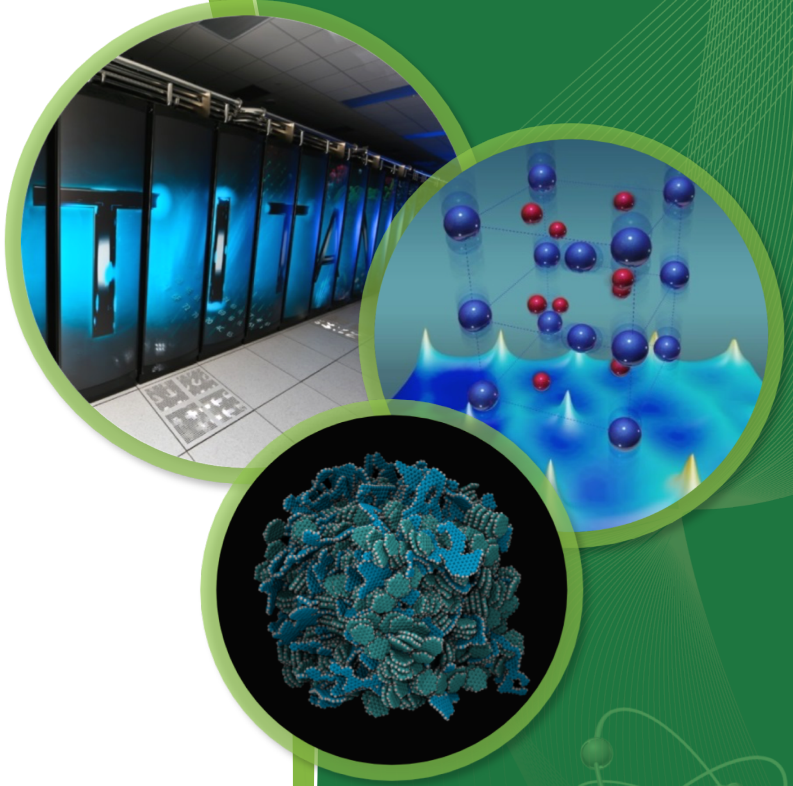
Summit Burst Buffer

Chris Zimmer



ORNL is managed by UT-Battelle
for the US Department of Energy

OAK RIDGE
National Laboratory

Summit

- Spider-3 center-wide GPFS parallel file system (PFS)
  - 250 PB @ 2.5 TB/s
  - ~540 MB/s write performance per node when all nodes are writing

- 4,608 nodes with NVMe SSDs (Samsung PM1725a)
  - 7.3 PB Total
  - 9.67 TB/s aggregate write 27 TB/s aggregate read performance when using all nodes

Burst Buffer

- Meant to absorb bursts of I/O traffic

- What it is on Summit
  - A high performance storage device 800 GB allocated for jobs
  - Software infrastructure to support different uses (transparent drain)
  - LSF Integration to support data draining after a job has ended

BBAPI for Transparent Drain

- Data will be drained from the Burst Buffer to Spider 3
  - Uses NVMeF Offloading
    - Allows the device blocks to be read over the network without using the CPU on the compute node.

- Application writes data to the burst buffer
  - Schedules the drain
  - Gets back to work
  - Data moves off on its own

OAK RIDGE
National Laboratory

BSCFS for Shared File Checkpoint

- IBM is providing BSCFS
  - Log structured file system using GPFS metadata
  - Enables scientific application to write a single shared non-overlapping file using the node local burst buffers as cache.
  - Requires directive calls in application to signal intent

Users can choose from multiple I/O options

1. Write to the PFS
2. Write to SSD, bulk copy at the end of the job
3. Write to SSD, async copy during the job (Spectral (BBAPI))
4. Use SCR to manage checkpoints, persist some, but not all
5. Use BSCFS (Single shared files to node-local burst buffers)

Things to know

- The PFS is a shared resource
  - Performance is based on shared access
    - If using the whole machine, it is 2.5 TBs/4600 nodes or ~540 MB/s/node
    - If using 20% of the machine, it is 2.7 GB/s/node *if no other jobs are writing*
  - Performance also depends on *ideal* write patterns (i.e. large, streaming writes)
    - GPFS' large random performance is much better (~90% of sequential) than Lustre's
- The node-local SSDs are not
  - Performance scales linearly with node count
  - Should exhibit low variability regardless of write pattern

Write to PFS

- Level of effort required – none; no code modification required
- Supports file-per-process, single shared file, or anything in between
- Lowest performance
- Highly variable performance

Write to SSD, bulk copy at the end of the job

- Level of effort required – minimal; no code modification required
  - Just change output directory
  - Add file copy command to job script
- Only supports file-per-process or file-per-node
- Better performance
- If copy is at the end of the job, the user may be charged for the time to “drain” to the PFS while the next job starts and runs

OAK RIDGE
National Laboratory

Write to SSD, async copy during the job

- Eg. LibSpectral
  - Level of effort required – small; no code modification required, but…
  - Need to set environment variable, link against library, and write to special directory
  - At file close(), asynchronously copies file to the PFS directory specified in the environment variable
- Only supports file-per-process or file-per-node
- Better performance, less “drain” time (less time charges)

Use SCR to manage checkpoints, persist some, but not all

- Level of effort required – moderate
  - Requires code modification and linking against the library
  - User chooses policy (persist all, persist 1 out of 10, etc.)
- Only supports file-per-process
- Best performance, especially when persisting less than all

Use BSCFS

- Level of required – Moderate; Code modification is required
  - Hybrid solution
  - Write to separate mount point
  - Uses FUSE to redirect writes to local SSD, generate log file
  - File close() triggers copy to PFS
  - Can only read local data until moved to PFS, then reads come from PFS
- Designed to support single shared file
- Better single shared file performance than writing direct to PFS

Summary

	File per Process N:N	Shared File N:1 N:M	Uses Local SSD	Allows Re-reads	Code Modification	Requires Linking	Aggregate Write Perf	Incurs Drain Charges	Visible by other systems	Data Reliability if Node Fails	Who/What Triggers I/O	Reduces Network Traffic
Write to PFS	Yes	Yes	No	Yes	No	No	2.5 TB/s	No	Yes	Yes	App/ write()	No
SSD bulk copy	Yes	No	Yes	Local Only	No	No	9.67 TB/s	Yes	After drain	No	Jobscript/ cp -r …	No
SSD async copy	Yes	No	Yes	Local Only	No	Yes	9.67 TB/s	Some	After drain	No	Intercept lib/ close()	No
SCR	Yes	No	Yes	No	Yes	Yes	9.67 TB/s	Some	Some, after drain	Option	SCR/ close()?	If not reliable
BSCFS	Yes	Yes	Yes	Local, remote from PFS	Yes	No	9.67 TB/s	Some	After drain	No	App/ close()	No