Using Darshan and CODES to Evaluate Application I/O Performance

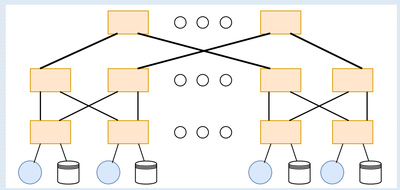
Harsh Khetawat, Christopher Zimmer, Frank Mueller,
Sudharshan Vazhkudai & Scott Atchley

NC STATE UNIVERSITY

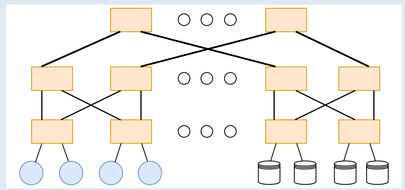
OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Motivation

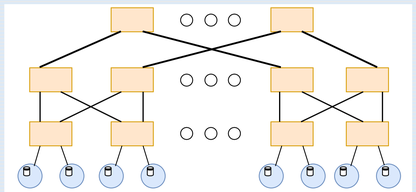
- As we approach exascale, architectural complexity and transistor density are decreasing the Mean Time To Failure (MTTF)
  - More frequent checkpointing
  - Need to decrease checkpointing overhead to not impede scientific progress
  - Storage system augmented with a layer of fast, expensive (10's of millions of dollars) SSD-based storage tier called burst buffers
- Several considerations have to be made to ensure the best investment
  - Different burst buffer architectures – Global, Grouped, Node-local
  - Network topology – Fat-Tree, Dragonfly, Torus
  - Application I/O behavior – specific to each HPC center



Grouped



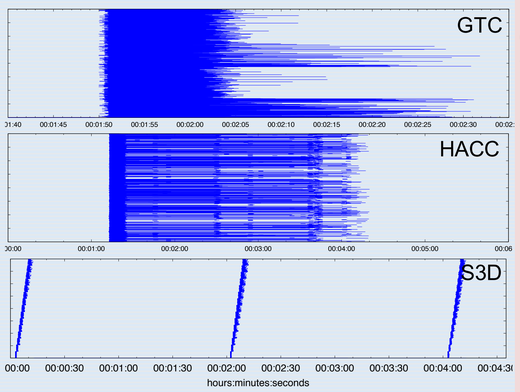
Global



Node-local

Problem

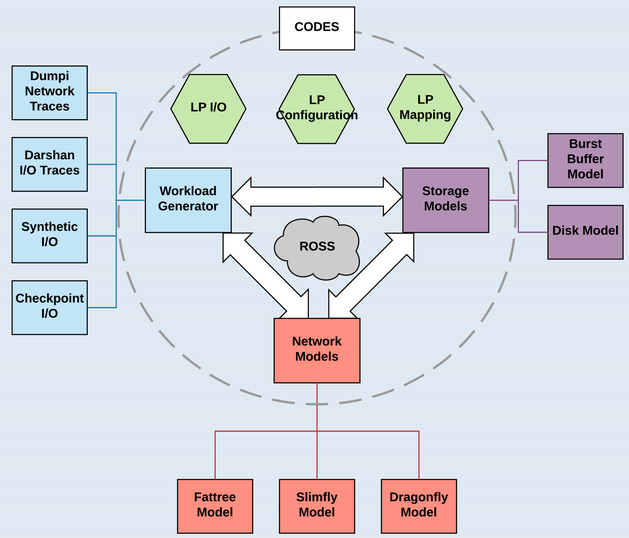
- Burst buffer architectures differ:
  - Performance – impact on storage and network congestion, length of I/O phase, application runtime
  - Capability – single shared file vs independent files, failure domains
  - Cost – storage devices, network and server infrastructure
- Current Darshan traces insufficient to simulate I/O behavior in future systems
  - Weak Scaling – applications can scale by increasing number of ranks
  - Problem Size – on bigger systems applications can have higher fidelity and hence bigger I/O sizes
  - Iterations – higher fidelity can increase the number of computation iterations, and hence I/O phases
  - Synchronization – traces don't have synchronization information and cannot capture accurate application behavior



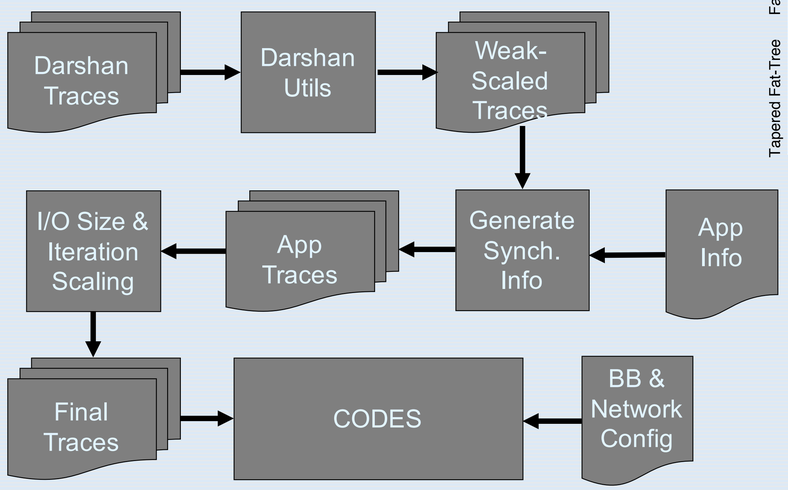
- Figure shows I/O phases for each rank of the application on timeline
- I/O patterns significantly affect performance:
  - Bulk Writes – all ranks simultaneously start I/O phase
  - Staggered – ranks stagger I/O to prevent over-saturating I/O bandwidth
  - Custom – application specific I/O patterns

Solution

- Create framework to simulate I/O from real-world application traces across burst buffer architectures and network topologies using CODES
  - CODES, built on ROSS, is a Parallel Discrete Event Simulator that allows modelling of different network and storage configurations in HPC
  - Ability to simulate full workloads from HPC centers
  - Several jobs running on the system at the same time
- We add models for different burst-buffer architectures on CODES along with support for:
  - Striping I/O across multiple storage devices
  - Client-side buffering of I/O
  - Tracking individual I/O phases of the application

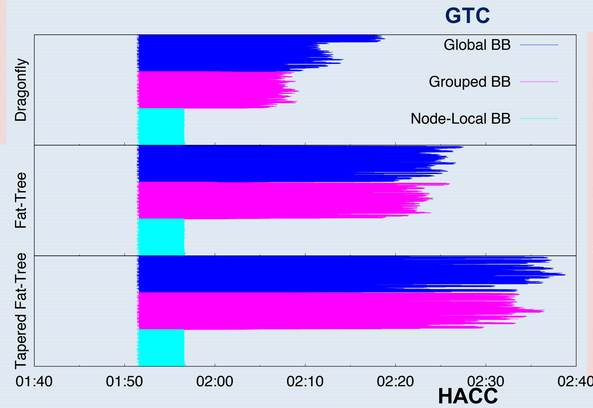


- To accurately simulate application I/O behavior in current and future systems, we augment darshan traces
  - Using darshan-utils we can weak-scale the application to increase the number of ranks while maintaining expected I/O patterns
  - We can also increase the number of I/O phases of the application to simulate future runtimes
  - We add synchronization information to the traces to capture application behavior
  - Finally, we scale the application I/O sizes to represent future workload sizes

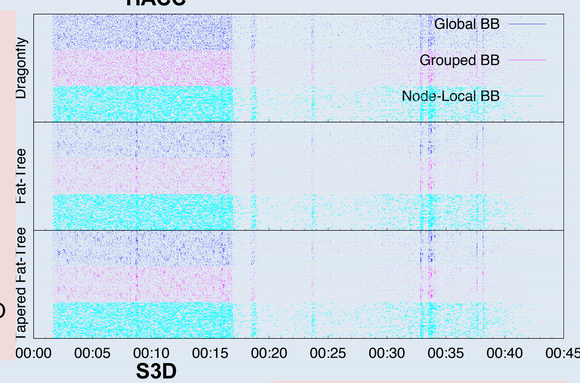


Results

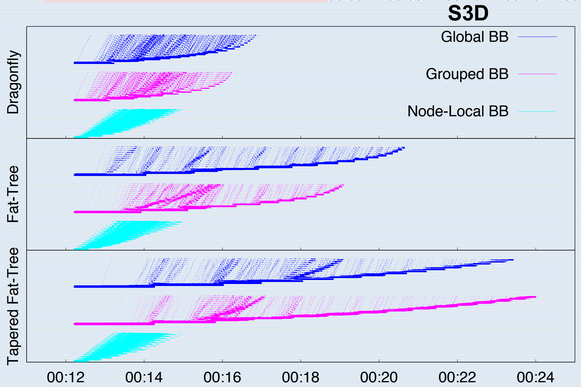
- We evaluate I/O performance of 3 applications – GTC, S3D and HACC across 3 network topologies and burst buffer configurations
  - Fat-Tree Network
  - Dragonfly Network
  - 2:1 Tapered Fat-Tree Network
- Figures show I/O phase of each rank on timeline for BB and network configuration



- Figure shows the I/O phase of GTC across BB and network configurations
- Node-local configuration is unaffected by choice of network and is fastest
- Grouped BBs perform significantly better for Dragonfly configuration
- Tapering in fat-tree networks has a significant impact on I/O performance



- HACC exhibits scattered I/O behavior
- Since I/O is not bursty, storage device bandwidth becomes bottleneck
- The I/O phase across configurations remains the same
- But ranks in node-local configuration spend more time in I/O due to limited I/O bandwidth



- S3D has a staggered I/O pattern, with only 400 ranks doing I/O at any given time
- Results in localized bursty traffic
- Node-local configuration completes I/O phase fastest
- Tapering slows grouped configuration drastically as network becomes bottleneck
- In the dragonfly network, global and grouped show similar performance

Conclusion

- Burst buffer architecture, network topology and application I/O pattern has significant impact on application I/O performance
- Darshan traces augmented with our framework can be used along with our burst buffer models to simulate expected performance of applications on future HPC systems
- We are also studying the impact on I/O performance of applications:
  - When they are scheduled alongside other HPC applications
  - In the presence of adversarial workloads