State of the Art I/O Tools

EPCC

February 28, 2018

Elsa Gonsiorowski

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC
Outline

Motivating Example
Questions from Applications

Measuring I/O Performance
MACSio

The I/O Stack

Burst Buffer Technologies
SCR and Performance Portability

Additional Projects
IO-500
Simulation Output

10  20  30  40  50
Simulation Output
Simulation Output

I/O Performance hasn’t changed
As computation performance increases, I/O must be re-evaluated.
Questions from Applications

1. Where do we fall in the I/O envelope?
2. Parameters to achieve best performance?
3. How do we best use new storage tiers?
Where do we fall in the I/O Envelope?

Given:

- Peak system I/O performance
- Current application performance
- I/O pattern or trace
- ... other details?

Answer:

- Where is the application losing performance?
- What will gains can be made?
Where do we fall in the I/O Envelope?

Current Examples

- Use IOR and mdtest to measure peak system performance
- I/O Specific proxy application
- Lots of work
Where do we fall in the I/O Envelope?

Unposed Questions

- What is the point of this I/O?
- Could this use-case be achieved in a more efficient way?
- How do we enable in-situ or co-situ processes?

High-level questions
Parameters to achieve best performance?

Given:

- Tuning of peak performing benchmark
- Current application I/O

Answer:

- What file system settings need to be tuned?
- Is metadata a bottleneck / file locking?
Parameters to achieve best performance?

Current Examples

- None.
- Validation of simulation models with counters, no analysis of real applications.
Parameters to achieve best performance?

Unposed Questions

- Can any of this be detected at a lower level?
- Automatic tuning of the file system during a workload
- How can this drive future procurements?

Lower level and inter-level questions
How do we best use new Storage Tiers?

Given:
- Scientific need
- System limitations

Answer:
- Which I/O patterns perform best
- Resiliency models
How do we best use new Storage Tiers?

Current Examples

- Defensive I/O Assumption
  - Optimal checkpoint interval
  - SCR with system-specific configuration
- Lossy compressions
  - HDF5 ZFP Compression
How do we best use new Storage Tiers?

Unposed Questions

- Interactions between resource schedulers and application
  - pre-stage / post-stage
  - dynamic job allocation resources
- What is the scientific need? How much precision is needed?
- Work flows to manage data movement

Questions requiring full-stack knowledge
Measuring I/O Performance

- Benchmarking
- Profiling
- Proxy Applications
Benchmarking

- IOR
- mdtest
- benchio
- IO_Bench
- MPI Tile IO
- b_eff_io
- SPIOBENCH
- iozone
- MADbench2

Mainly testing POSIX interface, with some MPI-IO.
Profiling

- Darshan
- Vampir
Proxy Applications

- MACSio
- HACC_IO / GenericIO
High Level-of-Abstraction

- Application-level I/O
- Utilize multiple layers of I/O middlewares
- Representative mesh data
Tunable I/O Patterns

- File-per-process
- Single shared file
- Middle ground: M files to N processes
Plug-in based Architecture

- HDF5
- netCDF
- SILO
- TyphoonIO
- ADIOS coming soon
The I/O Stack

<table>
<thead>
<tr>
<th>Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
</tr>
<tr>
<td>I/O Middleware and Libraries</td>
</tr>
<tr>
<td>Lustre Client</td>
</tr>
<tr>
<td>Linux VFS</td>
</tr>
<tr>
<td>ZFS</td>
</tr>
<tr>
<td>Buffer Cache</td>
</tr>
<tr>
<td>I/O Scheduler</td>
</tr>
<tr>
<td>RAID Z</td>
</tr>
<tr>
<td>HDD</td>
</tr>
</tbody>
</table>

Courtesy of John Bent
The I/O Stack

- Application
  - HDF5
  - netCDF
  - SILO
  - MPIIO
  - POSIX

- Lustre Client
- Linux VFS
- ZFS
- Buffer Cache
- I/O Scheduler
- RAID Z
- HDD
The I/O Stack

Application

I/O Middleware and Libraries

Lustre Client

Linux VFS

ZFS

Buffer Cache

I/O Scheduler

RAID Z

HDD
The I/O Stack

- Application
- I/O Middleware and Libraries
  - Lustre Client
  - Linux VFS
  - ZFS
  - Buffer Cache
  - I/O Scheduler
  - RAID Z
  - HDD
# Burst Buffer Technologies

<table>
<thead>
<tr>
<th>Type</th>
<th>Technology</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Local</td>
<td>IBM BBAPI</td>
<td>LLNL (Sierra)</td>
</tr>
<tr>
<td>Machine Global</td>
<td>Cray Datawarp</td>
<td>LANL (Trinity)</td>
</tr>
</tbody>
</table>
## Burst Buffer Technologies

<table>
<thead>
<tr>
<th>Type</th>
<th>Technology</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Local</td>
<td>IBM BBAPI</td>
<td>LLNL (Sierra)</td>
</tr>
<tr>
<td>Machine Global</td>
<td>Cray Datawarp</td>
<td>LANL (Trinity)</td>
</tr>
</tbody>
</table>

How can an application utilize this layer for I/O workloads?
Burst Buffers Use Case

- Relies on integration with resource scheduler
- Different for machine-global vs. node-local storage
- Does not address inter-job data movement
Burst Buffers Use Case

<table>
<thead>
<tr>
<th>CN Usage</th>
<th>job 21</th>
<th>My Super Job</th>
<th>job 23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Burst Buffer Usage</td>
<td>pre-stage</td>
<td>usage</td>
<td>post-stage</td>
</tr>
<tr>
<td>usage by job 21</td>
<td>usage by job 23</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Perfect for Checkpoint/Restart
Enable checkpointing applications to take advantage of system storage hierarchies
Enable checkpointing applications to take advantage of system storage hierarchies

- Efficient file movement between storage layers
- Data redundancy operations
SCR Components
SCRF Component: Backend Library

- Redirect application files
- Synchronous & asynchronous flush operations
  - Hardware specific capabilities
- Data redundancy
- Support for both checkpoint & output data
SCR Component: Frontend Scripts

- **On Startup** Locate most recent checkpoint and fetch for restart
- **Within Allocation** Detect application crash or system failures and trigger restart
- **During Execution** Manage datasets
- **Resource Scheduler Integration** Pre- and post-stage data movement
SCR Component: Configurations

- Define the levels of the hierarchy
- Define modes/groups of failure
- Define checkpointing and data residency needs
SCR Component: Configurations

- Define the levels of the hierarchy
- Define modes/groups of failure
- Define checkpointing and data residency needs

Machine Portability
Combining two codes: FTI and SCR

- FTI: variable-based checkpointing scheme
- Will support existing FTI and SCR applications
- User-level file system
- Shared namespace across distributed burst buffers
- I/O interception layer
Use parallel processes to perform file operations

- Executed within a job allocation
- `dbcast`: broadcast from PFS to node-local storage
- `dcp`: multiple file copy in parallel
- `drm`: delete files in parallel
- many more

https://github.com/hpc/mpifileutils
<table>
<thead>
<tr>
<th>Site</th>
<th>Score</th>
<th>BW (GiB/s)</th>
<th>MD (KIO/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCAHPC</td>
<td>JPN</td>
<td>101.48</td>
<td>471.25</td>
</tr>
<tr>
<td>Kaust</td>
<td>SAU</td>
<td>70.90</td>
<td>151.53</td>
</tr>
<tr>
<td>Kaust</td>
<td>SAU</td>
<td>41.00</td>
<td>54.17</td>
</tr>
<tr>
<td>JSC</td>
<td>DEU</td>
<td>35.77</td>
<td>14.24</td>
</tr>
<tr>
<td>DKRZ</td>
<td>DEU</td>
<td>32.15</td>
<td>22.77</td>
</tr>
</tbody>
</table>

vi4io.org, February 2018.