Topolgy-aware data aggregation for parallel I/O on Blue Gene/Q
François Tessier, Preeti Malakar, Venkatram Vishwanath, Emmanuel Jeanot, Florin Isaila
Email: f tessier@anl.gov - Argonne Leadership Computing Facility

Reading and writing data efficiently from storage systems is critical for high performance data-centric applications. These I/O systems are being increasingly characterized by complex topologies and deeper memory hierarchies. Effective parallel I/O solutions are needed to scale applications on current and future supercomputers.

General Overview
- The gap between the computational capacity and I/O performance of supercomputers is growing.
- Higher resolution and higher fidelity scientific simulations have high I/O requirements.

Scientific domain | Simulation | Data size |
--- | --- | --- |
Cosmology | Q Continuum | 2 PB / simulation |
High-Energy Physics | Higgs Boson | 10 PB / year |
Climate / Weather | Hurricane | 240 TB / simulation |

Performance of parallel I/O on current supercomputers
![Performance graph](image)

Data Aggregation
- Collect data from processes to write larger messages.
- Reduce network contention.
- Increase I/O bandwidth.

Analytical Model

Number of aggregators
Given:
- \( BS \): The block size of the target file system
- \( D_{ii} \): The total amount of data to be written
- \( Mem \): A available memory for an aggregator
- \( #Aggr \): The number of aggregators to select
- Find a number of aggregators such that:
  - The aggregator writes more data than the block size to mitigate file system overheads such as locking.
  - Current limitation: \( #Aggr \) is a power of 2.

\[
#Aggr = \lceil \log_2 \left( \frac{BS}{Mem} \right) \rceil
\]

TopoI/O-aware placement of aggregators
Given:
- \( V_C \): The set of compute nodes performing aggregation
- \( \omega(u, v) \): The data size exchanged between nodes \( u \) and \( v \)
- \( A \in V_C \): An aggregator chosen among compute nodes
- \( l \): The interconnect latency
- \( B \): The bandwidth between two compute nodes
- \( d(u, v) \): The number of hops between nodes \( u \) and \( v \)
- \( IO \): The I/O node

- Sending data from compute nodes to the aggregator:

\[
C_1 = \max \left( l \times d(i, A) + \frac{\omega(i, A)}{B_{i \rightarrow A}}, i \in V_C, i \neq A \right)
\]

- Sending data from the aggregator to the I/O node:

\[
C_2 = l \times d(A, IO) + \frac{\omega(A, IO)}{B_{A \rightarrow IO}}
\]

- Our objective function consists of minimizing the sum:

\[
\text{TopoAware}(A) = \min(C_1 + C_2)
\]

Challenges at scale
- Complex interconnect on supercomputers (5D Torus, Dragonfly) and complex/deep memory hierarchy (MC-DRAM, DRAM, NVRAM).
- Where to place aggregators in the topology?
- What is an efficient number of aggregators to manage data?

Experiments
- HACC-I/O: I/O part of Hardware Accelerated Cosmology Code
- Architecture:
  - 4096 nodes on Mira with 16 PowerPC A2 cores, 1600 MHz (65536 cores)
  - 5D Torus network, 1.8 Gbps per link
  - 1 GB Memory per core
  - GCC v4.4.7, MPICH2 v1.5
- Compared to two greedy strategies and MPI collective I/O.

Results

Figure 4: Write bandwidth for each aggregator mapping strategy according to the data size.

Observations
- Improved performance in case of collective writes for non-contiguous data over the default collective I/O.
- Up to 180% performance improvement for small messages compared to collective write of contiguous data.
- Topology-aware approach leads to better performance than greedy strategies except for 4MB message size.

Future work
- Scale to a larger core counts
- Expand to include more varieties of data patterns
- Generalize to incorporate other supercomputing architectures.