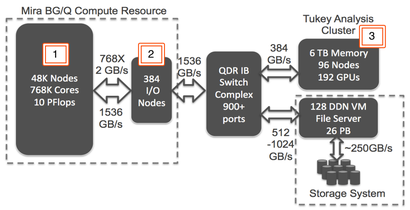
Improving I/O Performance for Sparse Data Movement in Leadership Systems

Huy Bui(abui4@uic.edu)°, Venkatram Vishwanath(venkat@anl.gov)[a],
Jason Leigh(spiff@uic.edu)°, Michael E. Papka(papka@anl.gov)[a]
°Electronic Visualization Laboratory, University of Illinois at Chicago, [a]Argonne National Laboratory

Argonne Leadership Computing Facility

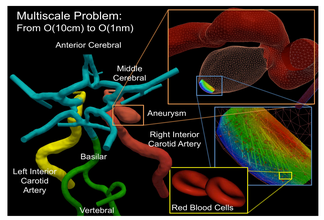
Motivations

In situ data analysis and visualization



Specific regions of interest need to be visualized.

Multiphysics application



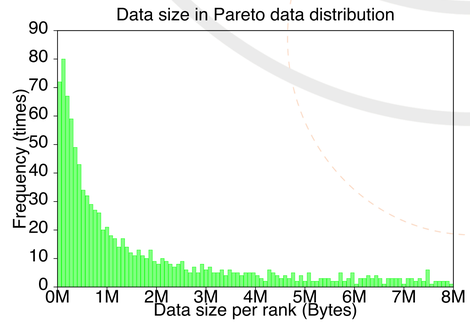
Each application writes data out at different frequencies.

In both cases, only a subset of data needs to be moved out while fully exploiting the underlying system resources. We call this as sparse data movement patterns.

Sparse Data Pattern

We developed a benchmark to model the sparse data pattern. Here, many processes have no data to write, and a few have much to move. We model this pattern based on a Pareto distribution function.

We developed a framework called GIO to handle sparse data movement.

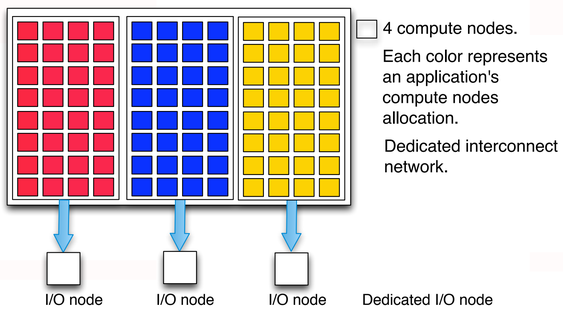


We compare performance of our framework GIO vs. MPI Collective IO.

Systems

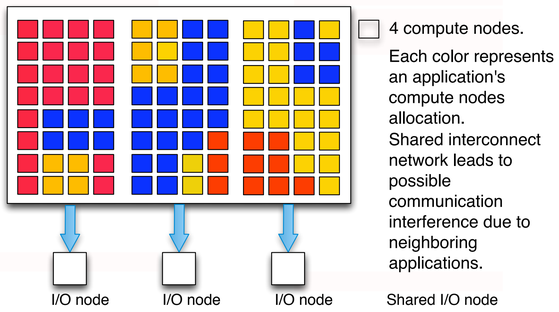
Blue Gene/Q

Every 128 compute nodes (pset) connects to a dedicated I/O node to transfer data by default.

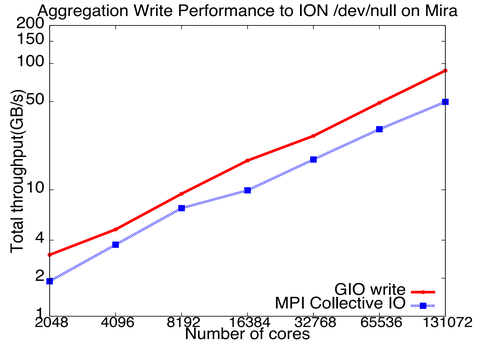


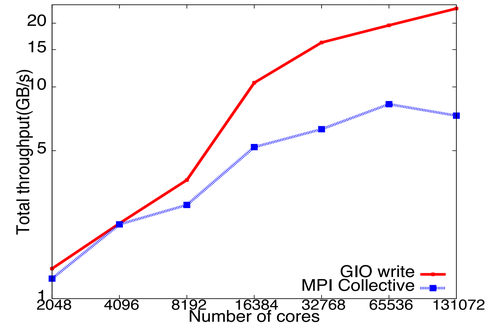
Cray XE6

Compute nodes of different applications can be interfered.



Microbenchmarks - Blue Gene/Q - Mira





Solutions

A. Custom 2-phase topology-aware I/O.

I. Number of aggregators.

T: total data size. S: data size each aggregator writes. D: maximum no. of write requests at a time can be handled by a system. Number of aggregators: N = T/S > D : D ? T/S.

II. Location of aggregators.

Blue Gene/Q

1. Distribute N aggregators uniformly in cluster, even at pset with no write requests.
2. Start from process 0, group processes into subgroups with an aggregator per subgroup and each subgroup has data size S.
3. Gather data to aggregators and write to file.

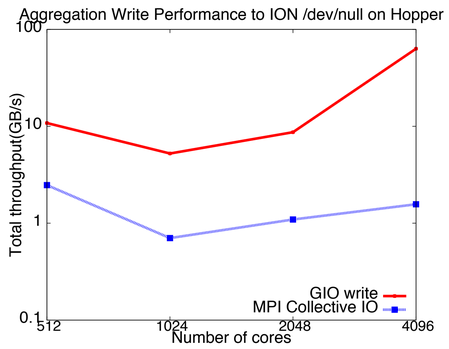
Cray XE6

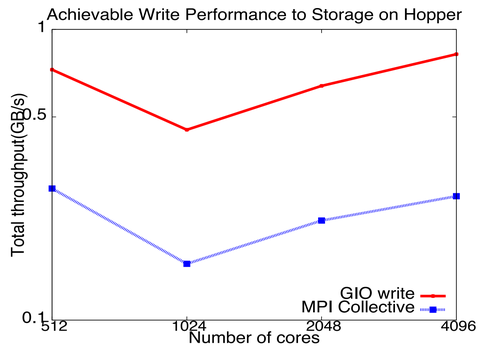
1. Order all processes by coordinates.
2. Group processes into subgroup start from process 0, to have data size S per group.
3. Gather data to aggregators and write to file.

B. Subfiling.

Align data to write with stripe size and lock boundaries: set S equal to stripe size.
Number of files: one file per IO node in BG/Q.

Microbenchmarks - Cray XE6 - Hopper





Conclusions and Future Work

- We are able to achieve 2-3 times better write performance to storage on both systems.
- Taking topology into account for data movement is of paramount important in current system, and will become critical in future systems as topology is expected to get more complex.
- We plan to cluster the processes based on their data size and distances and integrate our work into applications to see the performance on actual data.