High Performance IO on Busy Systems

Jay Lofstead, Qing Liu, Scott Klasky, Michael Booth,
Ron Oldfield, Karsten Schwan, Matthew Wolf
lofstead@cc.gatech.edu, liuq@ornl.gov, klasky@ornl.gov, mike@hpcresults.com
raoldfi@sandia.gov, schwan@cc.gatech.edu, mwolf@cc.gatech.edu
Center for Experimental Research in Computer Systems
College of Computing, Georgia Tech

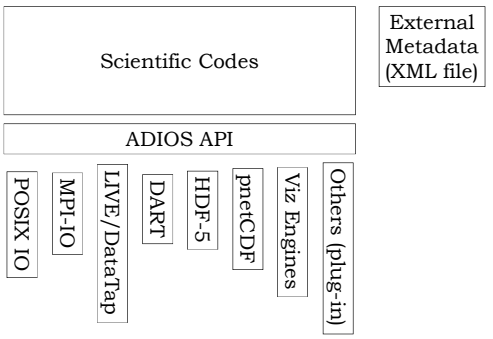
Domain Characteristics

- High Process Count
  - – Even with small per process data volumes, aggregate data volumes very large (10s of TB per output).
  - – Communication during IO can negatively impact performance.
- Large Storage Systems
  - – 100s of storage targets that must be managed to get performance.
  - – Shared use by analysis data preparation impacts other users.
- Multi-user Systems
  - – Simultaneous large jobs run concurrently (internal)
  - –File system may be shared across systems (external)
  - –Prep data in transit to aid downstream usage.

Project Goals

- ***Achieve high percentage peak IO*** on a more consistent basis
- ***Prepare data for analysis*** through annotation and embarrassingly parallel ops
- ***Flexible placement*** of operations.
- ***Flexible methods for IO*** to change inline ops to offline without changing code
- ***Improve analysis performance*** via data annotation and reorganization during output
- ***Portable*** libraries with limited dependencies for maximum options.

Host Application System View



Platform Concerns

- API performance on platform
  - – The best performing IO API for a platform varies.
  - – Some platforms do not have a working implementation of an API requiring selecting a different choice (e.g., HDF-5).
- File system characteristics vary
  - – Adjust the IO organization to meet system characteristics (stripe size/count, storage targets).
  - – Respond to variations in performance of the file system dynamically (adaptive IO techniques).
- Annotate data to aid in analysis
  - – Generate characteristics for locating data (min, max)
  - – Index data with characteristics to aid in finding
  - – Use resilient formats to protect output data

Non-Contiguous Data Layout
– Store on a per-process basis to avoid collective IO communication overheads and limits
– Resilient to failures by replicating metadata
– Move index to a footer to avoid data movement
– Annotate to handle 3-D domain decompositions easily

Mapping to Physical Resources
– IO method selected based on platform
– IO approach understands file system characteristics and adapts to take maximum advantage

Manage Variability
– Acknowledge systems are multi-user
– Acknowledge file systems may be shared across multiple systems
– Develop techniques to adapt to changing system characteristics

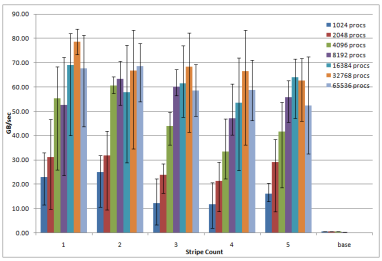
Consider Arbitrary Read Performance
– Write format has been shown to improve read performance for more than a handful of processes against arbitrary data decompositions
– Generate data characteristics to aid in data selection
  – min, max initially at write
  – other statistics (global spatial and spatial/time) at read time

Scientific Codes Integrated

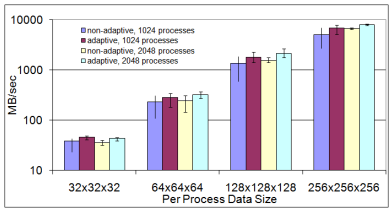
- Fusion
  - –GTC, GTS, XGC-1, XGC-0, M3D, M3D-K, Pixie3D.
- Astrophysics
  - – Chimera
- Combustion
  - – S3D
- AMR Frameworks
  - – Chombo
  - – Cactus (coming soon)
- Others
  - – GEM, GTK
  - – Starting discussions with climate and others

Performance Results

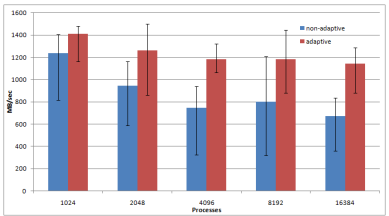
Staggered MPI-IO Large Scale Writing



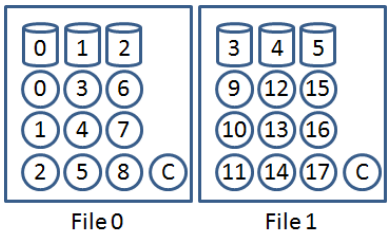
Pixie3D Adaptive Serial HDF-5 Writing



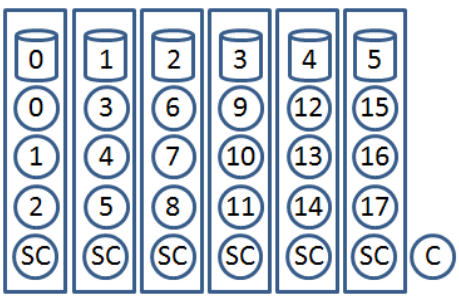
GTS Adaptive Serial HDF-5 Writing



Staggered Output Process Organization



Adaptive Output Process Organization



Summary

By working with both the science codes and the file system, the best IO method for the current platform can be selected. For each platform, different configurations must be taken into account in order to maximize the consistency and overall performance of IO. By spending a fraction more time during writing to annotate the data, reading can be greatly enhanced. Collection of simple statistics like min and max for local array pieces can be combined easily during a read operation with a nearly fixed time no matter the number of processes that wrote the data nor the size of the data written. These and other operations can not only improve write performance and reduce variability, but it can also aid in read performance.

References

1. Lofstead, J and Zheng, F and Klasky, S and Schwan, K. "Adaptable, Metadata Rich IO Methods for Portable High Performance IO.", Rome, Italy, May, 2009.
2. Lofstead, J and Zheng, F and Klasky, S and Schwan, K. Input/Output APIs and Data Organization for High Performance Scientific Computing. , Austin, Texas, November, 2008.
3. Lofstead, J and Klasky, S and Schwan K and Podhorszki, N and Jin, C. "Flexible IO and Integration for Scientific Codes Through The Adaptable IO System (ADIOS).", CLADE 2008 at HPDC, Boston, Massachusetts, June, 2008.

http://adiosapi.org/