

Exascale end to end computing

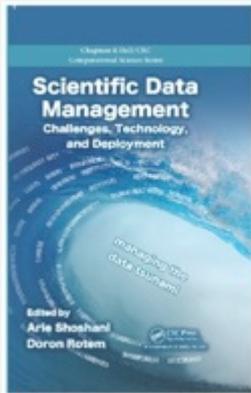
Workshop on Exascale Data Management, Analysis, and
Visualization

Houston TX

2/22/2011

Scott A. Klasky

klasky@ornl.gov



ORNL: Q. Liu, J. Logan, N. Podhorszki, R. Tchoua
Georgia Tech: H. Abbasi, G. Eisenhauer,
K. Schwan, M. Wolf.
Rutgers: C. Docan, M. Parashar, F. Zhang
Sandia: J. Lofstead, R. Oldfield
NCSU: X. Ma, N. Samatova, M. Vouk
LBNL: A. Shoshani, J. Wu
Auburn: Y. Tian, W. Yu
+ many more



Outline

- High End Computing Trends.
- The long road towards yotta-scale computing
- Conclusions.
- NOTE: The I/O system at all levels – chip to memory, memory to I/O node, I/O node to disk— will be much harder to manage due to the relative speeds of the components.
- ALSO NOTE: Programming on many cores will be more difficult as well.

Work supported under DOE funding:

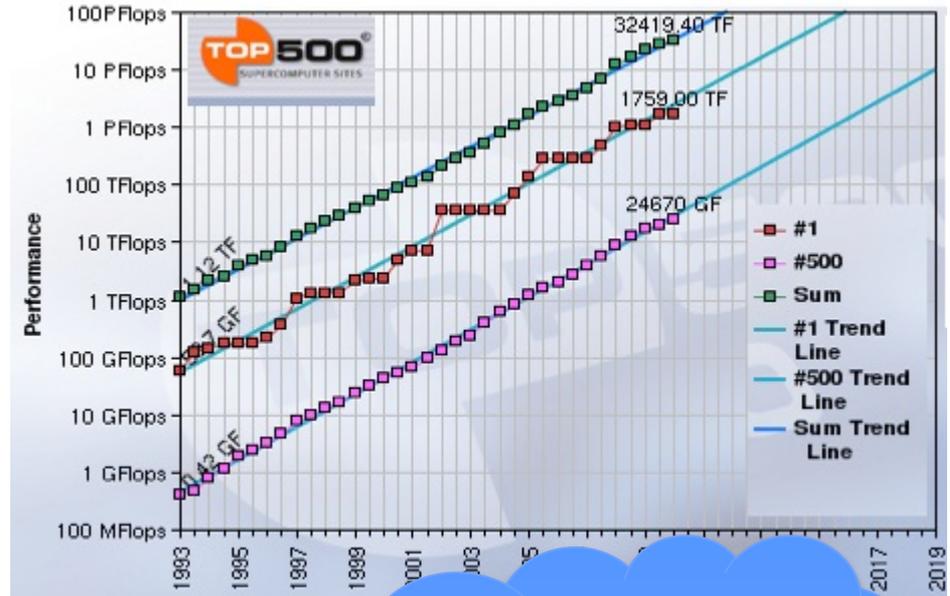
ASCR: SDM Center, CPES, Runtime Staging, SAP

OFES: GPSC, GSEP

NSF: HECURA, RDAV

Extreme scale computing.

- Trends
 - More FLOPS
 - Limited number of users at the extreme scale
- Problems
 - Performance
 - Resiliency
 - Debugging
 - Getting Science done
- Problems will get worse
 - Need a “revolutionary” way to store, access, debug to get the science done!



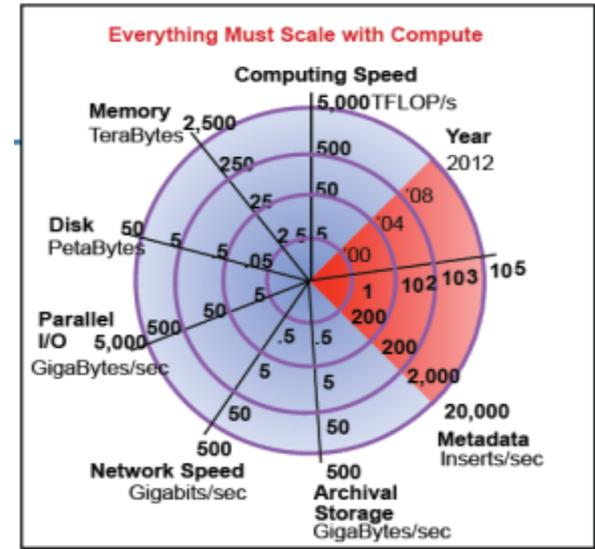
Systems	20	100	1,000	10,000
System Peak Flops/s	2	100	10,000	1,000,000
System Memory	100 MB	1 GB	10 GB	100 GB
Node Performance	100 MFlops	1 GFlops	10 TFlops	100 PFlops
Node Memory BW	100 MB/s	1 GB/s	10 GB/s	100 GB/s
Node Concurrency	100	1,000	10,000	100,000
Interconnect BW	1.5 GB/s	10 GB/s	100 GB/s	1,000 GB/s
System Size (Nodes)	18,700	100,000	1,000,000	10,000,000
Total Concurrency	225,000	3 Million	50 Million	1 Billion
Storage	15 PB	30 PB	150 PB	300 PB
I/O	0.2 TB/s	2 TB/s	10 TB/s	20 TB/s
MTTI	Days	Days	Days	0(1Day)
Power	6 MW	~10 MW	~10 MW	~20 MW

Most people get < 10 GB/s at scale

ILLUSTRATION: A. TOVEY

File System, Problems for the Xscale

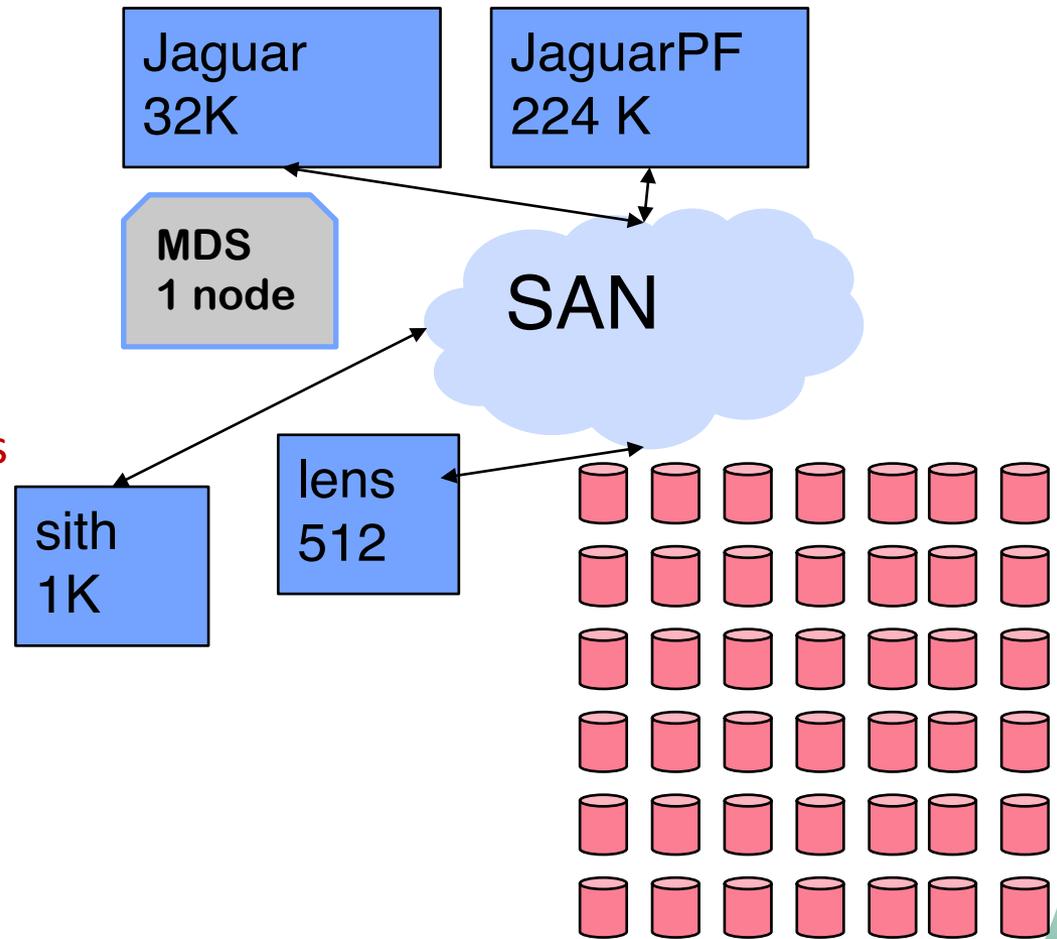
- The I/O on a HPC system is stressed because
 - Checkpoint-restart writing
 - Analysis and visualization writing
 - Analysis and visualization reading
- Our systems are growing by 2x FLOPS/year.
- Disk Bandwidth is growing ~20%/year.
- Need the number of increase faster than the number of nodes
- As the systems grow, the MTF grows.
- As the complexity of physics increases, the analysis/viz. output grows.
- Need new and innovative approaches in the field to cope with this problem.
- The biggest problem is the \$\$\$ of I/O, since it's not FLOPS



Garth Gibson 2010

Trends in HPC Centers

- Shared work-space
- Advantages
 - cheaper for total storage and bandwidth capacity
 - faster connection of resources to data
- Disadvantages
 - additional interference sources
 - potential single point of failure



Problems that apps face.

- They need to think about the way to write data for
 - Performance in writing.
 - Performance in reading.
 - Ease of archiving, moving to external resources.
- Choices are often made with incomplete knowledge of what's happening.
 - Data Layout?
 - Can users really understand the “most optimal” way to lay data on disk?
 - How many APIs should users be forced to learn?
- How to get an understanding of 1 PB+ of data.
 - Can you analyze this?
 - Can you visualize this?
 - Can you read the data?

Tricks to get you to the megascale

- Use Posix I/O (Fortran/C) I/O.
- Write 1 file.
- No restarts necessary.
- Look into self describing file formats.
- 100 MB/s IO on the Cray 1, on a separate system.



Tricks to get you to the gigascale

- Use Posix I/O (Fortran/C) I/O.
- Think about restarts.
- Try to make the data look at logically contiguous as possible.
- Separate SSD storage
- 32 GB of memory on some systems...



Netcdf-3, HDF4

- Netcdf-3
 - (<http://www.unidata.ucar.edu/software/netcdf/>)
 - Implementation is scalar.
 - 32-bit.
 - Self describing file format.
 - Stores data variable per variable in a logically contiguous layout.
- HDF4
 - <http://www.hdfgroup.org/products/hdf4/>
 - 32-bit.
 - Native file is not portable.

Tricks to get you to the terascale

- Use MPI I/O.
- Use HDF5 and netcdf for analysis, visualization .
- Write 1 file.
- Start working on restarts.
- 1 TB of disk storage.

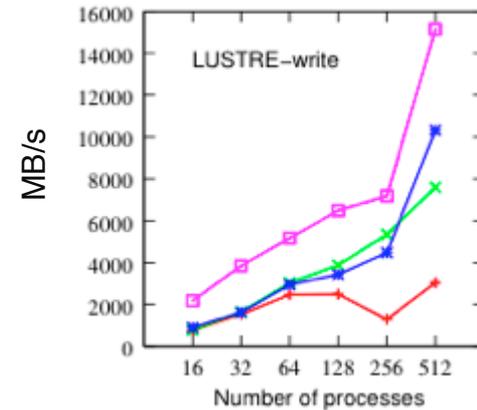
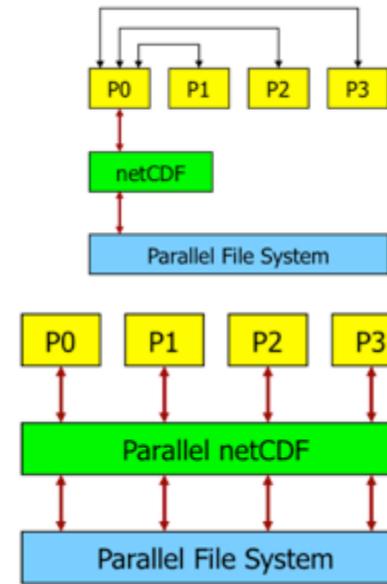
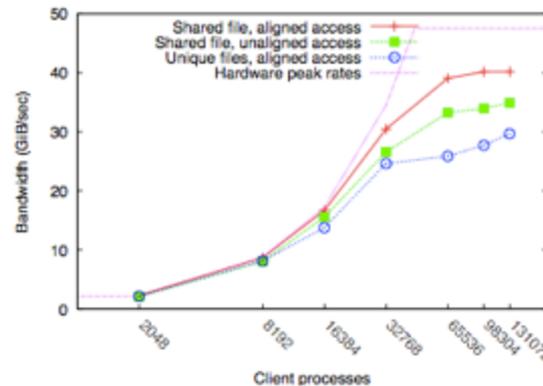
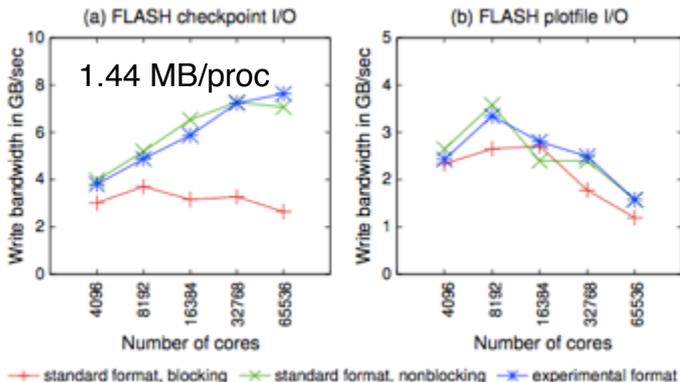


HDF5

- <http://www.hdfgroup.org/HDF5/>
- File format for storing scientific data
 - To store and organize all kinds of data
 - To share data , to port files from one platform to another
 - To overcome a limit on number and size of the objects in the file
- Software for accessing scientific data
 - Flexible I/O library (parallel, remote, etc.)
 - Efficient storage
 - Available on almost all platforms
 - C, F90, C++ , Java APIs
 - Tools (HDFView, utilities)

Parallel netCDF

- <http://trac.mcs.anl.gov/projects/parallel-netcdf>
- <http://cucis.ece.northwestern.edu/projects/PNETCDF/>
- New file format to allow for large array support
- New optimizations for non-blocking calls.
- New optimizations for sub-files.
- Idea is to allow netcdf to work in parallel and for large files and large arrays.



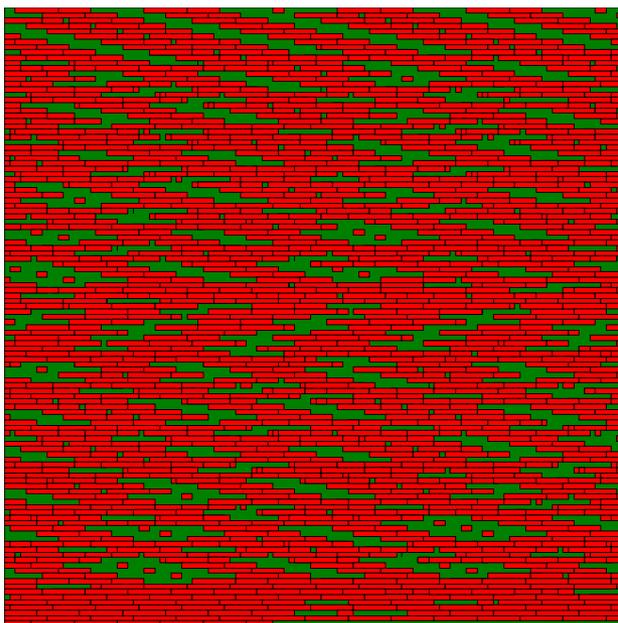
Write performance on Franklin

Using Subfilng to Improve Programming Flexibility and Performance of Parallel Shared-file I/O, Gao, Liao, Nisar, Choudhary, Ross, Latham, ICPP 2009.

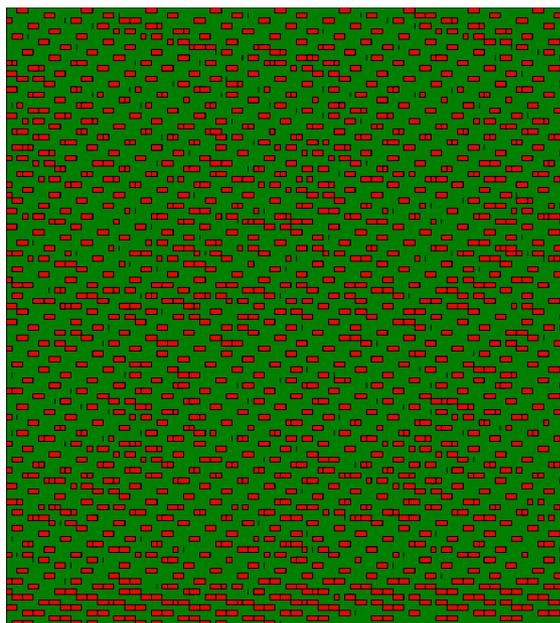
Visualizing and Tuning I/O Access

R. Ross et al.

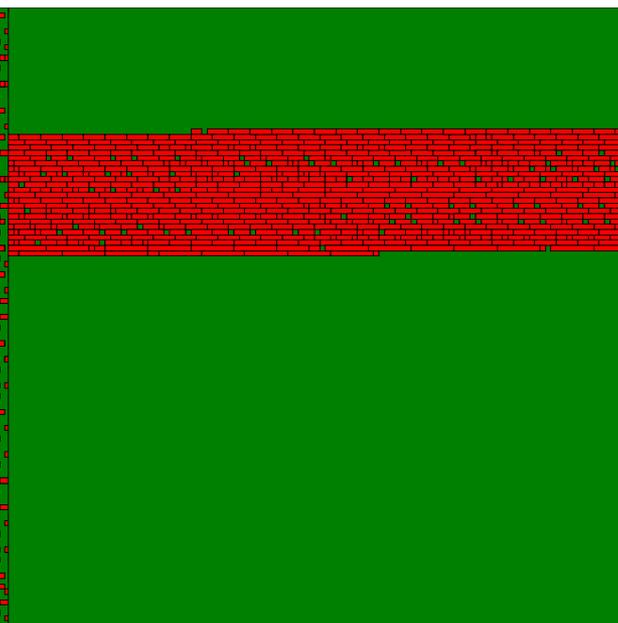
This view shows the entire 28 Gbyte dataset as a 2D array of blocks, for three separate runs. Renderer is visualizing one variable out of five. Red blocks were accessed. Access times in parenthesis.



Original Pattern



MPI-IO Tuning



PnetCDF Enhancements

Data is stored in the netCDF “record” format, where variables are interleaved in file (36.0 sec). Adjusting MPI-IO parameters (right) resulted in significant I/O reduction (18.9 sec).

New PnetCDF large variable support stores data contiguously (13.1 sec).

Run file based workflows for processing data near real time for analysis/visualization, and job control



ParameterSet



CreateDirectory



GetTimestep



Full-ELM cycle workflow version 1.0, March 2008

Author: Norbert Podhorszki, ORNL

Init



XGC



NetCDF/HDF5 Processing



XGC <-> M3D-OMP Coupling for computing Equilibrium

ELITE for linear stability check

Equilibrium Step

m3dfile

Elite

unstable

Stop unstable XGC



M3D-MPP for ELM Crash simulation



Restart preparation for XGC

RestartPrep XGC

www.kepler-project.org



Tricks to get you to the petascale

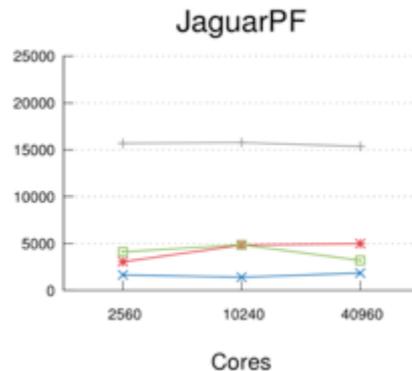
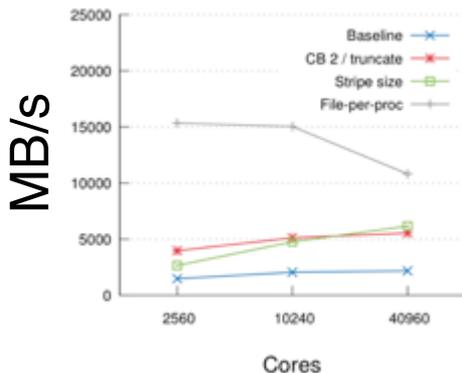
- Do not write in logically contiguous file formats.
- Write number of files proportionate to the number of Storage Targets Lustre.
- Align all requests.
- Worry about resiliency in file formats, and in the I/O pipes.
- Think about data staging for both writing and reading.
- Understand how the data will be accessed for analysis.
- Try to limit the internal interference when writing.
- Checkpoint calculations at least every 2 hours.
- Understand I/O external interference.



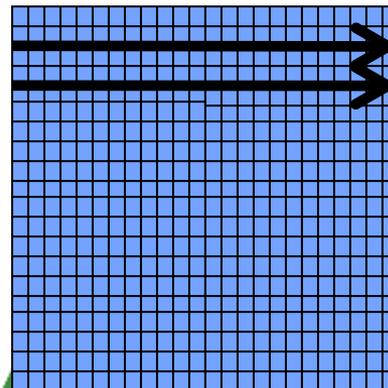
Parallel netCDF-4/ HDF5

- <http://www.unidata.ucar.edu/software/netcdf/>
- Use HDF5 for the file format.
- Keep backward compatibility in tools to read netCDF 3 files.
- HDF5 optimized chunking
- New journaling techniques to handle resiliency.
- Many other optimizations
 - http://www.hdfgroup.org/pubs/papers/howison_hdf5_lustre_iasds2010.pdf

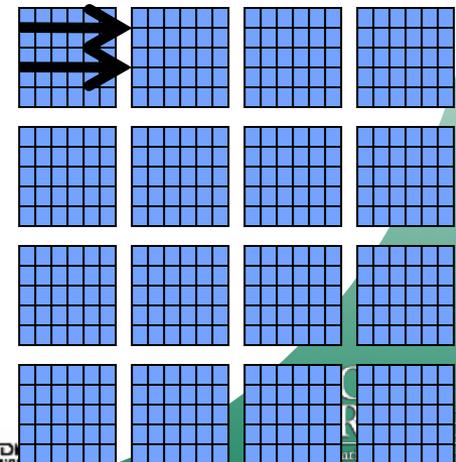
Vorpal, 40³: 1.4 MB/proc, 11.5 MB/proc
JaguarPF



Contiguous

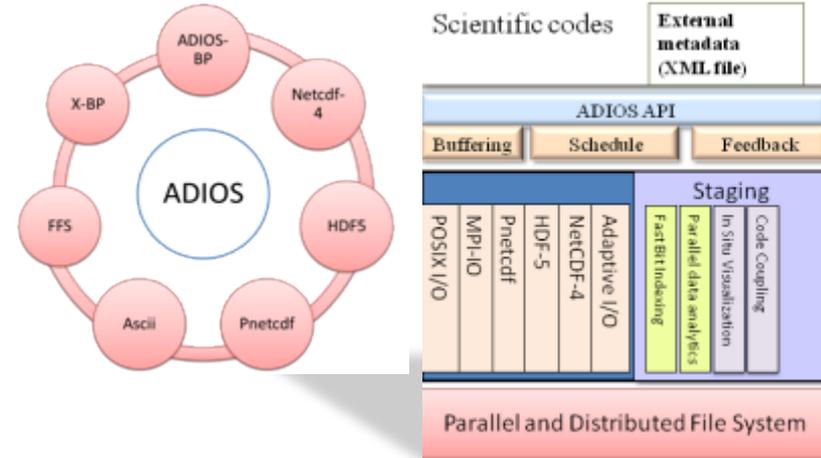


Chunked



Our solution: ADIOS: Adaptable I/O System

- Provides portable, fast, scalable, easy-to-use, metadata rich output
- Simple API
- Change I/O method by changing XML file only
- Layered software architecture:
 - Allows plug-ins for different I/O implementations
 - Abstracts the API from the method used for I/O
 - New file format (ADIOS-BP), for petascale- exascale.
- Open source:
 - <http://www.nccs.gov/user-support/center-projects/adios/>
- Research methods from many groups:
 - Rutgers: DataSpaces/DART
 - Georgia Tech: DataTap
 - Sandia: NSSI, Netcdf-4
 - ORNL: MPI_AMR

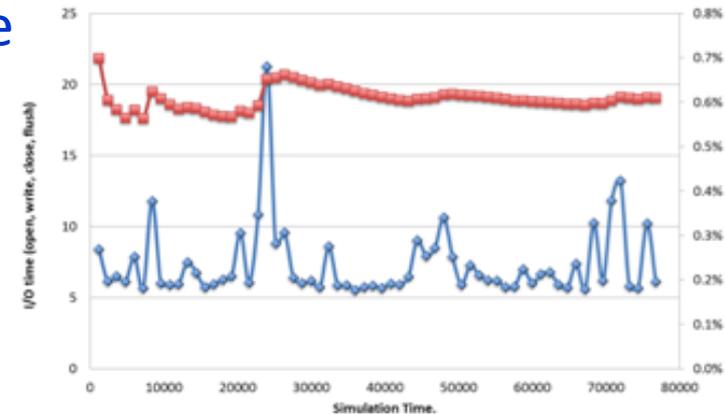


ADIOS 1.2 write speeds

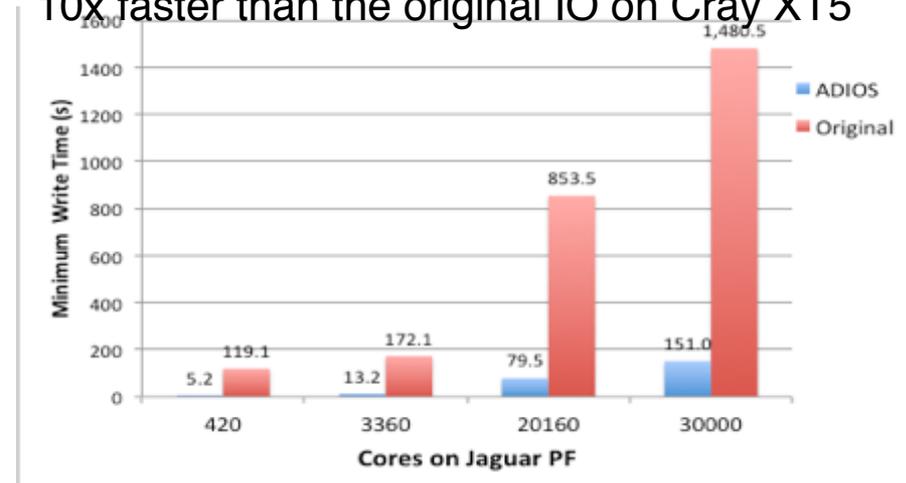
- Synchronous write speeds:
- S3D: 32 GB/s with 96K cores, 1.9MB/core
- XGC1 code → 40 GB/s
- SCEC code 30 GB/s
- GTC code: 40 GB/s
- GTS code: 35 GB/s
- + many more.
- All times include (open, write, close, flush)

Latest S3D simulation on JaguarPF
96,000 cores, each writing 1.9 MB. I/O = 25.7 GB/s, stdev=5.5 GB/s, overhead = 0.6%.

0.6% I/O overhead



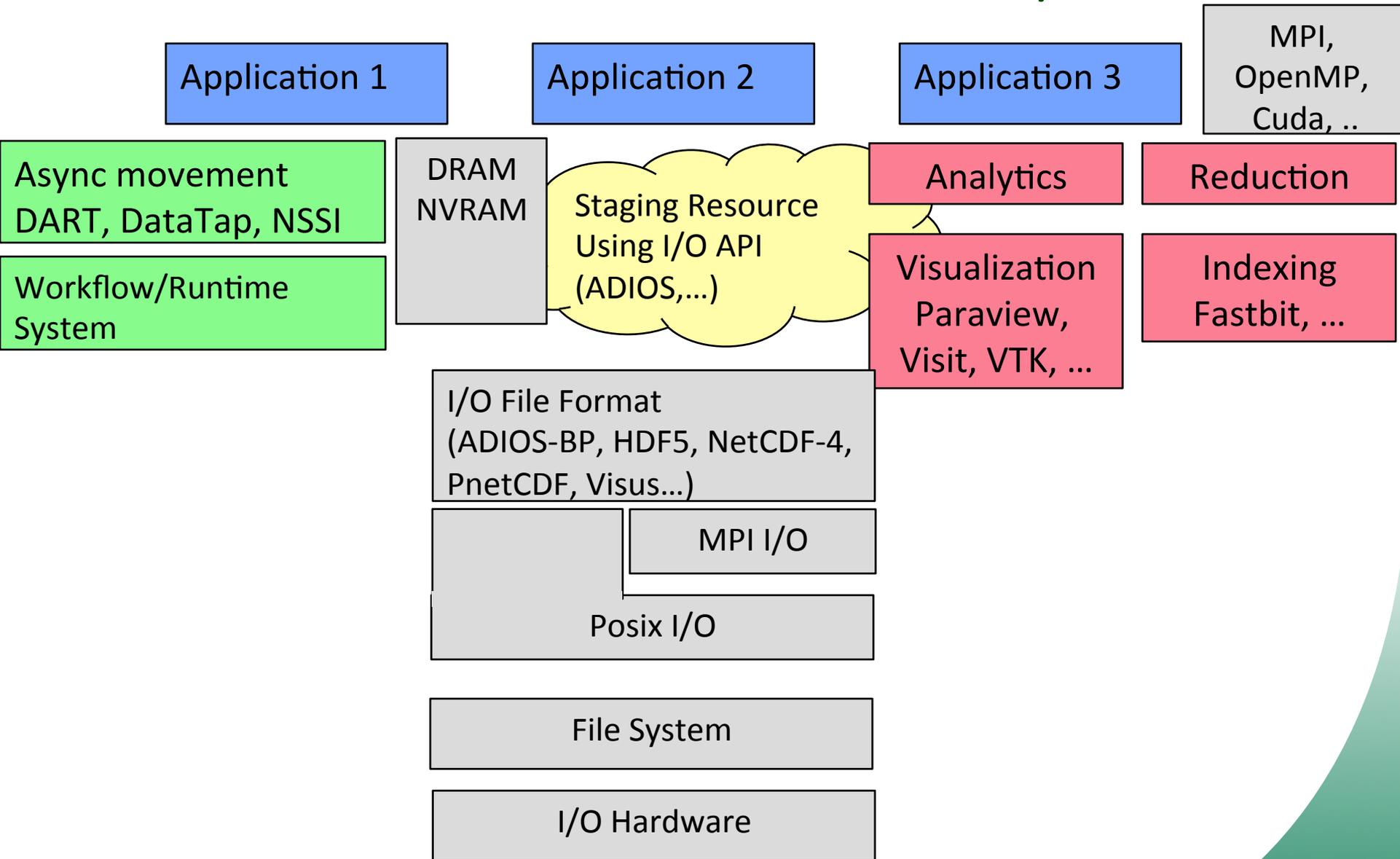
ADIOS 1.2 for the PMCL3D code, showing 10x faster than the original IO on Cray XT5



Research to look at for the exascale

- I/O pipelines need be constructed to include
 - Indexing.
 - Analytics.
 - Multi resolution analysis.
 - Writing to the file system.
 - Compression.
- Different I/O methods for different I/O patterns for both reading and writing. (Restarts, Analysis, Visualization, ...).
 - I/O for Analysis and visualization need to be re-examined for multi-resolution.
- Checkpoint in memory first, and then stage to disk, if necessary.
- Rethink file formats for resiliency.
- Must think carefully about QoS-like techniques.
- Helpful to include I/O componentization.
- Rethink for energy cost/savings.

Software Stack for Exascale I/O



The “early days”

- S. Klasky, S. Ethier, Z. Lin, K. Martins, D. McCune, R. Samtaney, “Grid -Based Parallel Data Streaming implemented for the Gyrokinetic Toroidal Code,” SC 2003 Conference.
 - V. Bhat, S. Klasky, S. Atchley, M. Beck, D. McCune, and M. Parashar, “High Performance Threaded Data Streaming for Large Scale Simulations” “5th IEEE/ACM International Workshop on Grid Computing (Grid 2004)
- **Key IDEAS:**
 - Focus on I/O and WAN for an application driven approach.
 - Buffer Data, and combine all I/O requests from all variables into 1 write call.
 - Thread the I/O.
 - Write HDF5 data out on the receiving side.
 - Visualize the data near-real-time
 - **Focus on the 5% rule..**

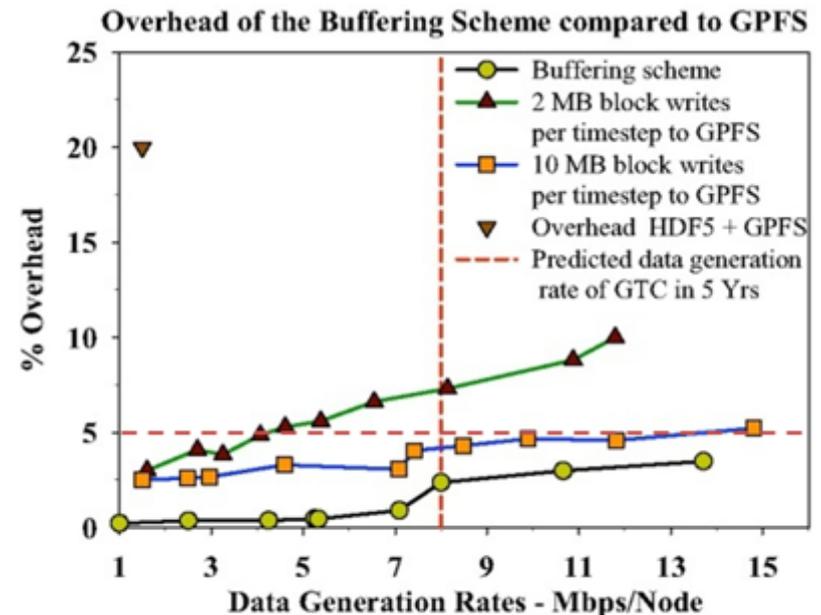
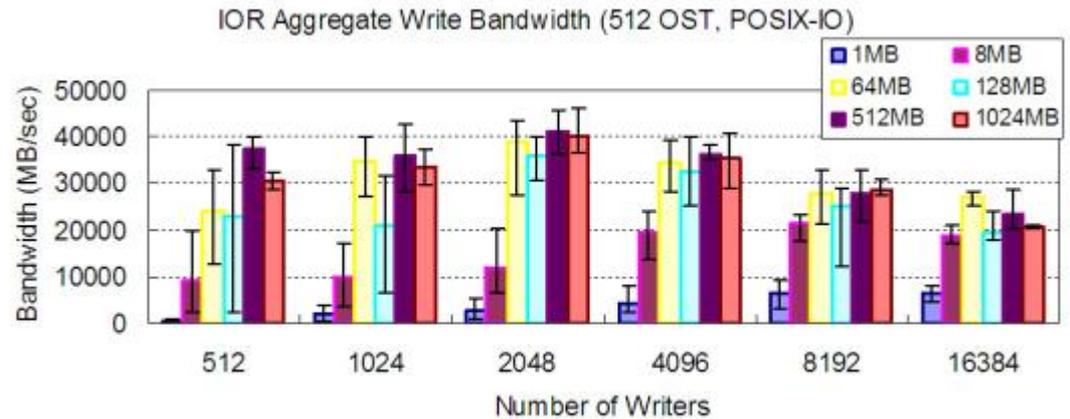


Figure: 8. Overhead with Buffering Scheme compared to GPFS (I/O).

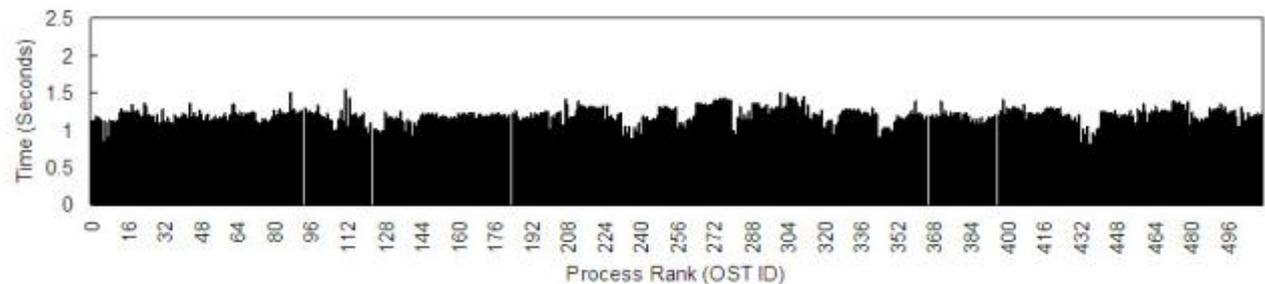
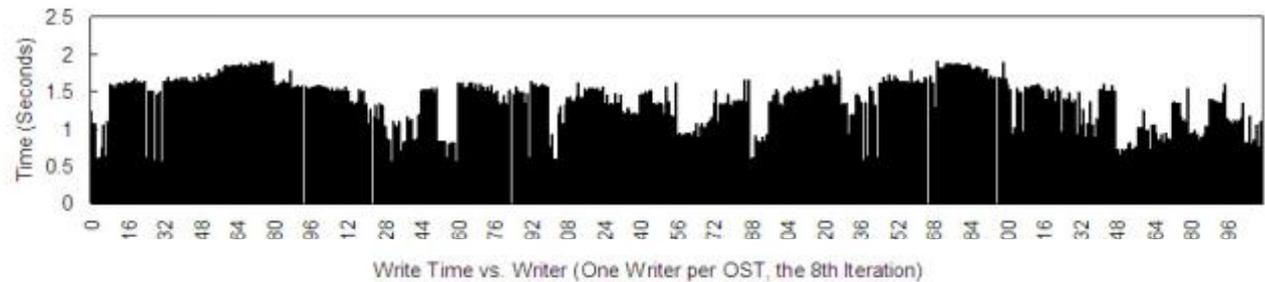
I/O interference

- Internal: at 128 MB/proc, 8k->16k process, bandwidth degrades 16-28%



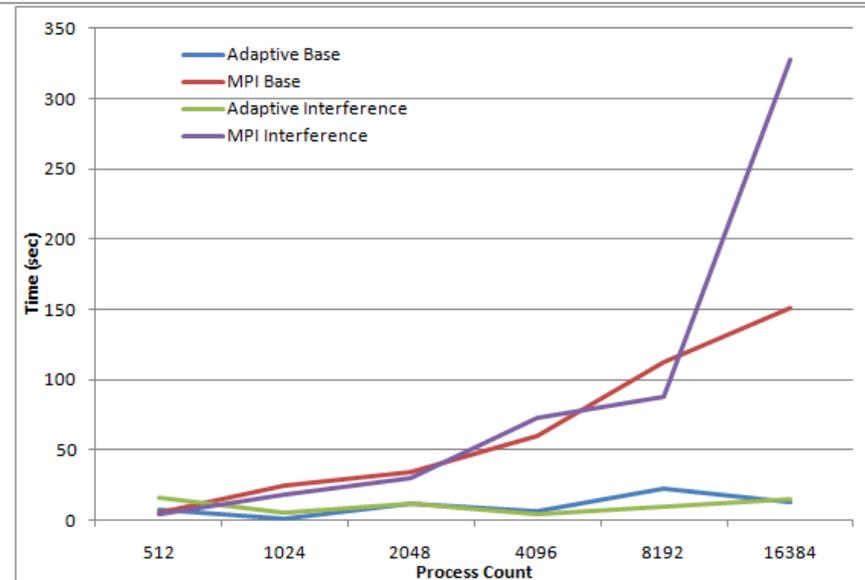
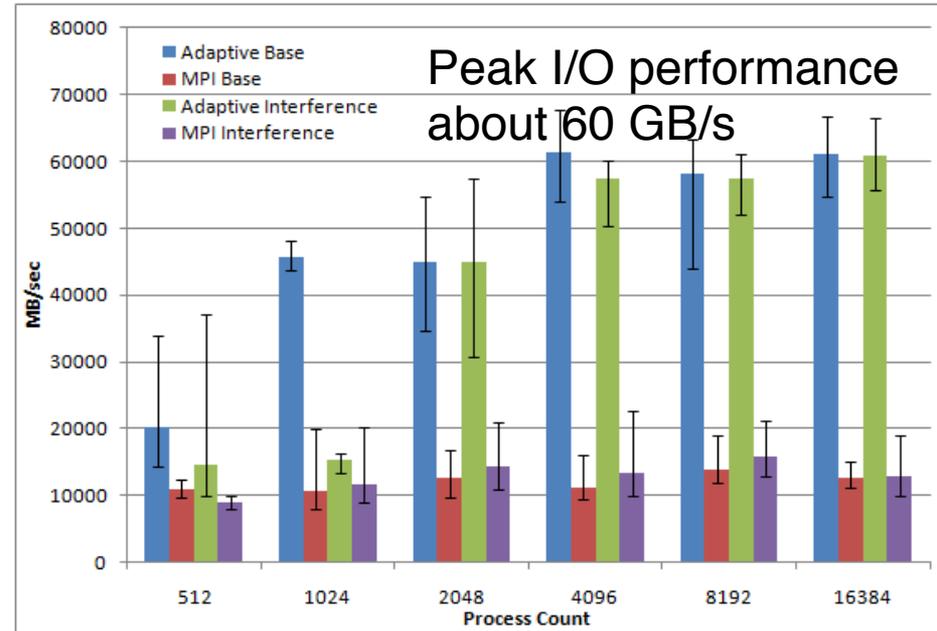
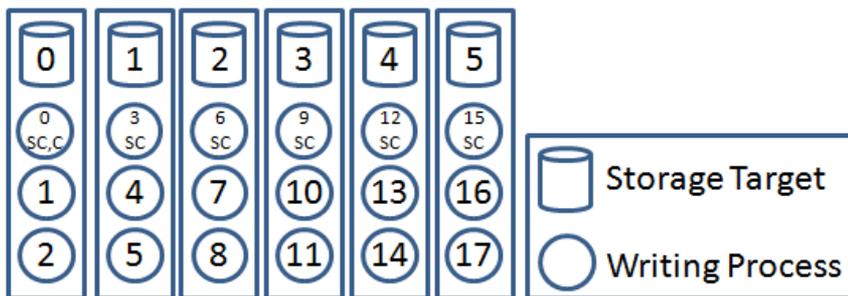
3.44 vs. 1.86 imbalance factor

128 MB/process,
3 minutes apart



Reduce the variability of I/O

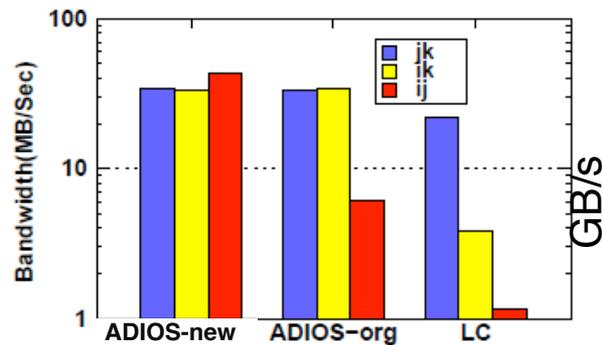
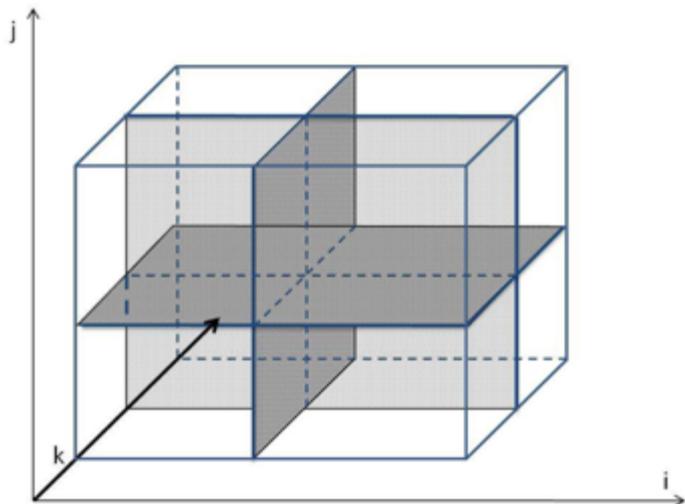
- Adaptive methods meant to handle the variability of the writes. (Lofstead et. al SC 2010).
- Creates sub files on each storage target of different sizes.



Understand the “typical” read access patterns

- Read all of the variables from an integer multiple of the original number of processors.
 - **Example: restart data.**
- Read in just a few variables on a small number of processors.
 - **Visualization**
- Read in a 2D slice from a 3D dataset (or lower dimensional reads) on a small number of processors.
 - **Analysis.**
- Read in a sub volume of a 3D dataset from a small number of processors.
 - **Analysis.**
- Read in data in multi-resolution data.

Problem of reading in 2D data from 3D dataset

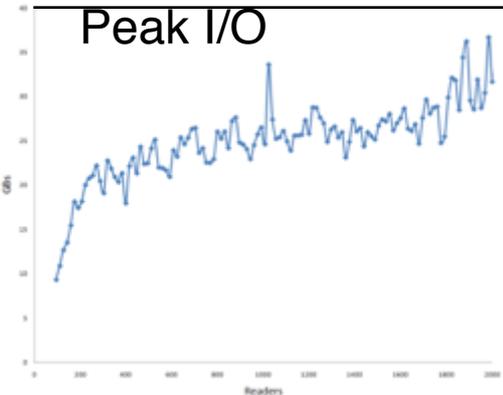


(a) Small (stripes=128)

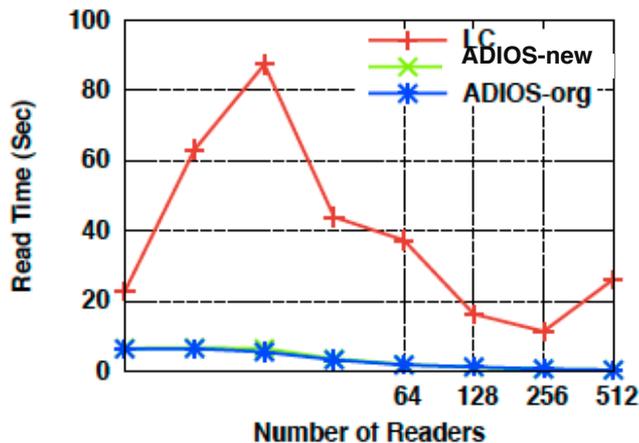
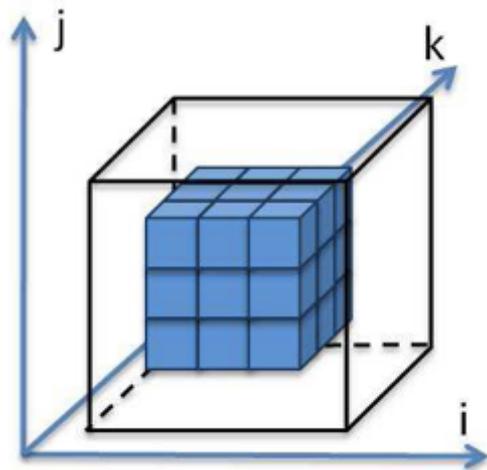
Read speed
For GTC on Jaguar

GTC Particle Data (52 GB) written in ADIOS by from 32K cores.

Peak I/O

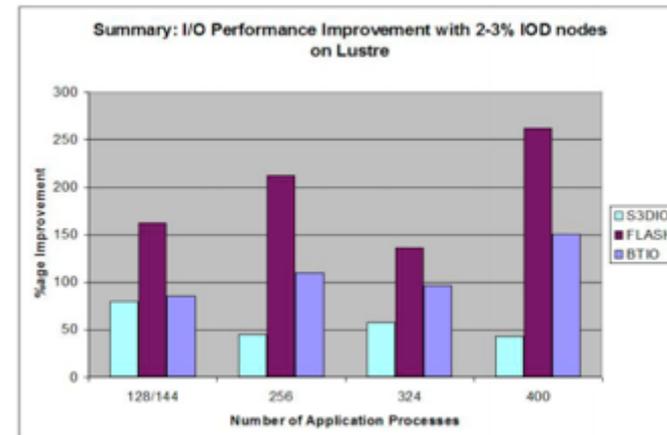
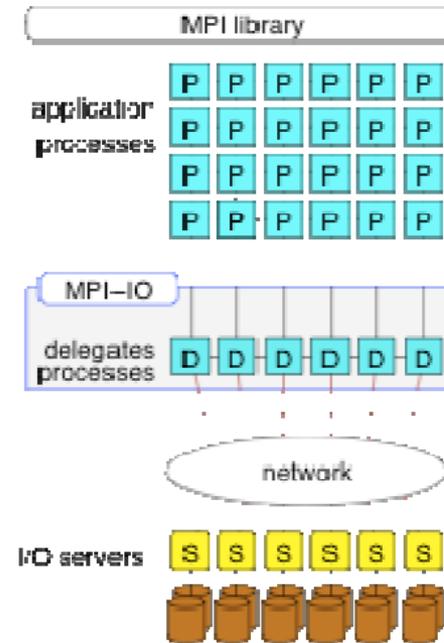


Readers



I/O delegation – Towards Active I/O management (Nisar, Choudhary et al.)

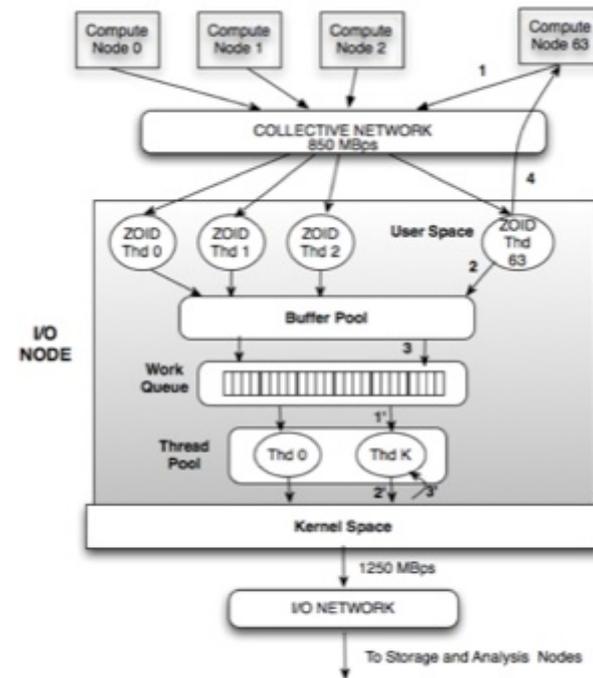
- File system is a shared resource
- Relative smaller no. I/O servers
- All clients compete to access
- Consistency is enforced across all clients
- I/O through dedicated I/O processes reduce no. clients accessing the file system
- Enable optimization, such as collaborative caching



Accelerating I/O Forwarding in IBM Blue Gene/P Systems

V. Vishwanath, et al. SC 2010.

- Uses Data Staging to improve I/O performance.
- Uses asynchronous data transfer to increase performance.
- Schedules work-queue model.
- Uses the “best” number of threads to increase performance.



Augmenting I/O forwarding with asynchronous data

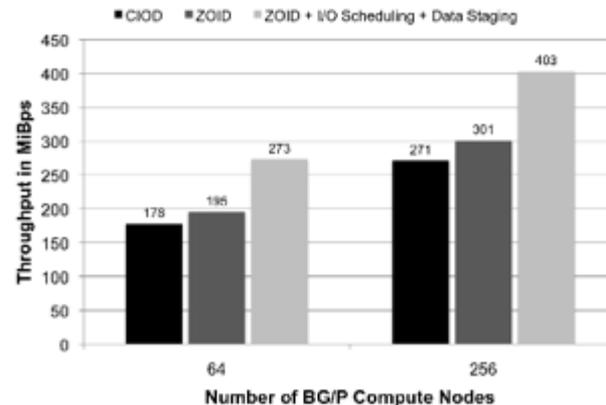


Fig. 13. Performance of the MADBench2 application benchmark using the I/O forwarding mechanisms.

Staging I/O

- Why asynchronous I/O?

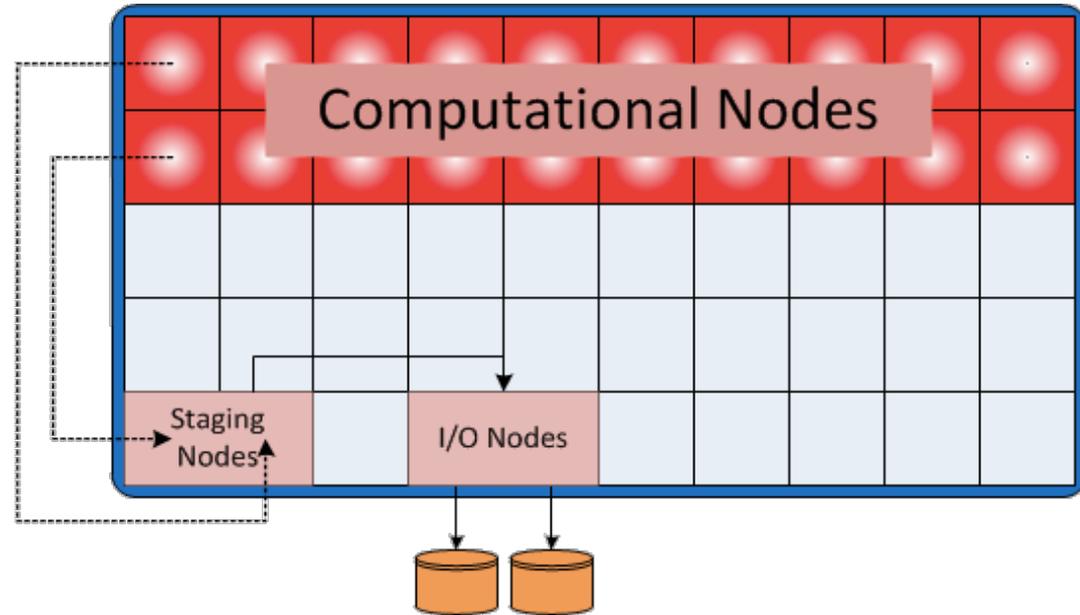
- Reduces performance linkage between I/O subsystem and application
- Decouple file system performance variations and limitations from application run time

- Enables optimizations based on dynamic number of writers

- High bandwidth data extraction from application

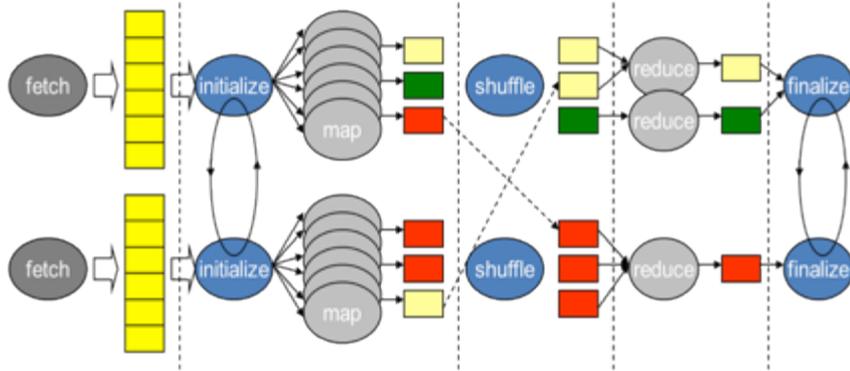
- Scalable data movement with shared resources requires us to manage the transfers

- Scheduling properly can greatly reduce the impact of I/O



Creation of I/O pipelines to reduce file activity

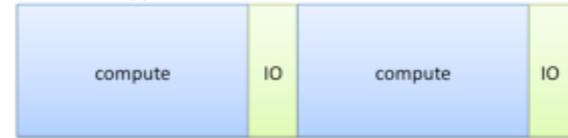
Streaming Processing in Staging Area



Differences with MapReduce:

- Two-pass streaming processing (In compute nodes or Staging Area)
- In-memory storage for speed
- Customizable shuffling phase and additional initialize/finalize phases

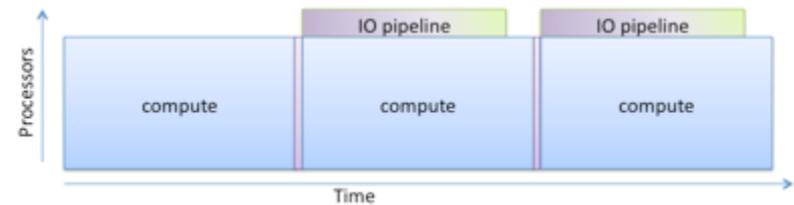
Traditional approach



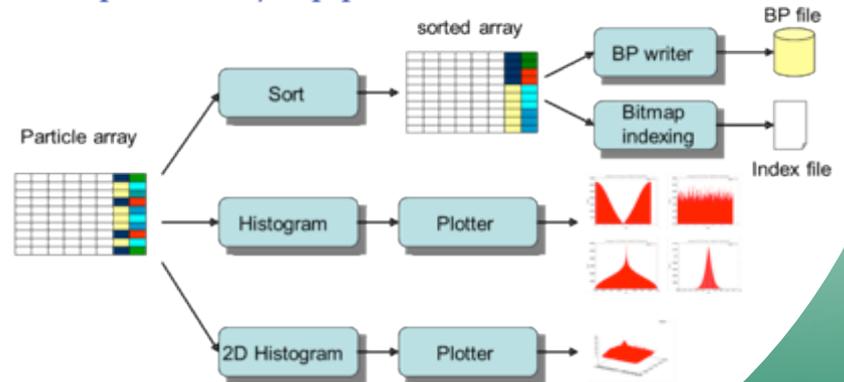
In-Compute-Node (ICN) approach



Asynchronous I/O pipeline approach with DataTap and SmartTap

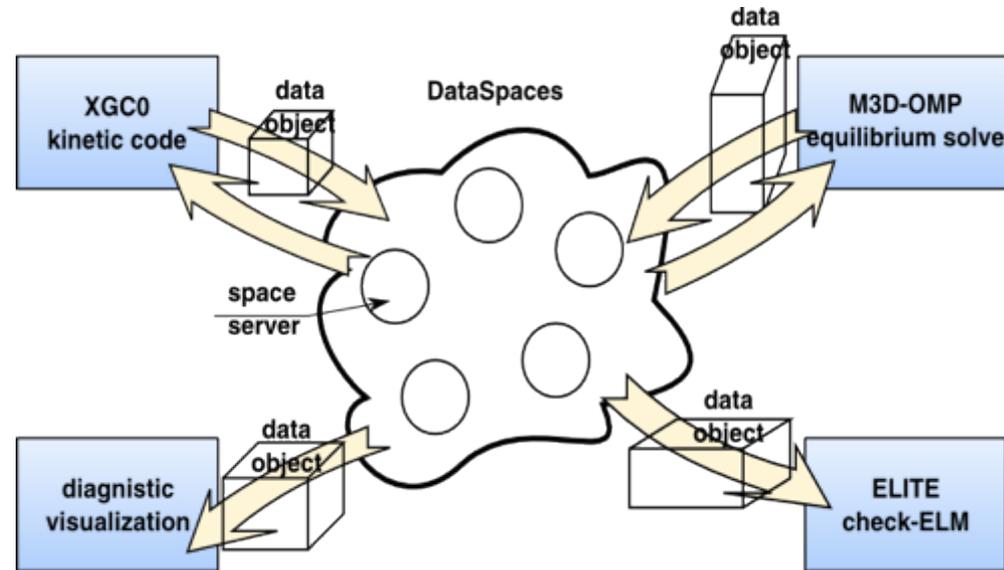


Example of an I/O pipeline



ADIOS with DataSpaces for in-memory loose code coupling

- Semantically-specialized virtual shared space
- Constructed on-the-fly on the cloud of staging nodes
 - Indexes data for quick access and retrieval
 - Provides asynchronous coordination and interaction and realizes the shared-space abstraction
- Complements existing interaction/coordination mechanisms
- In-memory code coupling becomes part of the I/O pipeline



- Supports complex geometry-based queries
- In-space (online) data transformation and manipulations
- Robust decentralized data analysis in-the-space

Move towards

- Diskless check pointing.
 - J. S. Plank and K. Li and M. A. Puening, “Diskless Checkpointing”, IEEE Transactions on Parallel and Distributed Systems, 9, 10, 1998 pp 972-986.
- New file system which include (memory to buffer on node, NV memory on certain “staging” nodes, disks, tape?).
- Active messages.
 - Great if the hardware would support this.
- Minimize data movement
 - File formats must be able to be different on disk for different systems, for different runs. Must be flexible!
- Include data analytics.
- Provenance capturing capability.
 - VisTrails, Kepler, ...

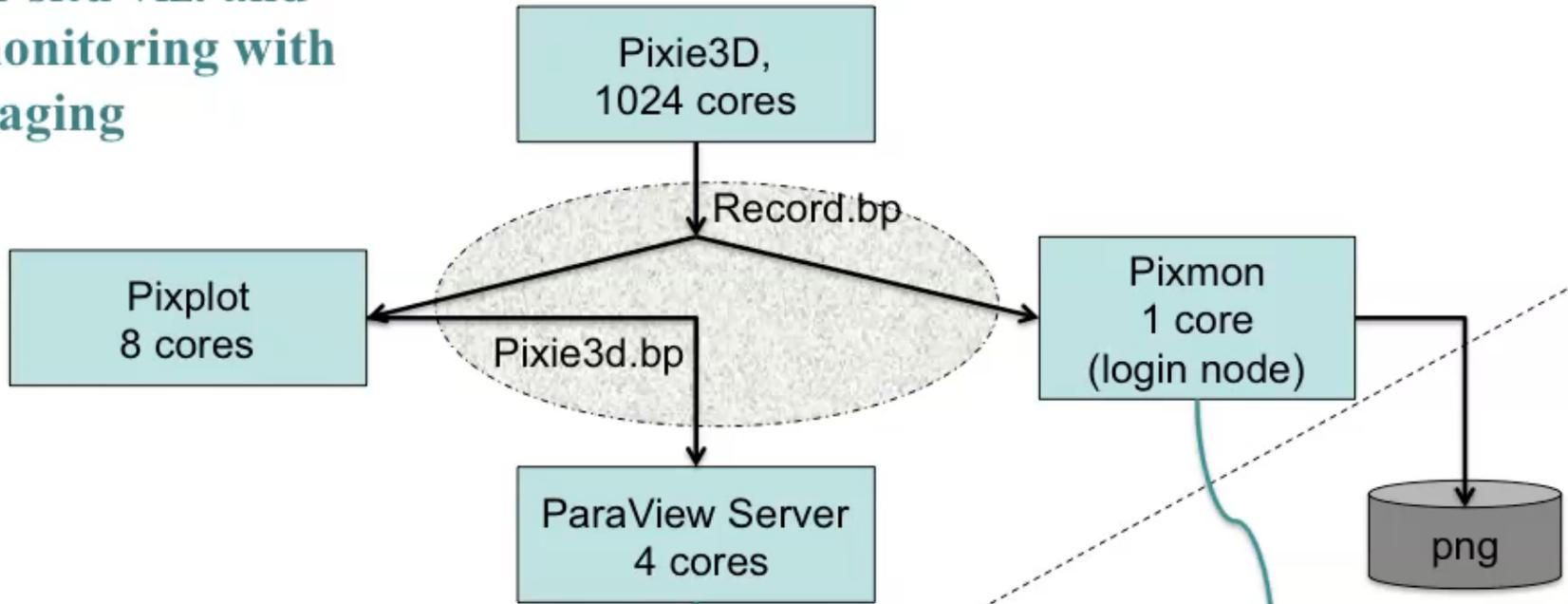
Tricks to get you to the yotta scale.

- Do a lot of praying!
- Or just assume that it will be “fixed in the next release”

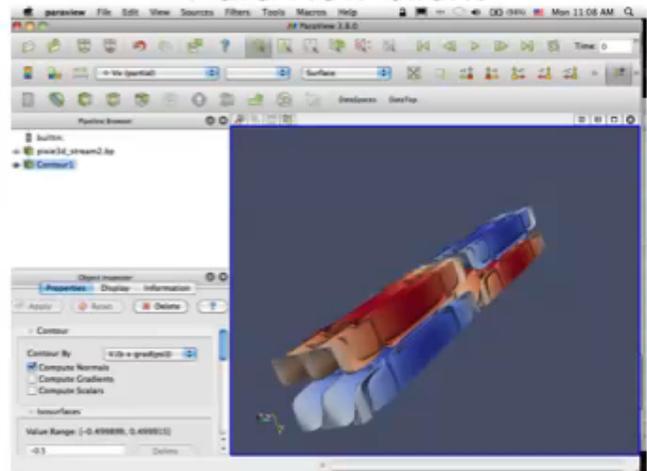
Complexity leads to a SOA approach

- Service Oriented Architecture (SOA): Software as a composition of “services”
 - *Service: “... a well-defined, self-contained, and independently developed software element that does not depend on the context or state of other services.”*
 - **Abstraction & Separation**
 - Computations from compositions and coordination
 - Interface from implementations
 - Existing and proven concept - widely accepted/used by the enterprise computing community

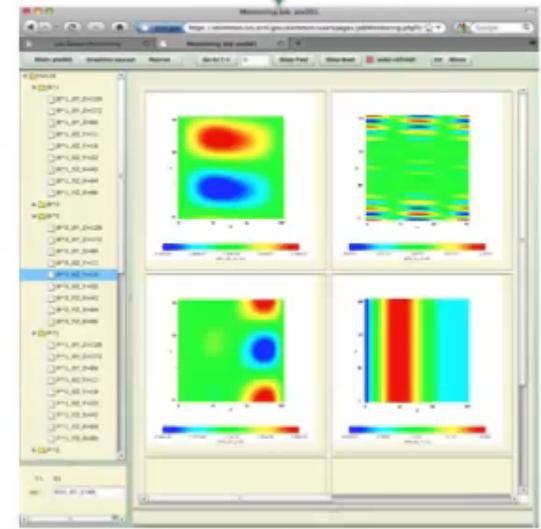
In-situ viz. and monitoring with staging



ParaView client



Port forwarding



Summary and Conclusions

- **Unprecedented opportunities for dramatic insights through computation!**
- To get to the exascale we will have to manage memory hierarchies, and combine data management, visualization, into easy-to-use solutions for applications.
- **Challenge:** *Manage complexity* while maintaining performance/scalability.
 - complexity from the **problem** (complex physics)
 - complexity from the **codes** and how they are developed and implemented
 - complexity of underlying **infrastructure** (disruptive hardware trends)
 - complexity from **coordination** across codes and research teams
- Lots of research techniques out there that need to be made into simulations.