using an I/O profiling tool like Darshan.

- How to use Darshan
- Compile your application as you normally would
- Run your application as you normally would
- Look at NERSC’s ‘Completed Jobs’ page for I/O statistics
On the completed jobs webpage you will see output like the below image

**IO Summary from Darshan**

<table>
<thead>
<tr>
<th>Exec. Runtime</th>
<th>MB Read</th>
<th>MB Written</th>
<th>Read Time (s)</th>
<th>Write Time (s)</th>
<th>Read Rate (MB/s)</th>
<th>Write Rate (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12-13 04:03:39 - 12-13 17:21:39</td>
<td>16203033.3</td>
<td>314607.21</td>
<td>1.29026e+06</td>
<td>510309</td>
<td>12.56</td>
<td>0.62</td>
</tr>
</tbody>
</table>

- When an application is compiled against the Darshan library, I/O calls are intercepted and recorded in a central logfile
- Examine distribution of write sizes
- Measure I/O rates
Comparing the I/O patterns of two users

User 1

Read

User 2

Read

Distribution of block sizes
In summary, think about the big picture in terms of your simulation, output, and visualization needs.

Determine your I/O priorities: Performance? Data Portability? Ease of analysis?

Write large blocks of I/O

Understand the type of file system you are using and make local modifications.
THE END
A simulation writing a shared file, with a stripe count of 2, will achieve a maximum write performance of ~800 MB/sec.

No matter how many processors are used in the simulation.

For large shared files, increase the stripe count.
Striping over all OSTS increases bandwidth available to application

The next table gives guidelines on setting the stripe count
Lustre file system on Hopper

Hopper With Genimi Network

Infiniband

Fiber Channel

SCRATCH1/2

Note: SCRATCH1 and SCRATCH2 have identical configurations.
Disk rates are improving, but not nearly as fast as compute performance.

In 1956 IBM produced the first computer to include a disk drive.

The rate of performance improvement in supercomputing systems, as measured by Linpack, since 1993.

Source: R. Freitas of IBM Almaden Research Center
A file system is a software layer between the operating system and storage device

- Files
- Directories
- Access permissions

Source: J.M. May “Parallel IO for High Performance Computing, techCrunch, howstuffworks.com
File striping is a technique used to increase I/O performance by simultaneously writing/reading data from multiple disks.