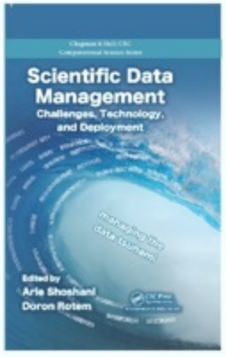
In situ data processing for extreme-scale computing



Scientific Data Management

Challenges, Technology, and Deployment

Edited by Arie Shoshani, Doron Rotem

INT Exascale Workshop

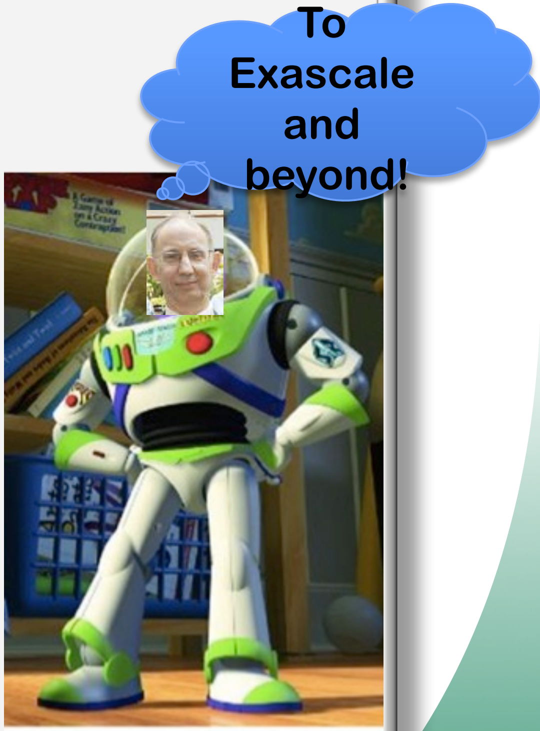
6/29/2011

Scott A. Klasky

klasky@ornl.gov

Hasan Abbasi[2], Qing Liu[1], Jeremy Logan[1], Manish Parashar[6], Karsten Schwan[4], Arie Shoshani[3], Matthew Wolf[4], Sean Ahern[1], Ilkay Altintas[9], Wes Bethel[3], Luis Chacon[1], CS Chang[10], Jackie Chen[5], Hank Childs[3], Julian Cummings[13], Ciprian Docan[6], Greg Eisenhauer[4], Stephane Ethier[10], Ray Grout[7], Jinoh Kim[3], Ricky Kendall[1], Zhihong Lin[10], Qing Liu[2], Jay Lofstead[5], Xiaosong Ma[15], Kenneth Moreland[5], Valerio Pascucci[12], Norbert Podhorszki[1], Nagiza Samatova[15], Will Schroeder[8], Roselyne Tchoua[1], Yuan Tian[14], Raju Vatsavai[1], Mladen Vouk[15], Yandong Wang[14], John Wu[3], Weikuan Yu[14], Fan Zhang[6], Fang Zheng[4]

[1]ORNL, [2] U.T. Knoxville, [3]LBNL, [4]Georgia Tech, [5]Sandia Labs, [6] Rutgers, [7]NREL, [8]Kitware, [9]UCSD, [10]PPPL, [11]UC Irvine, [12]U. Utah, [13] Caltech, [14]Auburn University, [15]NCSU



Outline

- Why care?
- What are we doing?
- High End Computing Trends.
- The long road towards yotta-scale computing
- Conclusions.
- Some of our papers from 2008 - 2011

Work supported under DOE funding:
ASCR: SDM Center, CPES, Runtime Staging, SAP, OLCF
OFES: GPSC, GSEP
NSF: HECURA, RDAV
NASA: (soon)

2011 I/O Pipeline Publications

1. S. Klasky, et al., “In Situ data processing for extreme-scale computing”, to appear SciDAC 2011.
2. A. Shoshani, et al., “The Scientific Data Management Center: Available Technologies and Highlights”, SciDAC 2011.
3. T. Critchlow, et al., “Working with Workflows: Highlights from 5 years Building Scientific Workflows”, SciDAC 2011.
4. S. Lakshminarasimhan, N. Shah, Stephane Ethier, Scott Klasky, Rob Latham, Rob Ross, Nagiza F. Samatova, “Compressing the Incompressible with ISABELA: In Situ Reduction of Spatio-Temporal Data”, Europar 2011.
5. S. Lakshminarasimhan, J. Jenkins, I. Arkatkar, Z. Gong, H. Kolla, S. H. Ku, S. Ethier, J. Chen, C.S. Chang, S. Klasky, R. Latham, R. Ross, , N. F. Samatova, “ISABELA-QA: Query-driven Analytics with ISABELA-compressed Extreme-Scale Scientific Data”, to appear SC 2011.
6. Y. Tian, S. Klasky, H. Abbasi, J. Lofstead, R. Grout, N. Podhorszki, Q. Liu, Y. Wang, W. Yu, “EDO: Improving Read Performance for Scientific Applications Through Elastic Data Organization”, to appear Cluster 2011.
7. K. Wu, R. Sinha, C. Jones, S. Ethier, S. Klasky, K. L. Ma, A. Shoshani, M. Winslett, “Finding Regions of Interest on Toroidal Meshes”, to appear in Journal of Computational Science and Discovery, 2011.
8. J. Lofstead, M. Polte, G. Gibson, S. Klasky, K. Schwan, R. Oldfield, M. Wolf, “Six Degrees of Scientific Data: Reading Patterns for extreme scale science IO”, HPDC 2011.
9. H. Abbasi, G. Eisenhauer, S. Klasky, K. Schwan, M. Wolf. “Just In Time: Adding Value to IO Pipelines of High Performance Applications with JITStaging, HPDC 2011.
10. C. Docan, J. Cummings, S. Klasky, M. Parashar, “Moving the Code to the Data – Dynamic Code Deployment using ActiveSpaces”, IPDPS 2011.
11. Y. Tian, “SRC: Enabling Petascale Data Analysis for Scientific Applications Through Data Reorganization”, ICS 2011, “First Place: Student Research Competition”
12. F. Zhang, C. Docan, M. Parashar, S. Klasky, “Enabling Multi-Physics Coupled Simulations within the PGAS Programming Framework”, CCgrid 2011.

2010 I/O Pipeline Publications

13. A. Shoshani, S. Klasky, R. Ross, “Scientific Data Management: Challenges and Approaches in the Extreme Scale Era”, SciDAC 2010.
14. J. Lofstead, F. Zheng, Q. Liu, S. Klasky, R. Oldfield, T. Kordenbrock, Karsten Schwan, Matthew Wolf. "Managing Variability in the IO Performance of Petascale Storage Systems". In Proceedings of SC 10. New Orleans, LA. November 2010.
15. C. Docan, S. Klasky, M. Parashar, “DataSpaces: An Interaction and Coordination Framework for Coupled Simulation Workflows”, HPDC’10. ACM, Chicago Ill.
16. H. Abbasi, M. Wolf, G. Eisenhauer, S. Klasky, K. Schwan, F. Zheng, “DataStager: scalable data staging services for petascale applications”, Cluster Computer, Springer 1386-7857, pp. 277-290, Vol 13, Issue 3, 2010.
17. F. Zheng, H. Abbasi, C. Docan, J. Lofstead, Q. Liu, S. Klasky, M. Parashar, N. Podhorszki, K. Schwan, M. Wolf, “PreDatA - Preparatory Data Analytics on Peta-Scale Machines”, IPDPS 2010, IEEE Computer Society Press 2010.
18. *Cummings, Klasky, Podhorszki, Barreto, Lofstead, Schwan, Docan, Parashar, Sim, Shoshani, “EFFIS: and End-to-end Framework for Fusion Integrated Simulation”, PDP 2010,* http://www.pdp2010.org/.
19. C. Docan, J. Cummings, S. Klasky, M. Parashar, N. Podhorszki, F. Zhang, “Experiments with Memory-to-Memory Coupling for End-to-End fusion Simulation Workflows”, ccGrid2010, IEEE Computer Society Press 2010.
20. Y. Xiao, I. Holod, W. L. Zhang, S. Klasky, Z. H. Lin, “Fluctuation characteristics and transport properties of collisionless trapped electron mode turbulence”, Physics of Plasmas, 17, 2010.
21. R. Tchoua, S. Klasky, N. Podhorszki, B. Grimm, A. Khan, E. Santos, C. Silva, P. Mouallem, M. Vouk, "Collaborative Monitoring and Analysis for Simulation Scientist", in Proceedings of CTS 2010.
22. C. Docan, M. Parashar, S. Klasky: Enabling high-speed asynchronous data extraction and transfer using DART. Concurrency and Computation: Practice and Experience 22(9): 1181-1204 (2010)

2009 I/O Pipeline Publications

24. N. Podhorszki, S. Klasky, Q. Liu, C. Docan, M. Parashar, H. Abbasi, J. Lofstead, et al.: Plasma fusion code coupling using scalable I/O services and scientific workflows. SC-WORKS 2009

25. P. Mouallem, R. Barreto, S. Klasky, N. Podhorszki, M. Vouk. 2009. Tracking Files in the Kepler Provenance Framework. In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management* (SSDBM 2009), Marianne Winslett (Ed.). Springer-Verlag, Berlin, Heidelberg, 273-282

26. J. Lofstead, F. Zheng, S. Klasky, K. Schwan, "Adaptable Metadata Rich IO Methods for Portable High Performance IO", IPDPS 2009, IEEE Computer Society Press 2009.

27. H. Abbasi, Wolf, M., Eisenhauer, G., Klasky, S., Schwan, K., , Zheng, F. 2009. DataStager: scalable data staging services for petascale applications. In *Proceedings of the 18th ACM international Symposium on High Performance Distributed Computing* (Garching, Germany, June 11 - 13, 2009). HPDC '09. ACM, New York, NY, 39-48.

28. H. Abbasi, J. Lofstead, F. Zheng, S. Klasky, K. Schwan, M. Wolf, "Extending I/O through High Performance Data Services", Cluster Computing 2009, New Orleans, LA, August 2009.

29. R. Barreto, S. Klasky, N. Podhorszki, P. Mouallem, M. Vouk: "Collaboration Portal for Petascale Simulations", International Symposium on Collaborative Technologies and Systems, pp. 384-393, Baltimore, Maryland, May 2009.

30. M. Polte, J. Lofstead, J. Bent, G. Gibson, S. Klasky, "...And Eat it Too: High Read Performance in Write-Optimized HPC I/O Middleware File Formats," in *In Proceedings of Petascale Data Storage Workshop 2009 at Supercomputing 2009*, ed, 2009.

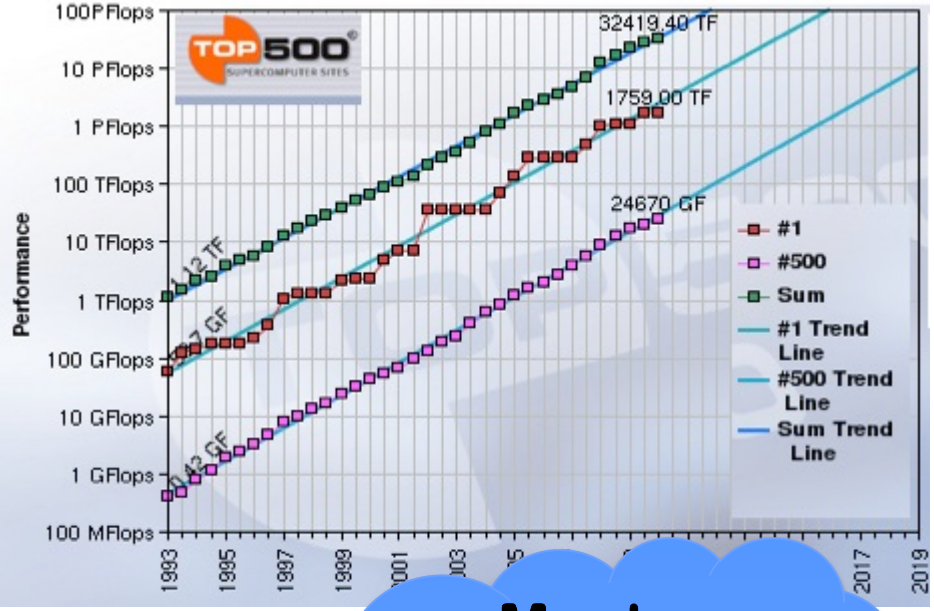
31. S. Klasky, et al., "High Throughput Data Movement,", "Scientific Data Management: Challenges, Existing Technologies, and Deployment," Editors: A. Shoshani and D. Rotem, Chapman and Hall,, 2009

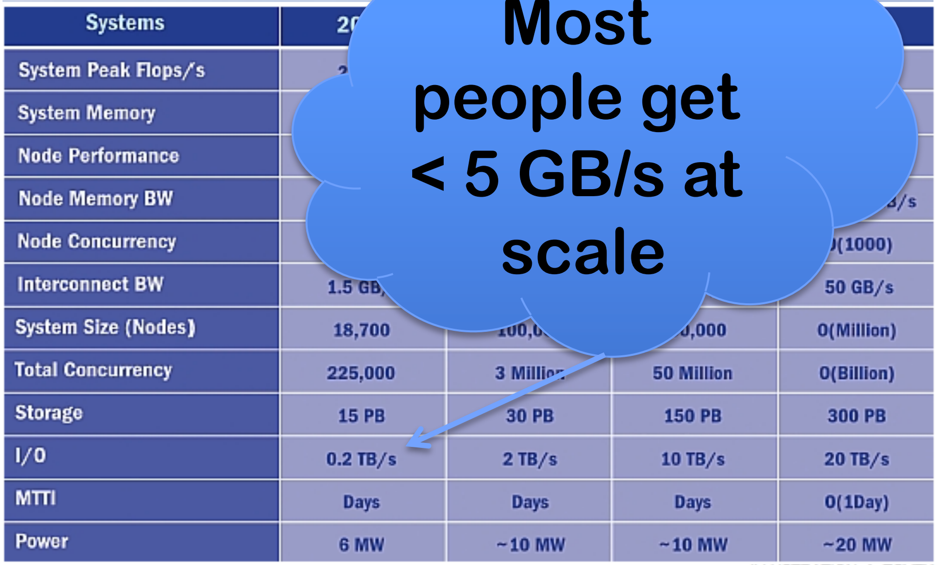
2008 I/O Pipeline Publications

32. Lofstead, Zheng, Klasky, Schwan, "Input/Output APIs and Data Organization for High Performance Scientific Computing", PDSI SC2008.
33. C. Docan, M. Parashar, S. Klasky, "Enabling High Speed Asynchronous Data Extraction and Transfer Using DART," *Proceedings of the 17th International Symposium on High-Performance Distributed Computing (HPDC)*, Boston, MA, USA, IEEE Computer Society Press, June 2008.
34. J. Lofstead, S. Klasky, K. Schwan, N. Podhorszki, C. Jin, "Flexible IO and Integration for Scientific Codes Through the adaptable IO System", Challenges of Large Applications in Distributed Environments (CLADE), June 2008.
35. Klasky, et al., "Collaborative Visualization Spaces for Petascale Simulations", to appear in the 2008 International Symposium on Collaborative Technologies and Systems (CTS 2008).
36. S. Klasky, C. Jin, S. Hodson, T. White, W. Yu, J. Lofstead, K. Schwan, M. Wolf, W. Liao, A. Choudhary, M. Parashar, C. Docan, "Adaptive IO System (ADIOS)", Cray User Group Meeting 2008.
37. H. Abbasi, M. Wolf, K. Schwan, S. Klasky, "Managed Streams: Scalable I/O for", HPDC 2008.

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	[illegible]	[illegible]	[illegible]
System Peak Flops/s	2	[illegible]	[illegible]	[illegible]
System Memory	[illegible]	[illegible]	[illegible]	[illegible]
Node Performance	[illegible]	[illegible]	[illegible]	[illegible]
Node Memory BW	[illegible]	[illegible]	[illegible]	B/s
Node Concurrency	[illegible]	[illegible]	[illegible]	(1000)
Interconnect BW	1.5 GB	[illegible]	[illegible]	50 GB/s
System Size (Nodes)	18,700	100,0	,000	O(Million)
Total Concurrency	225,000	3 Millio	50 Million	O(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	O(1Day)
Power	6 MW	~10 MW	~10 MW	~20 MW

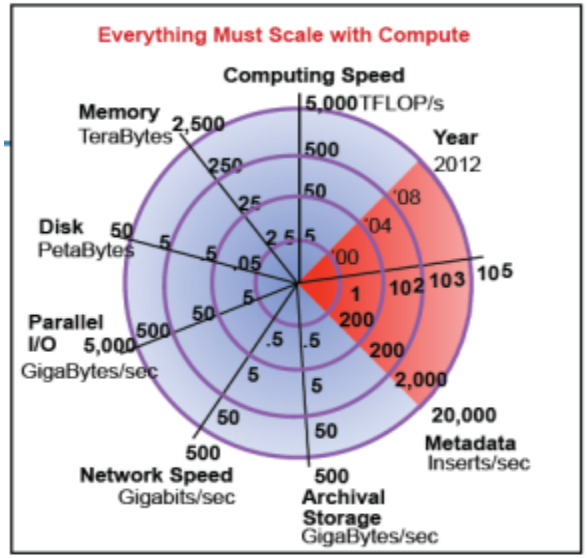
ILLUSTRATION: A. TOVEY

Most people get < 5 GB/s at scale

From J. Dongarra, “Impact of Architecture and Technology for Extreme Scale on Software and Algorithm Design,” Cross-cutting Technologies for Computing at the Exascale, February 2-5, 2010.

File System, Problems for the Xscale

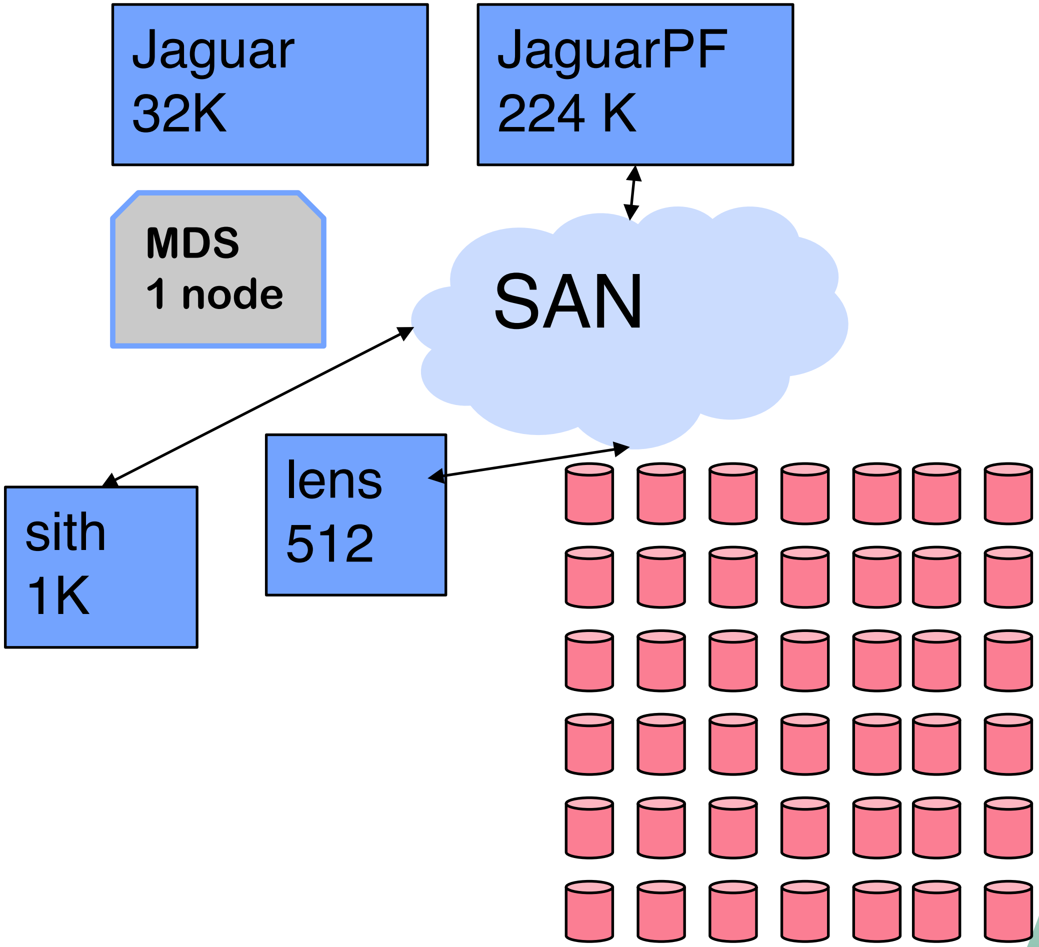
- The I/O on a HPC system is stressed because
  - Checkpoint-restart writing
  - Analysis and visualization writing
  - Analysis and visualization reading
- Our systems are growing by 2x FLOPS/year.
- Disk Bandwidth is growing ~20%/year.
- Need the number of increase faster than the number of nodes
- 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 XB+ of data.
  - Can you analyze this?
  - Can you visualize this?
  - Can you read the data?

Problems to face for the exascale

- I/O will have to be dramatically reduced (output data/ Flops).
- Applications, debugging, Visualization, Analytics, must be tied to I/O
- Challenge is to reduce the impact of I/O on "real" calculations.
- Forces us to rethink I/O.
- File formats must meet new challenges for these challenges.
  - Need to be "unbiased", "reduce network and I/O cost"
- Allow scientist to "plug-in" analytics, into I/O pipelines.
  - Make "plug-ins" **crash proof** to the application.

Requirements for our framework

- Provide the software infrastructure to enable a diverse set of fusion scientists the ability to compose, run, couple, debug, monitor, analyze, and automate the tracking of fusion codes through common standards and easy-to-use interfaces.
- Individual computational tasks may range from codes running on workstations to leadership-class computers.
- Scientists need access to a software infrastructure that can span the full range of resources needed by the science in one coherent framework.

Design Philosophy

- The overarching design philosophy of the framework is based on the Service-Oriented Architecture for software
  - Has been successfully used by enterprise software systems to deal with system/application complexity, rapidly changing requirements, rapidly evolving target platforms, and diverse development teams.
- Software systems and applications are constructed by assembling services based on a universal view of their functionality using a well-defined API.
- Services and their implementations can be changed easily, and workflows can be customized to fit application requirements.
- A fusion simulation code can be assembled using physics, math, and computer science service realizations such as solver libraries, I/O services, partitioners, and communication services, which are created independently.
- Integrated simulation systems can be assembled using these codes as well as coupling and data-movement services
- End-to-end application workflows can be constructed by composing the coupled systems with services for data visualization, archiving, analysis, code verification, etc.

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

SOA Scales

- e.g., Yahoo Data Challenges – sound familiar?

  Data Challenges at Yahoo! Ricardo Baeza-Yates & Raghu Ramakrishnan, Yahoo! Research

  - Data diversity – text (tagged/non-tagged), streams, structured data (i.e., formatted), multimedia (us: checkpoints, analysis, coupling, analysis results/ dashboard displays-graphs, …)
  - Rich set of processing – not just database queries (SQL), but analytics (transformation, aggregation, …)
  - Attain scale (350K requests/sec! and growing) via asynchrony, loose coupling, weak consistency (us: decoupling via ADIOS, data staging, …)
  - Leverage file system's high bandwidth (us: Lustre vs. them: DFS++)
  - Use multiple ways to represent data (us: BP, tuple spaces, …; them: row/ column stores, DHTs)
  - Deal with reliability (us: robust data format , checkpointing; them: DFS-based replication/recoverability)
  - Make it easy to use: self-management, self-tuning (us: adaptive I/O)
  - Make it easy to change: adaptability, i.e., new analyses readily added (us: that's the whole point of the SOA)

- If Yahoo and Google can do it, so can we!

K. Schwan

The “early days”: 2001, Reduce I/O overhead for 1 TB data.

- 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 data out on the receiving side.
  - Visualize the data near-real-time
  - **Focus on the 5% rule..**

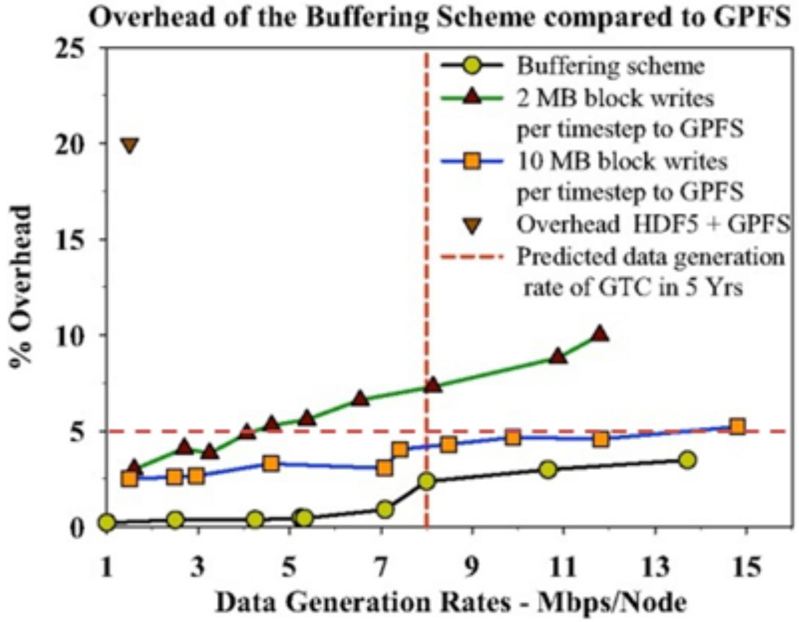
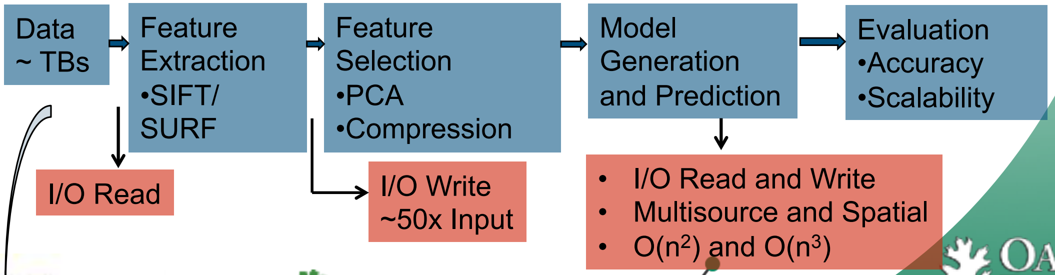


Figure: 8. Overhead with Buffering Scheme compared to GPFS (I/O).

Problems people come up to me and ask for a solution.

- Reduce the variability of I/O, and reduce the time spent writing.
- Reduce the I/O time for my post processing.
- Let me couple codes (memory/file).
- “Plug-in” my visualization code, with no changes.
- Latest challenge:
  - Read an process 12M images (2 TB) on a LCF, and write 50X the data.
  - Problem:
    - Read image (0.2 MB), process image in 10 seconds, write out 10 MB
    - Work on 100K cores
    - For I/O to <5 %, need open, read + write, close, <0.5 s



Data
~ TBs

Feature Extraction
•SIFT/ SURF

Feature Selection
•PCA
•Compression

Model Generation and Prediction

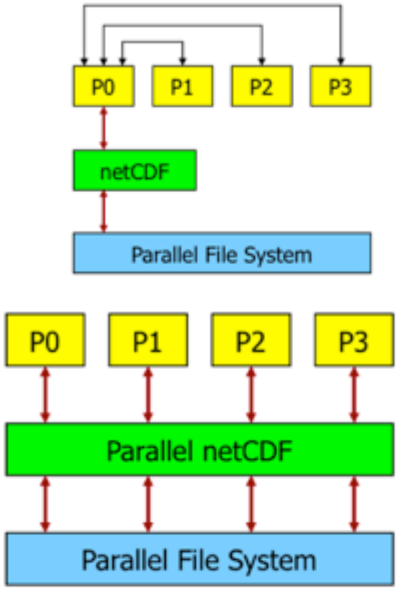
Evaluation
•Accuracy
•Scalability

I/O Read

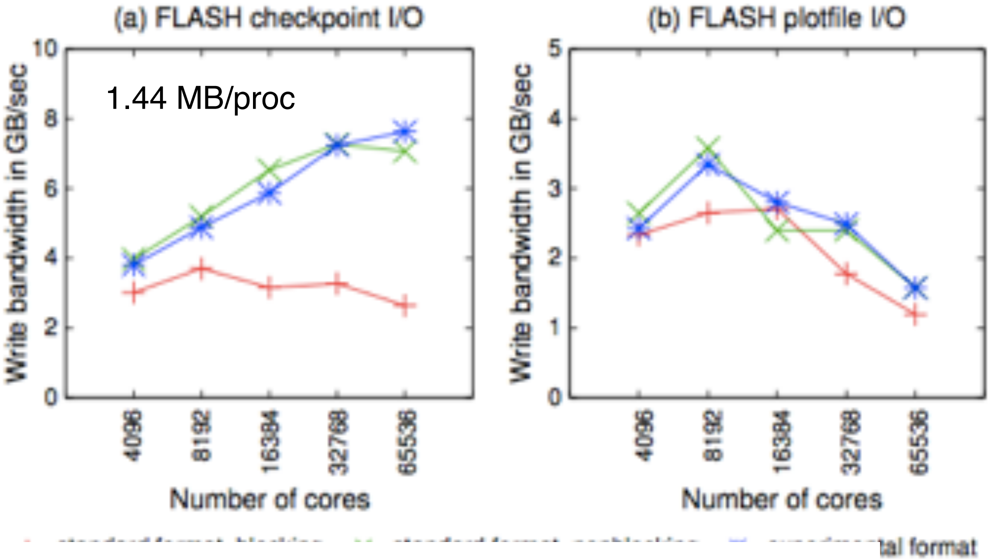
I/O Write
~50x Input

- I/O Read and Write
- Multisource and Spatial
- $O(n^2)$ and $O(n^3)$

Parallel netCDF

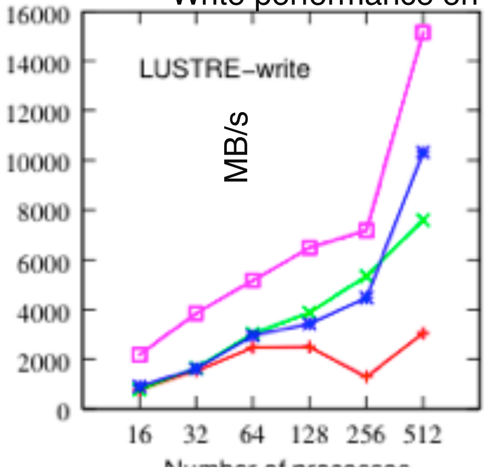


- http://trac.mcs.anl.gov/projects/parallel-netcdf
- New file format to allow for large array support
- New optimizations for non-blocking calls.
- New optimizations for sub-files.
- Idea is to allow netcdf to work in parallel and for large files and large arrays.



http://www.mcs.anl.gov/uploads/cels/papers/P1819.pdf

Write performance on Franklin



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

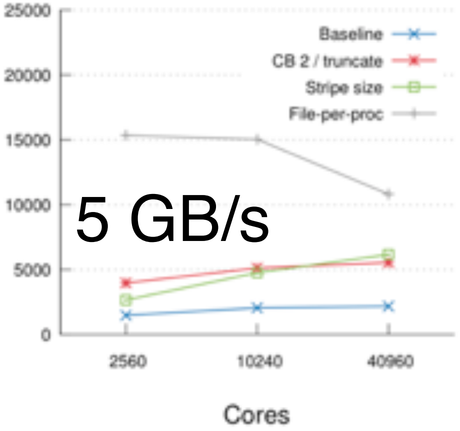
HDF5

- 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)

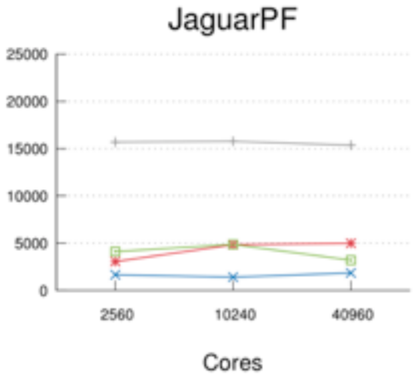
Parallel netCDF-4/ HDF5

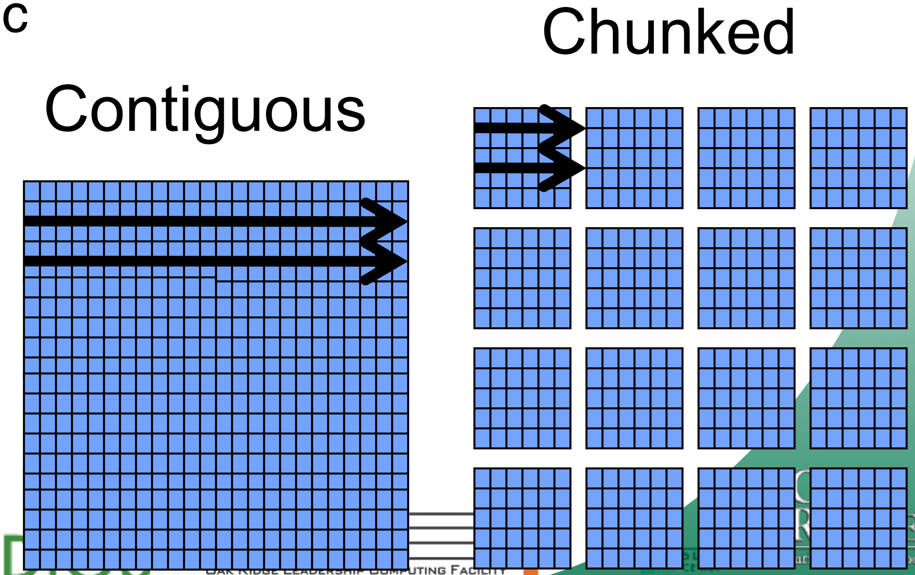
- http://www.unidata.ucar.edu/software/netcdf/
- Use HDF5 for the file format.
- Keep backward compatibility in tools to read netCDF 3 files.
- HDF5 optimized chunking
- New journaling techniques to handle resiliency.
- Many other optimizations
  - http://www.hdfgroup.org/pubs/papers/howison_hdf5_lustre_iasds2010.pdf

Vorpal, 40^3: 1.4 MB/proc, 11.5 MB/proc JaguarPF



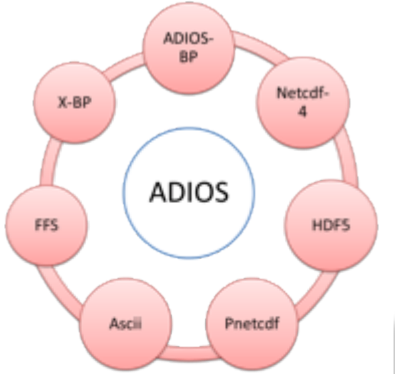
5 GB/s

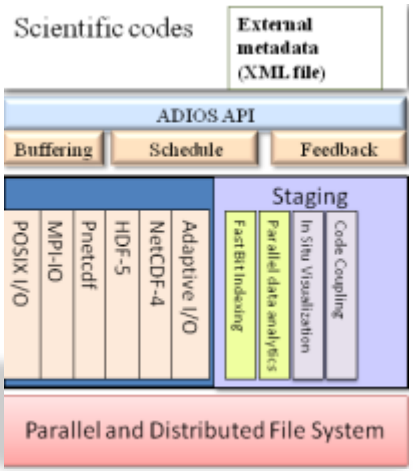




ADIOS: Adaptable I/O System

- 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)
- Open source:
  - http://www.nccs.gov/user-support/center-projects/adi
- Research methods from many groups:
  - Rutgers: DataSpaces/DART , Georgia Tech: DataTap, Sandia: NSSI, Netcdf-4, ORNL: MPI_AMR





ADIOS 1.2 is Here!
easy-to-use, fast, scalable, and portable I/O

Below are links for the Adios download and user manual. For additional support, please contact help@nccs.gov.

Download

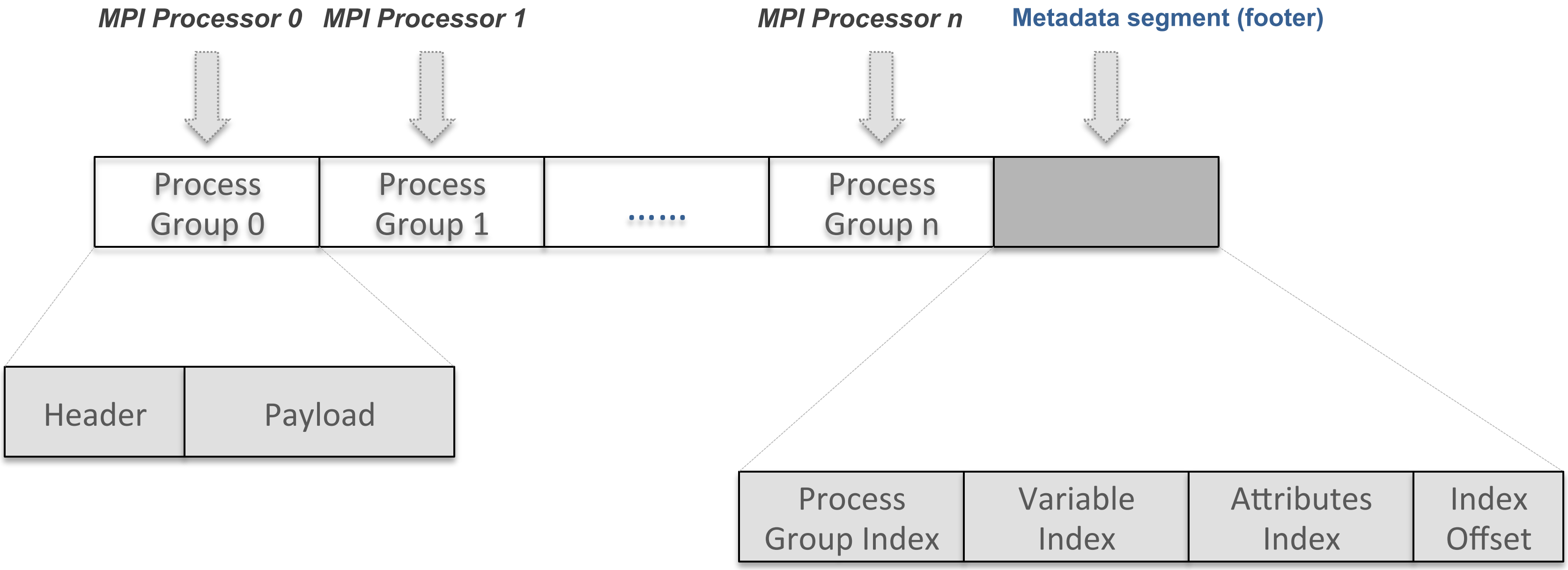
Current stable version: ADIOS 1.2 (Download 1.2)

Revision History

1.2 Released July 15, 2010

- XML and non-XML write APIs are available
- More statistics (min/max/avg/std-dev/histograms) available without overhead in write performance
- Added MPI_AMR method for advanced mesh refinement codes and for aggregated writing (N procs to P writers to M disks)
- Added support for subfiles
  - POSIX method many-file output can be read as one file
  - MPI_AMR method uses subfiles to improve write performance
- Added NetCDF-4 transport method
- Asynchronous, staging methods on Jaguar XT5 at ORNL
  - DataTap from Georgia Tech
  - DataSpaces from Rutgers
  - NSSI from Sandia
- MPI_LUSTRE method for best performance on Lustre file system

ADIOS BP File Format

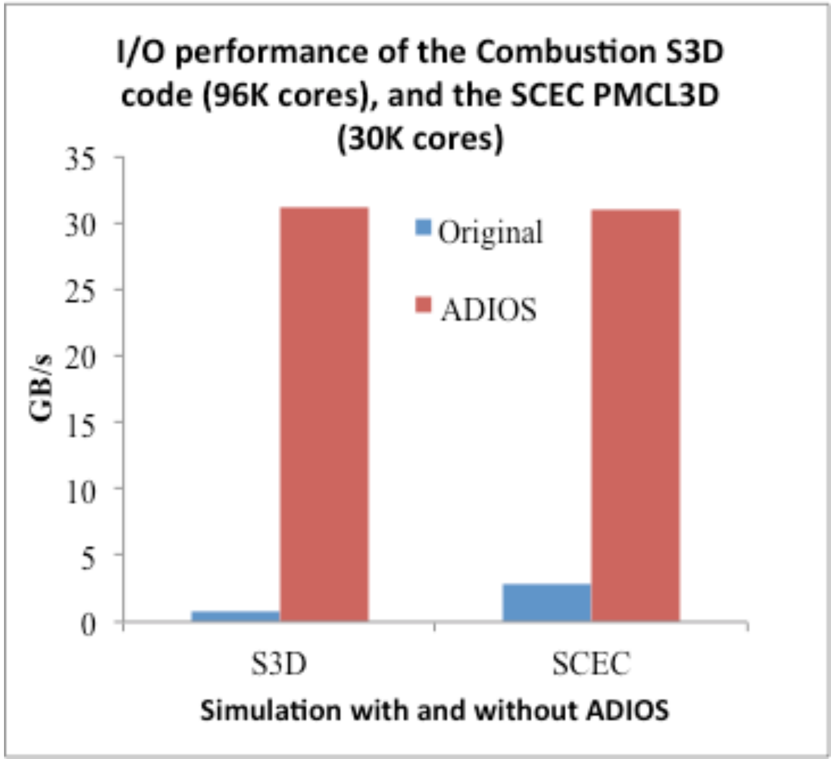


1) ADIOS BP File Format – single file case

- Fault tolerance is critical for success of a parallel file format.
- Failure of a single writer is not fatal.
- Necessary to have a hierarchical view of the data (like HDF5).
- Tested at scale (140K processors for XGC-1) with over 20TB in a single file.

ADIOS 1.2 write speeds

- Synchronous write speeds:
- S3D: 32 GB/s with 96K cores, 1.9MB/core: 0.6% I/O overhead.
- 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)



How to use ADIOS to write out data

- User Fortran code

```
call adios_open (adios_handle, "particles”, fname, "w“, comm, err)
#include "gwrite_particles.fh“ (automatically genereated)
call adios_close (adios_handle,err)
```

- Sample XML file

```
<adios-group name="particles" coordination-communicator="comm">
    <var name="mype" type="integer"/>
    <var name="nparam" type="integer"/>
    <var name="nspecies" type="integer"/>
    <var name="numberpe" type="integer"/>
    <var name="nparam*numberpe" type="integer" />
    <var name="nparam*mype" type="integer" />
    <var name="ntracke" type="integer" gwrite="ntrackp(2)"/>
<global-bounds dimensions="nparam*numberpe,ntracke" offsets="nparam*mype,0">
    <var name=“electrons" type="real" dimensions="nparam,ntracke"
   gwrite="ptrackede(:,1:ntrackp(2))"/>
</global-bounds>
</adios-group>
<transport method=“MPI" group="particles"/>
```

ADIOS MPI_LUSTRE Method

- Improved version of MPI method.
- The file is written out with Lustre stripe-aligned.
- Automatically set Lustre I/O parameters from XML file.

i.e., stripe count, stripe size and write block size.

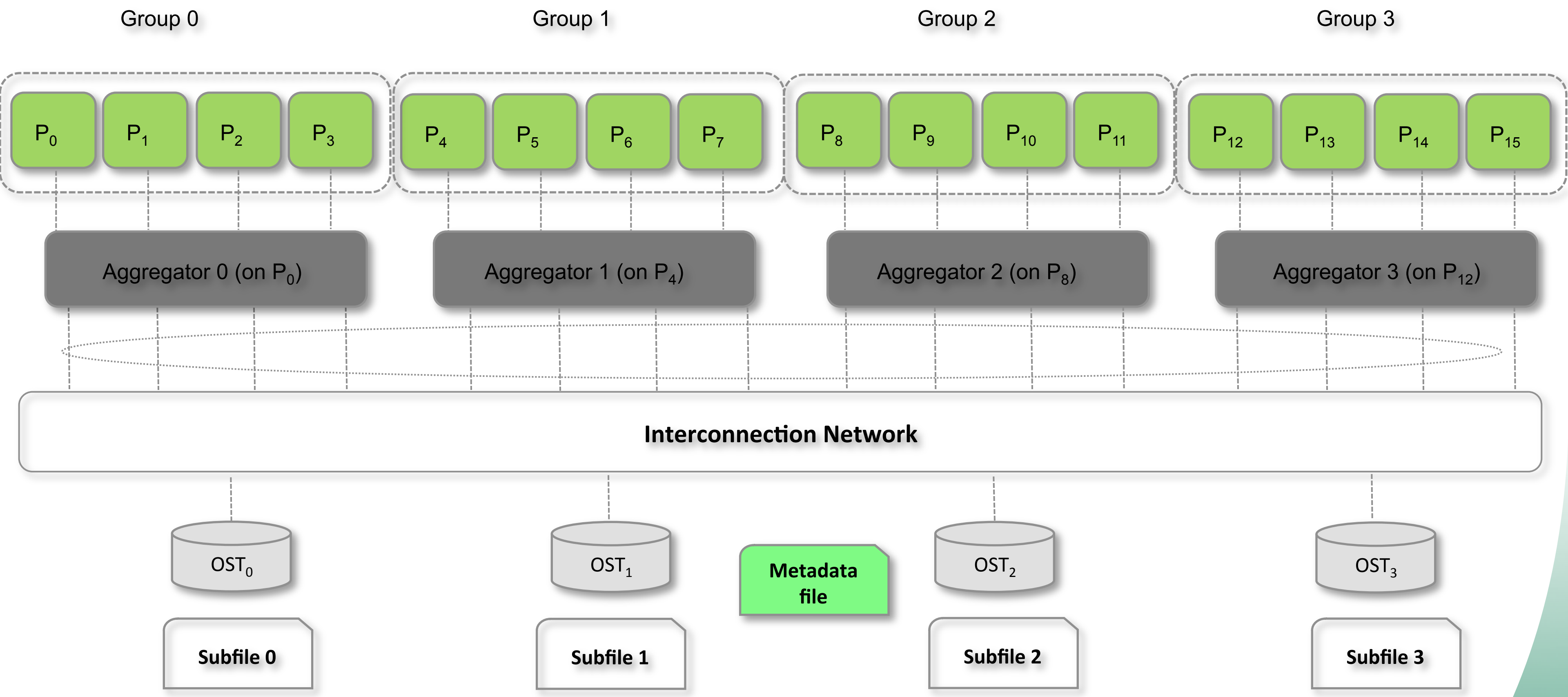
For example, to stripe your file on 16 OST’ s with stripe size 4MB and write block size 512KB,

```
<method group="temperature" method="MPI_LUSTRE">
   stripe_count=16;stripe_size=4194304;block_size=524288
</method>
```

ADIOS MPI_AMR Method

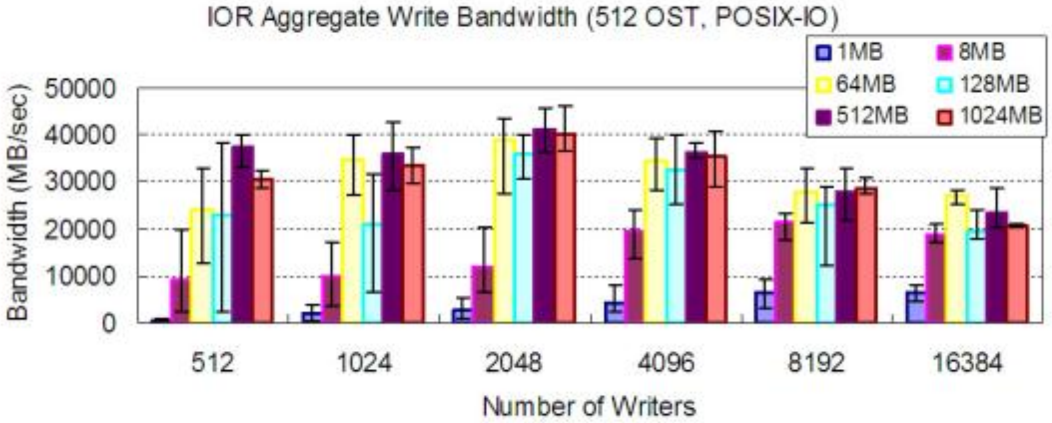
- Newly developed method that further improves IO performance on Lustre Parallel File System
- Key improvements:
  - Eliminate lock contention: Write out multiple subfiles with each file striped on 1 storage target (OST)
  - Aggregate IO among processors: Selected processors gathers all the data and write them out in a big chunk
  - Threaded file opens: Simulation can continue while waiting for file to be opened
  - Good usability: Other than telling ADIOS the # of aggregators to use, everything is the same as writing/reading one file to users.

ADIOS MPI_AMR Method
How does it work?



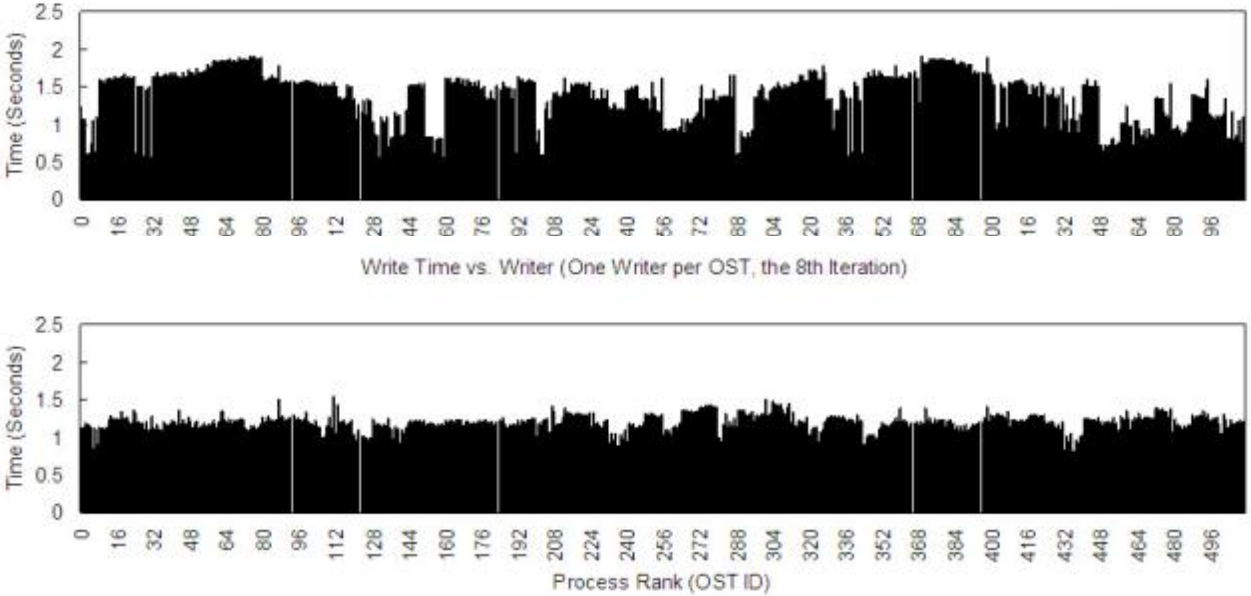
I/O interference

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



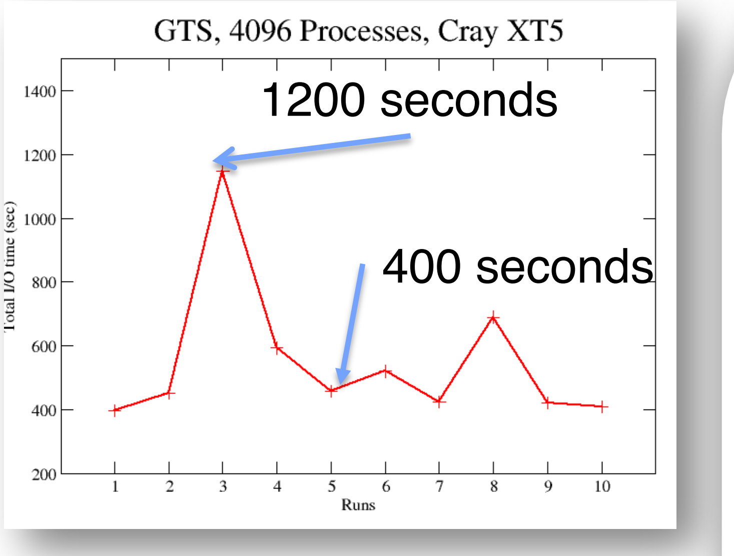
3.44 vs. 1.86 imbalance factor

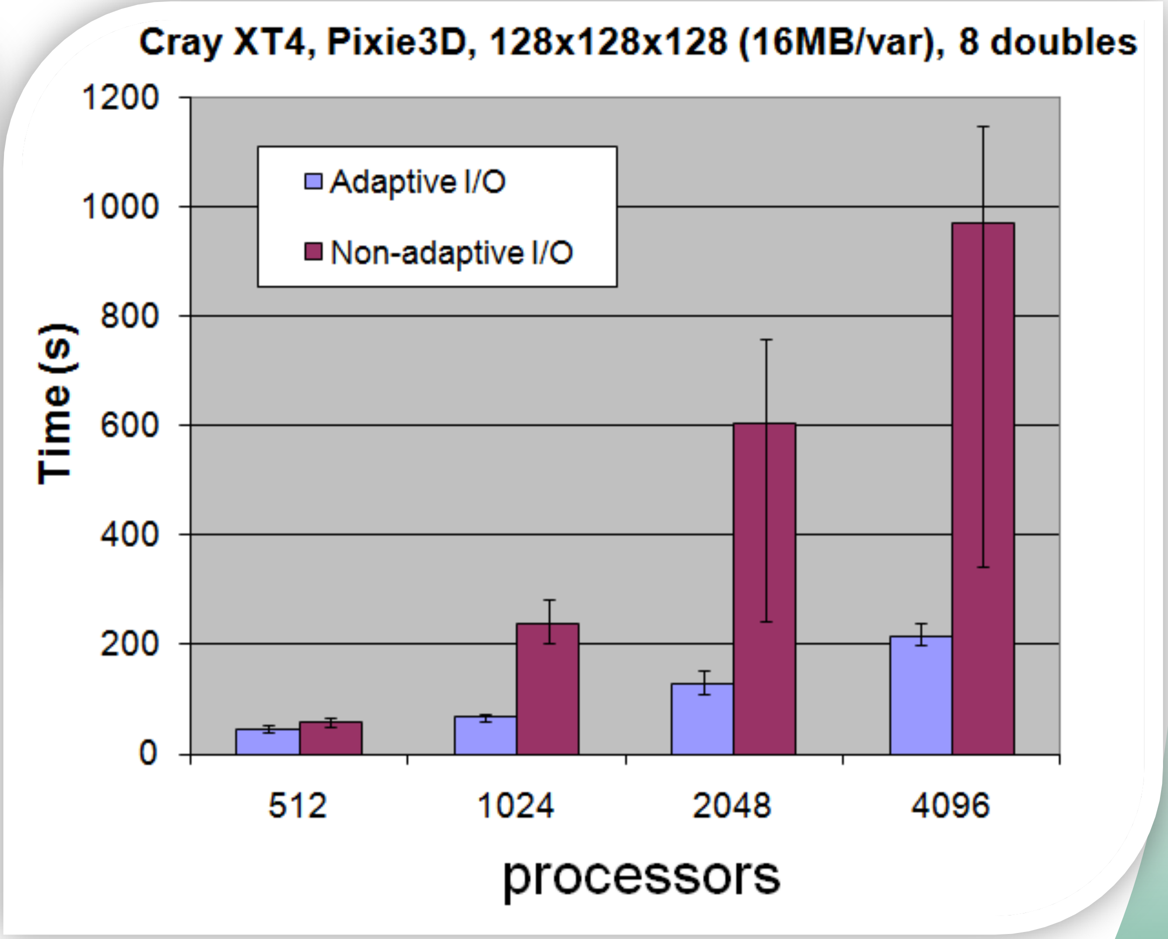
128 MB/process,
3 minutes apart



Write Time vs. Writer (One Writer per OST, the 8th Iteration)

Variability in I/O is a reality.

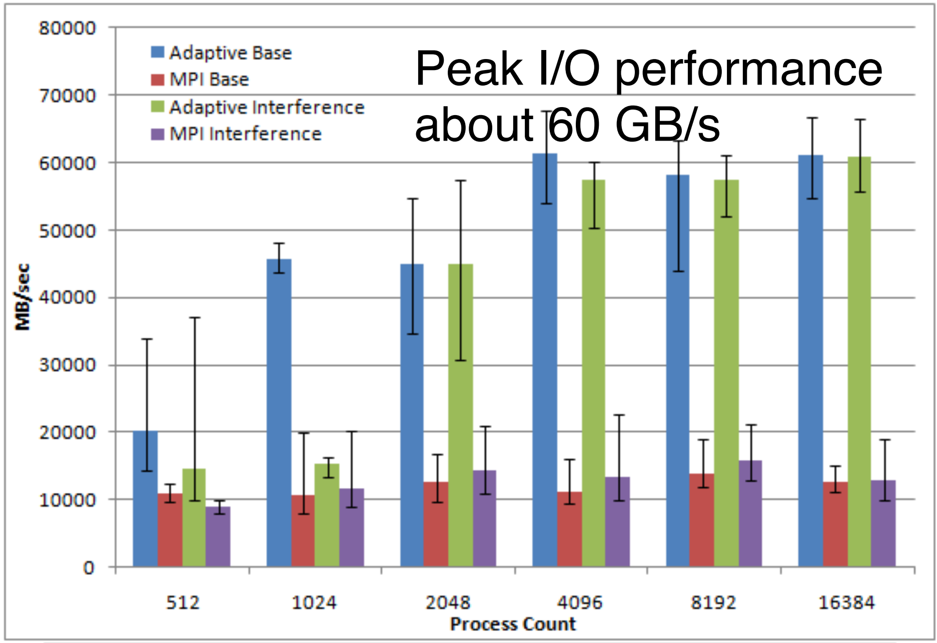




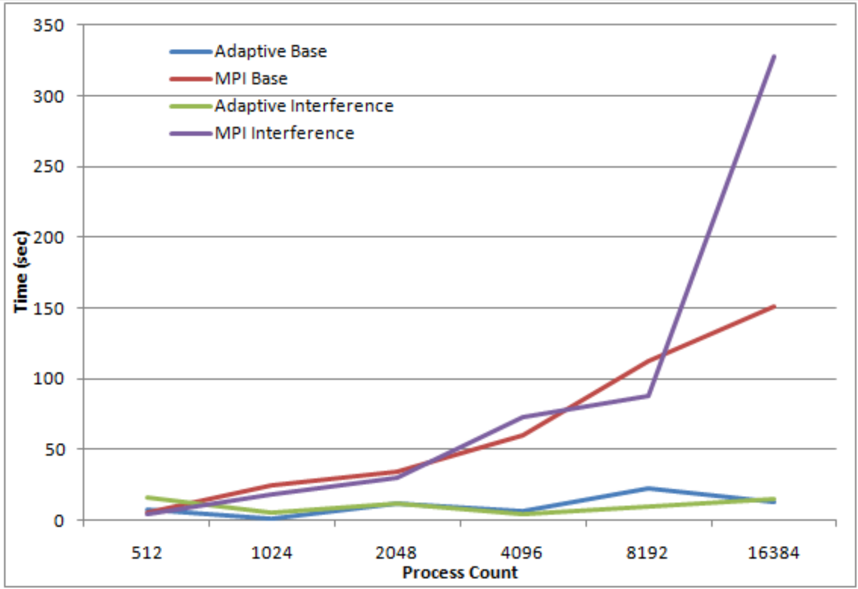
- Application scientist want consistent results.
- Minimize network and file system congestion.

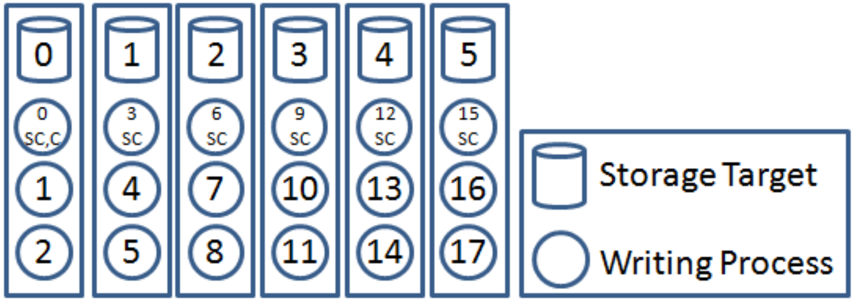
Reduce the variability of I/O

- 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.



Peak I/O performance about 60 GB/s





Storage Target

Writing Process

bpls (can extract any portion of data).

- **$ time bpls -l record.bp -v**

```
  of groups:     1
  of variables:  32
  of attributes: 0
  time steps:    10 starting from 1
  file size:     162 GB
  bp version:    1
Group record:
  double   /time              {10} = 0.003 / 0.03
  integer  /itime             {10} = 3 / 30
  double   /dt                {10} = 0.001 / 0.001
  integer  /nvar              scalar = 8
  integer  /dimensions/nxd+2  scalar = 1026
  integer  /dimensions/nyd+2  scalar = 514
  integer  /dimensions/nzd+2  scalar = 514
  double   /var/v1            {10, 514, 514, 1026} = 1 / 1
  double   /var/v2            {10, 514, 514, 1026} = -2.07946e-06 / 3.43263e-08
  double   /var/v3            {10, 514, 514, 1026} = -1.17581e-10 / 1.24015e-10
  double   /var/v4            {10, 514, 514, 1026} = -3.65092e-13 / 3.65092e-13
  double   /var/v5            {10, 514, 514, 1026} = -7.95953e-11 / 7.95953e-11
  double   /var/v6            {10, 514, 514, 1026} = -0.184178 / 0.0123478
  double   /var/v7            {10, 514, 514, 1026} = -0.000488281 / 0.984914
  double   /var/v8            {10, 514, 514, 1026} = 0 / 0
  byte     /name/v1_name      {20} = 32 / 111
  byte     /name/v2_name      {20} = 32 / 94
  byte     /name/v3_name      {20} = 32 / 94
  byte     /name/v4_name      {20} = 32 / 94
  byte     /name/v5_name      {20} = 32 / 94
  byte     /name/v6_name      {20} = 32 / 94
  byte     /name/v7_name      {20} = 32 / 94
  byte     /name/v8_name      {20} = 32 / 101
  integer  /bconds            {48} = -4 / 7
real   0m2.091s
```

ADIOS Read API

1. open restart.bp file

```
ADIOS_FILE * f = adios_fopen (“restart.bp”, MPI_COMM_WORLD);
```

2. open a ADIOS group called “temperature”

```
ADIOS_GROUP * g = adios_gopen (f, “temperature”);
```

3. inquire the variable you want to read by its ID

```
for (i = 0; i < g->vars_count; i++) {
        ADIOS_VARINFO * v = adios_inq_var_byid (g, i);
}
```

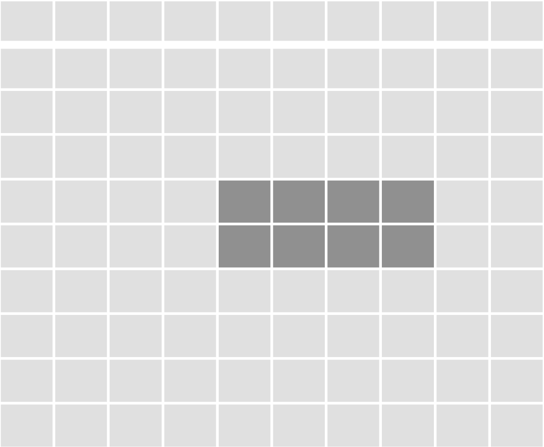
or a more common way is to inquire var by its name

```
ADIOS_VARINFO * v = adios_inq_var (g, “v2”);
```

4. read data

```
bytes_read = adios_read_var (g, “v2”, start, count, data);
```

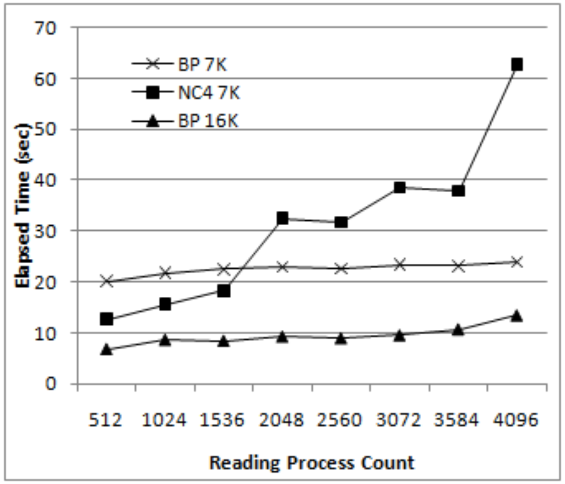
10x10 2D array

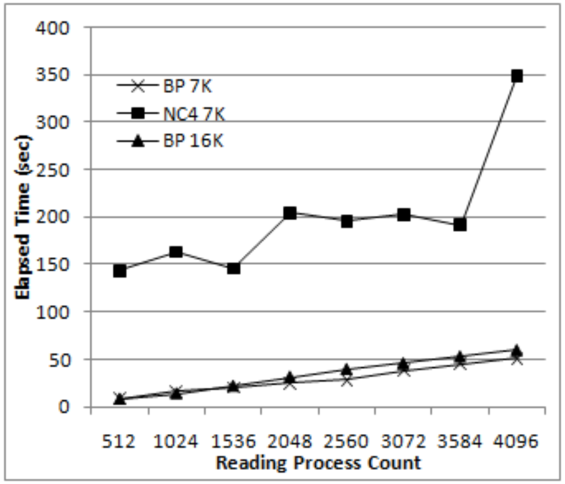


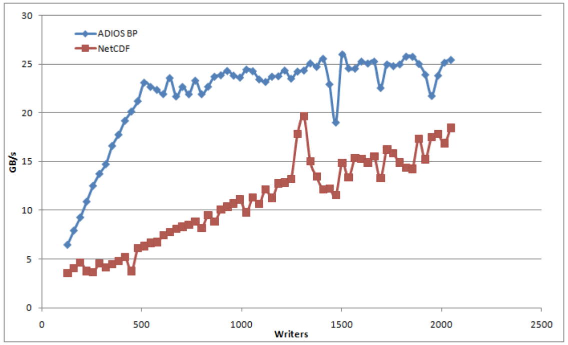
```
start[0]  = 4;
start[1]  = 4;
count[0] = 2;
count[1] = 4;
```

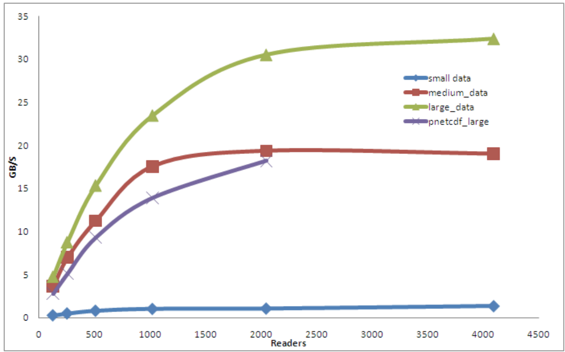
What about Read performance from ADIOS-BP?

- 4 papers: simple conclusion.
  - Chunking has a profound effect on read performance.



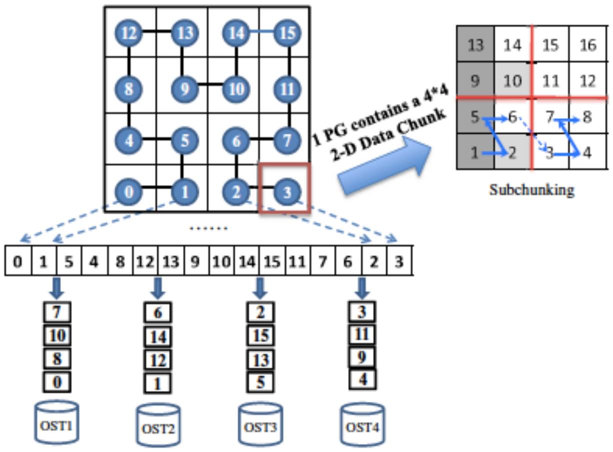




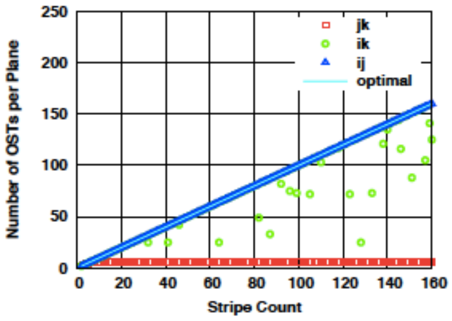


But Why? (Look at reading 2D plane from 3D dataset)

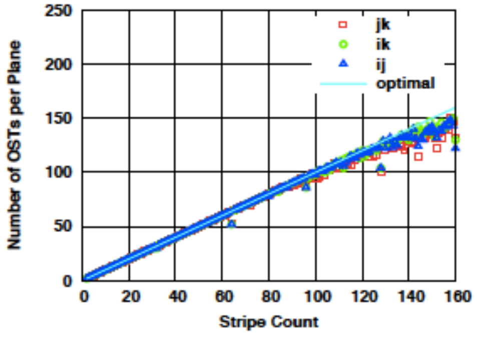
- Use Hilbert curve to place chunks on lustre file system with an Elastic Data Organization.



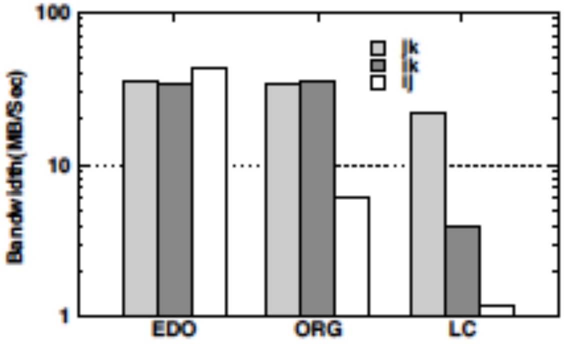
(b) SFC-based Placement with EDO



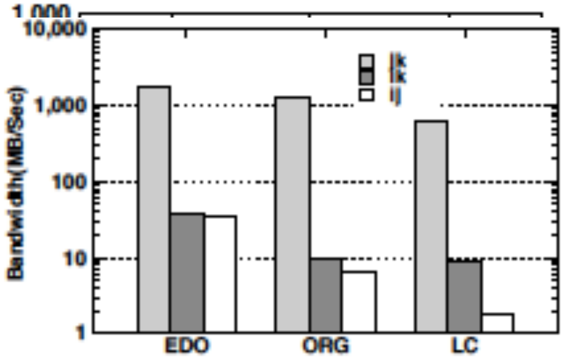
(a) Logically Contiguous



(c) Hilbert Curve



(a) Case S (stripe=128)

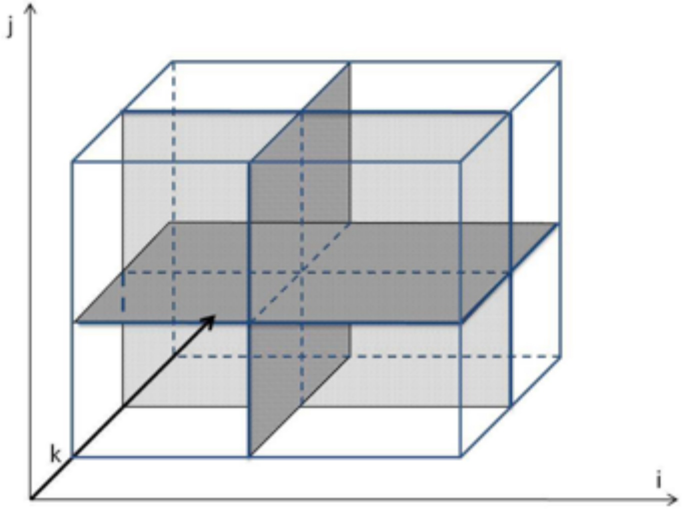


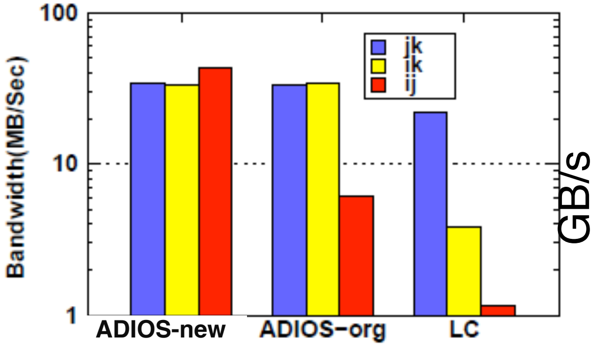
(b) Case X (stripe=128)

Six “degrees” of scientific data: Reading Patterns for Extreme scale data.

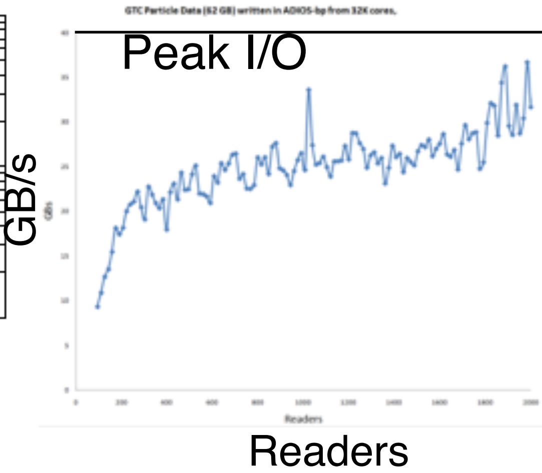
- Read all of the variables from an integer multiple of the original number of processors.
  - Example: restart data.
- Read in just a few variables on a small number of processors.
  - Visualization
- Read in a 2D slice from a 3D dataset (or lower dimensional reads) on a small number of processors.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

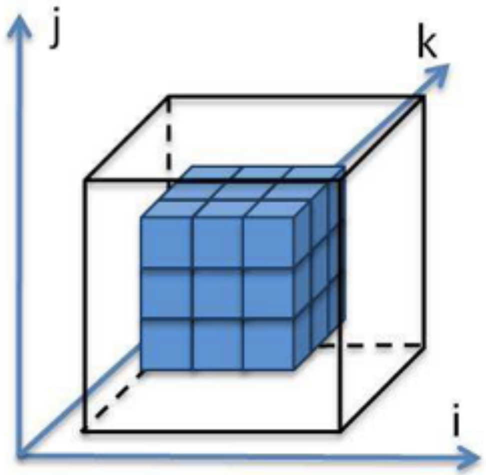


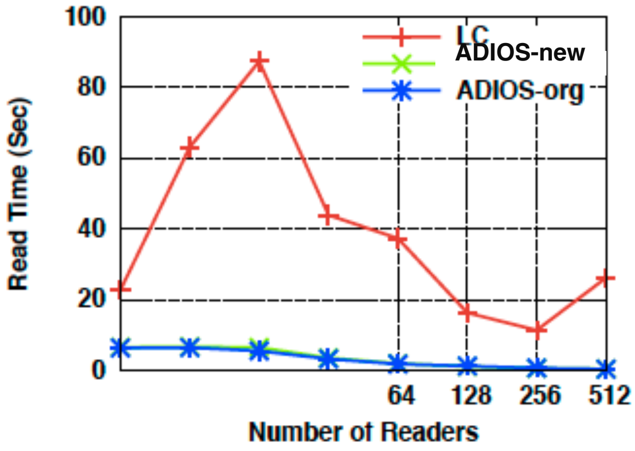


(a) Small(stripes=128)

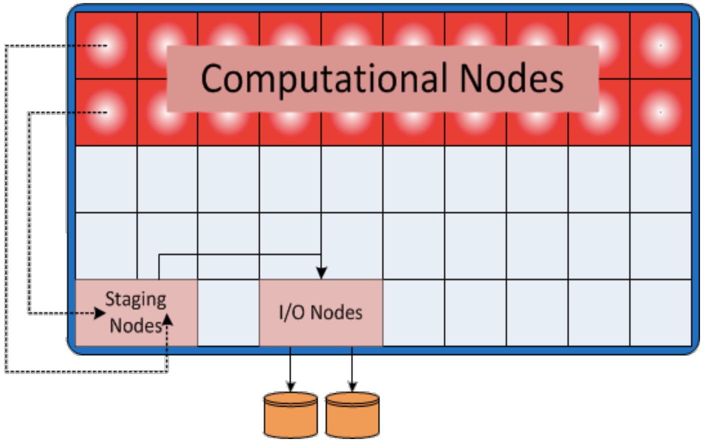


Read speed
For GTC on Jaguar



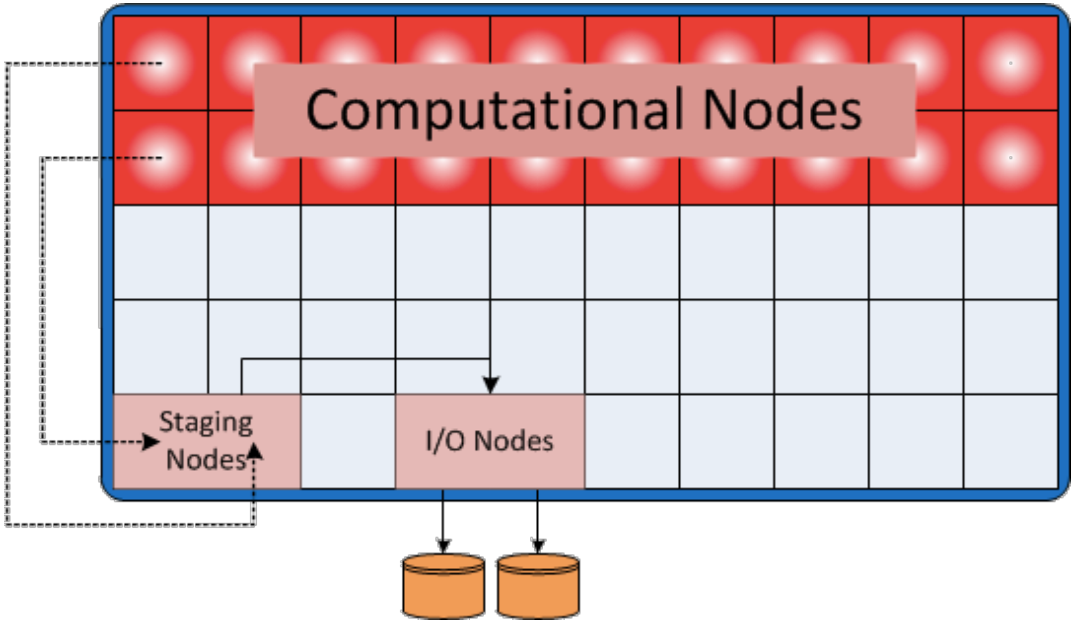


New methods to read data in ADIOS 1.3



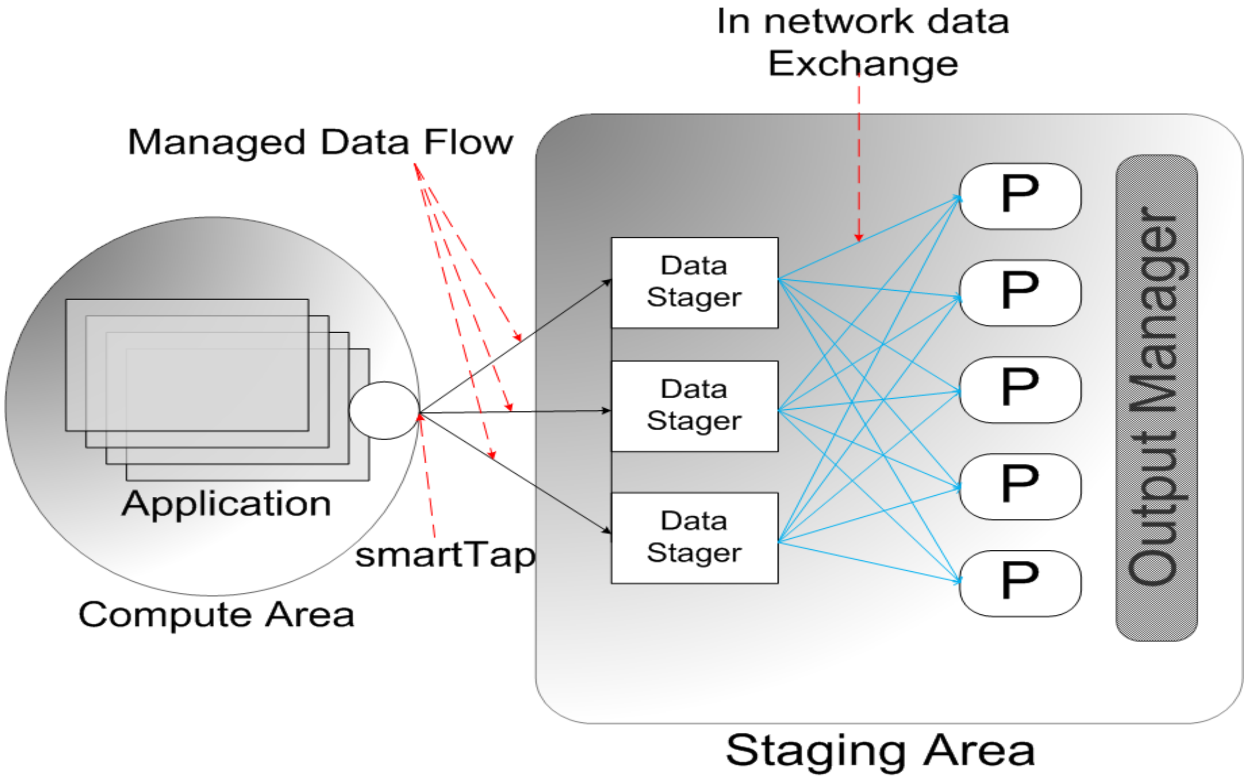
- Stage reads, and reduce the number of “readers”.
- Initial results when using “real” S3D data indicate 12X improvement of reading analysis data from arbitrary number of processors with sub-files.

Staging I/O



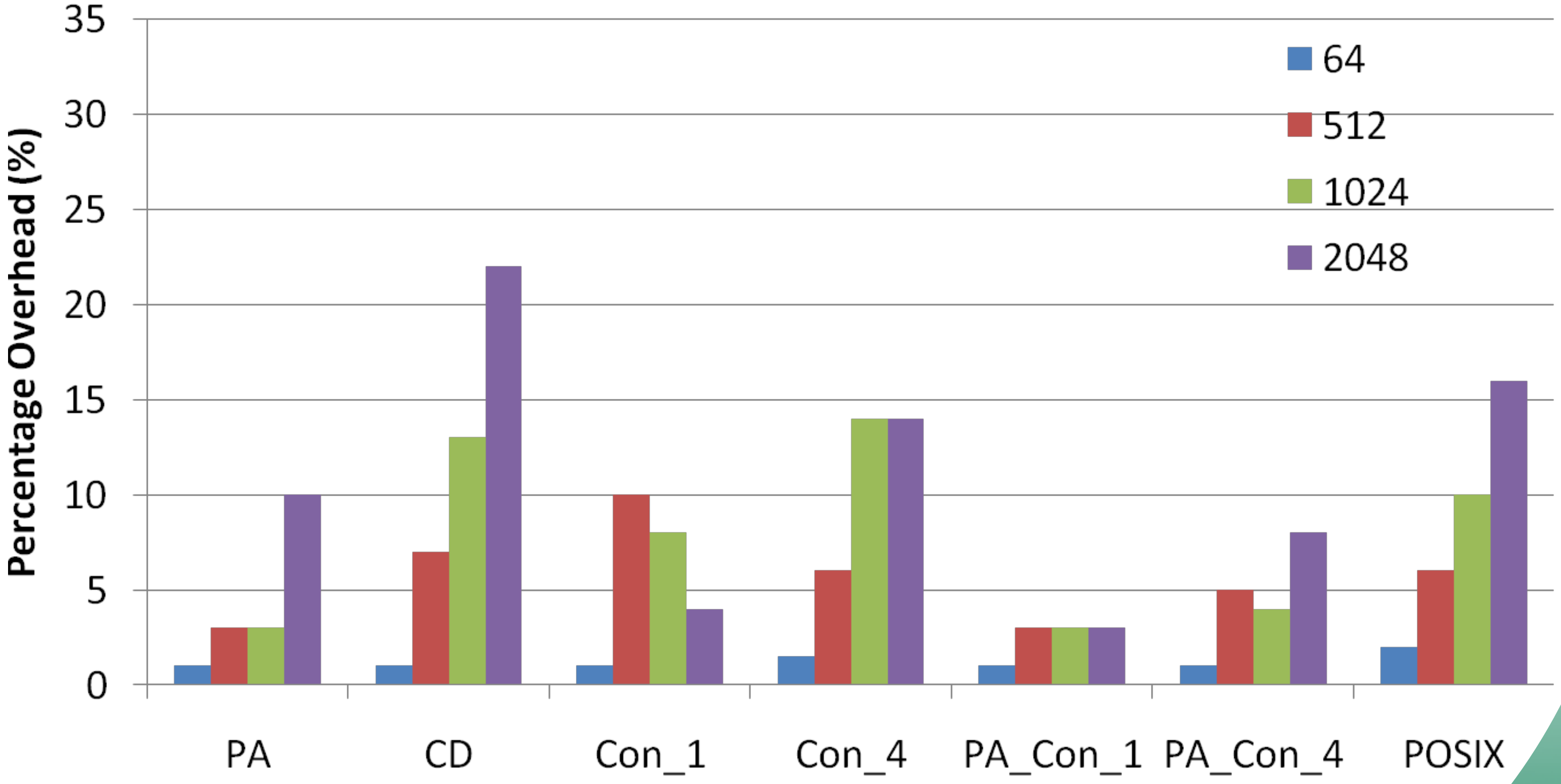
- Why asynchronous I/O?
  - Reduces performance linkage between I/O subsystem and application
  - Decouple file system performance variations and limitations from application run time
- Enables optimizations based on dynamic number of writers
- High bandwidth data extraction from application
- Scalable data movement with shared resources requires us to manage the transfers
- Scheduling properly can greatly reduce the impact of I/O

Data Service Approach



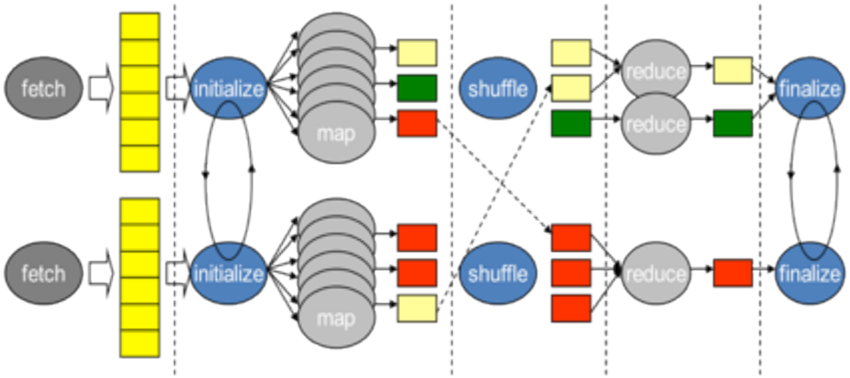
- Output costs can be reduced
- Total data size can be managed
- Input cost to workflow can be reduced
- Meta-operations can aid eventual analysis
- Application is decoupled from storage bottlenecks

Runtime Overhead comparison for all evaluated scheduling mechanism 16 Stagers



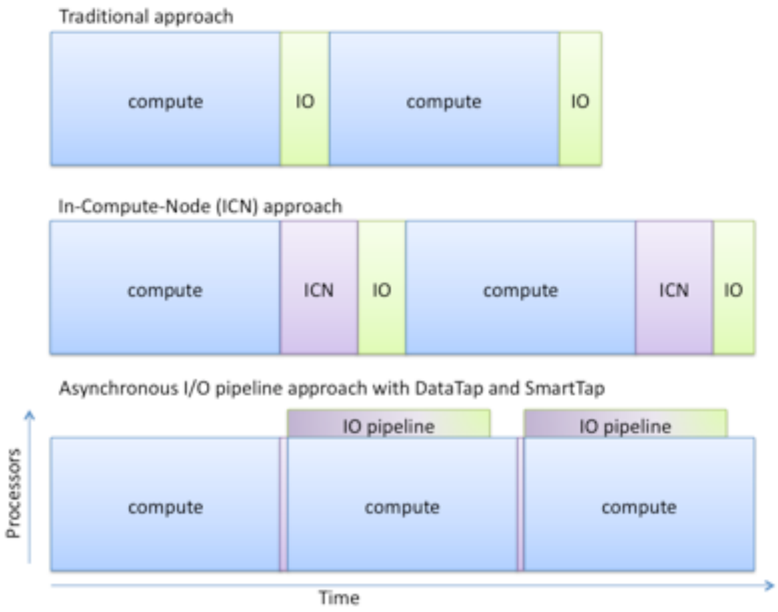
Creation of I/O pipelines to reduce file activity

Streaming Processing in Staging Area

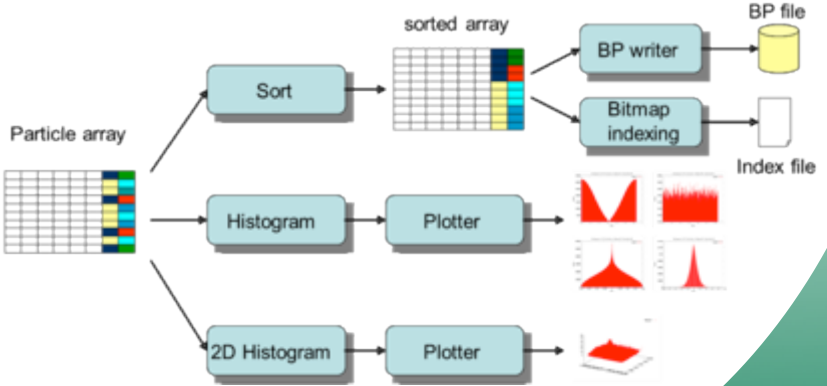


Differences with MapReduce:

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

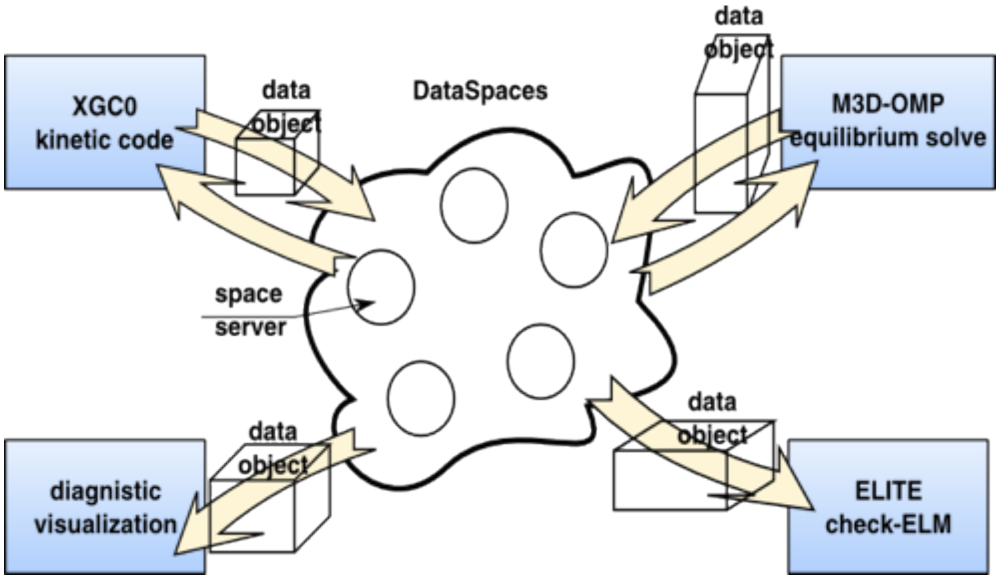


Example of an I/O pipeline



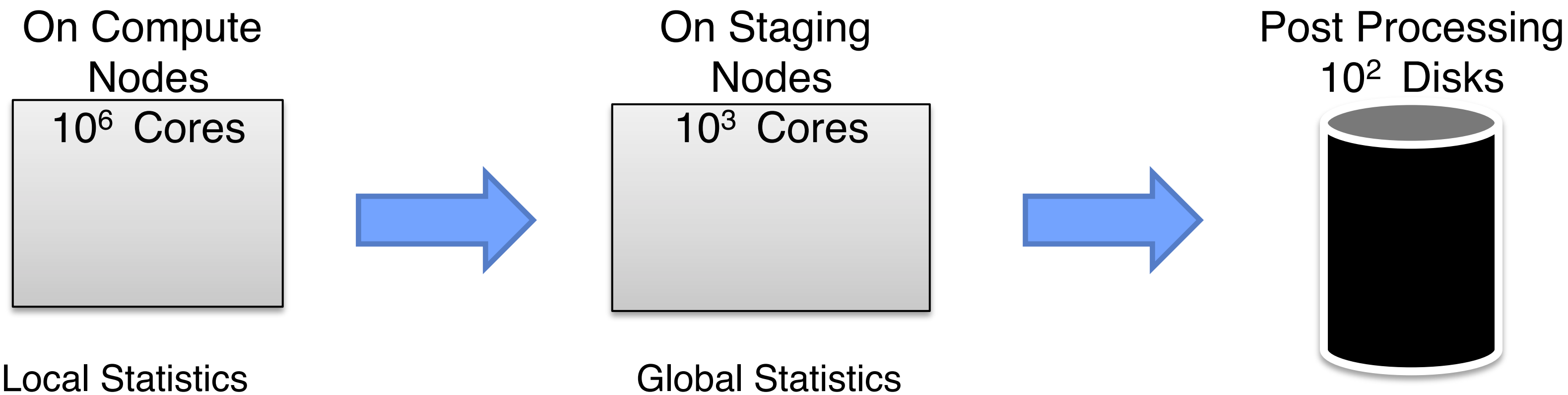
ADIOS with DataSpaces for in-memory loose code coupling

- 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

Dividing up the pipeline



On Compute Nodes

10^6 Cores

Local Statistics (Min, Max, Mean)
Local features
Binning of data
Sorting within a bucket
Compression

Minimize cross node communication

On Staging Nodes

10^3 Cores

Global Statistics
Global features
Ordered sorts
Indexing of data
Compression
Spatial correlation
Topology mapping

Minimize cross timestep communication

Post Processing

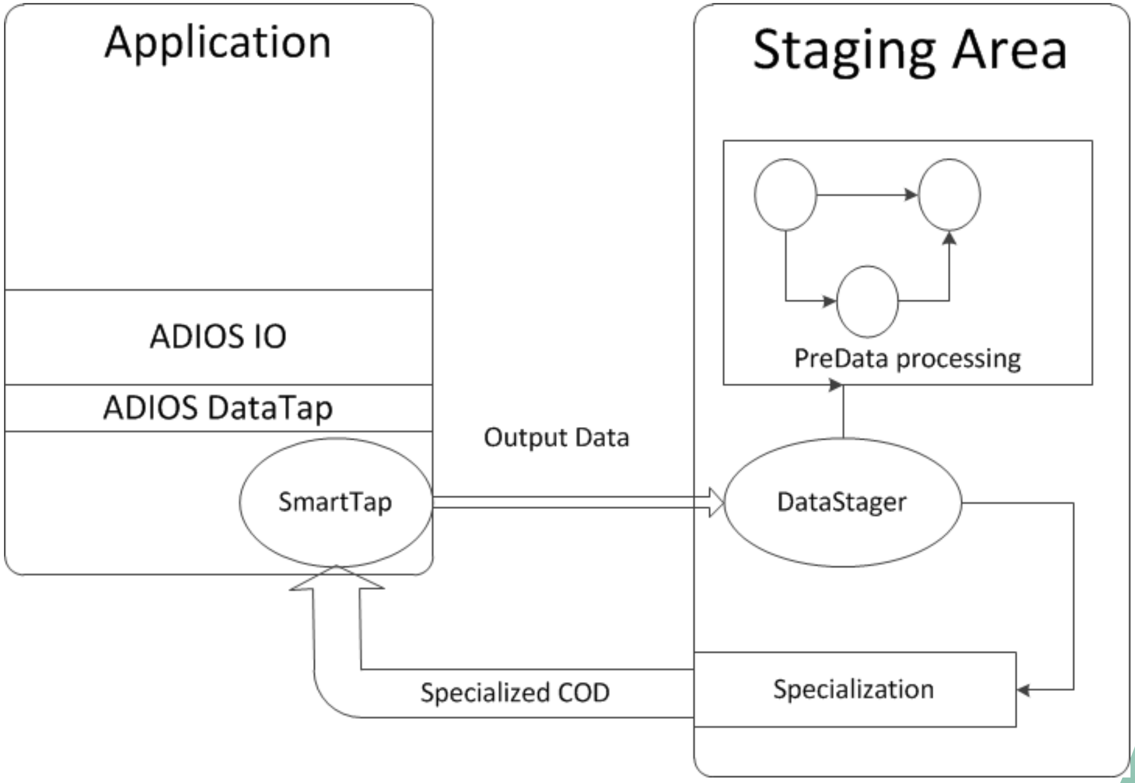
10^2 Disks

Temporal Correlations

Everything else

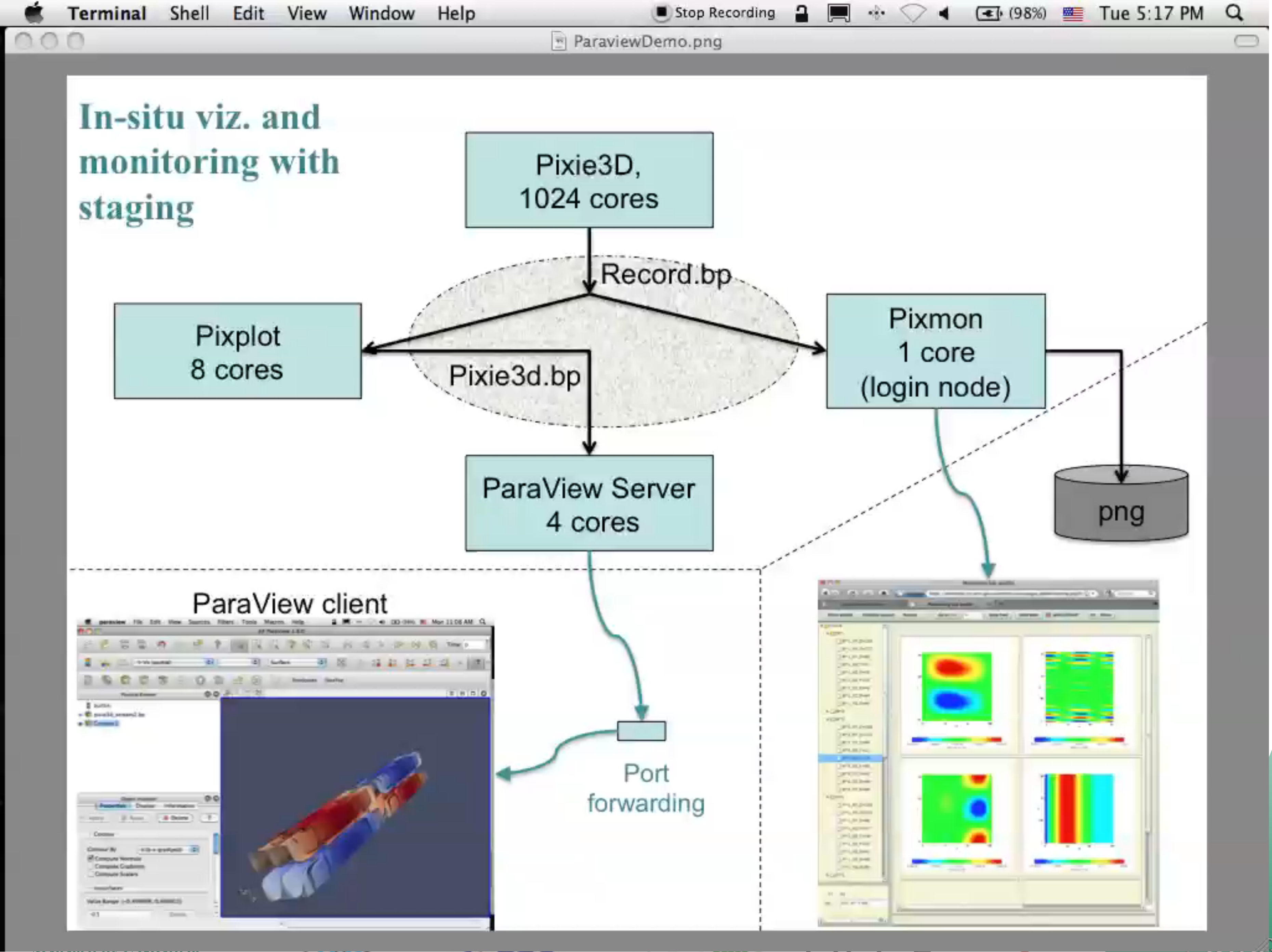
JITStager

- Runtime placement decisions
- Dynamic code generation
- Filter specialization
- Integrated with ADIOS
- Moves code to data.



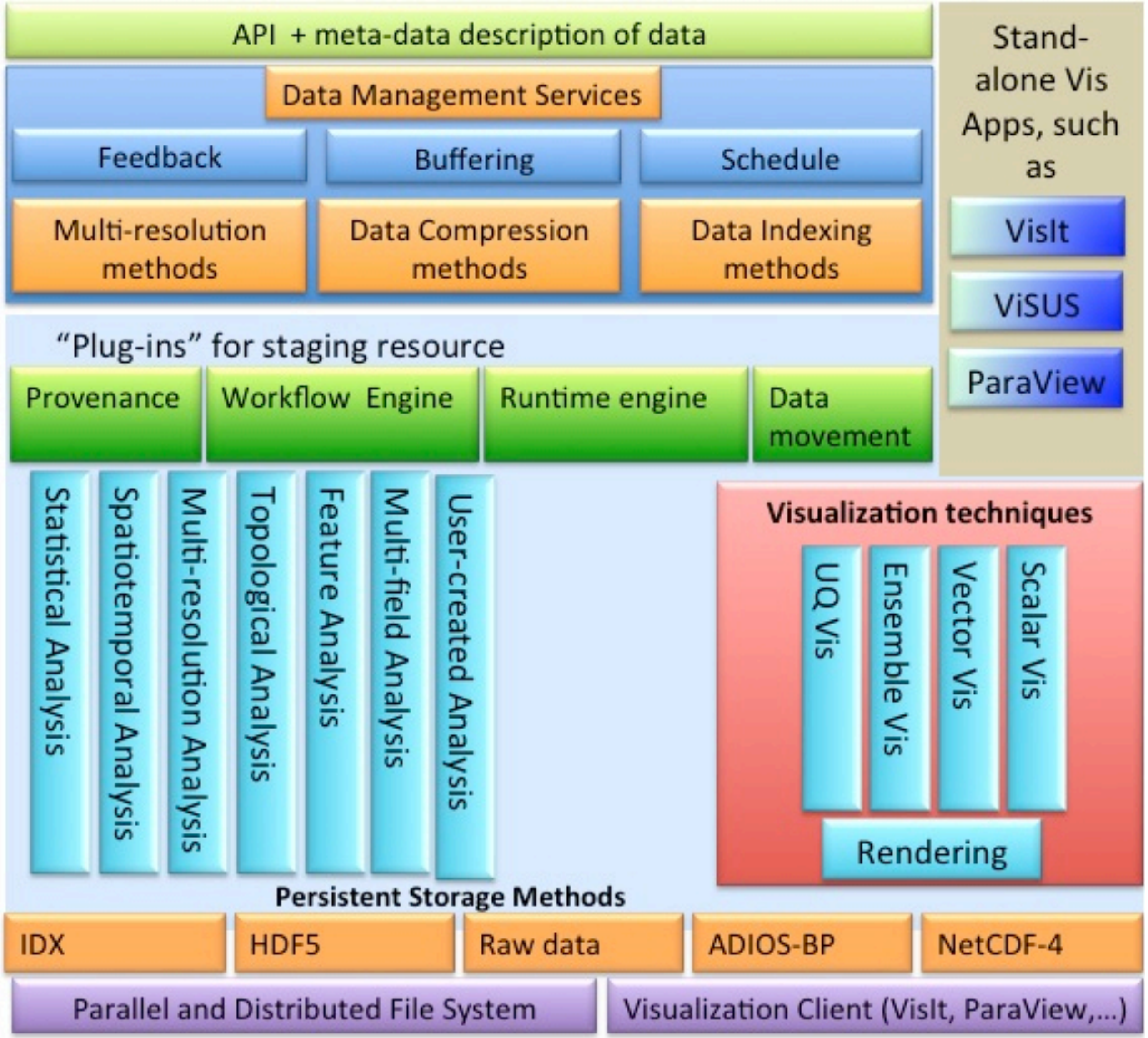
ActiveSpaces: Dynamic Code Deployment

- Provide the programming support to define custom data kernels to operate on data objects of interest
- Provide the runtime system to dynamically deploy binary code to DataSpaces, execute them on the relevant data objects in parallel, and return results
- Advantages
  - Data kernel size is typically smaller than data sizes
    - Processing often reduces data size
  - Data processing is offloaded to external resources such as the staging node
  - Faster processing time due to better data-locality in the staging area (*i.e.*, the data source)



In-situ viz. and monitoring with staging

Next generation analytics stack



API + meta-data description of data

Data Management Services

Feedback

Buffering

Schedule

Multi-resolution methods

Data Compression methods

Data Indexing methods

Stand-alone Vis Apps, such as

VisIt

ViSUS

ParaView

"Plug-ins" for staging resource

Provenance

Workflow Engine

Runtime engine

Data movement

Statistical Analysis

Spatiotemporal Analysis

Multi-resolution Analysis

Topological Analysis

Feature Analysis

Multi-field Analysis

User-created Analysis

Visualization techniques

UQ Vis

Ensemble Vis

Vector Vis

Scalar Vis

Rendering

Persistent Storage Methods

IDX

HDF5

Raw data

ADIOS-BP

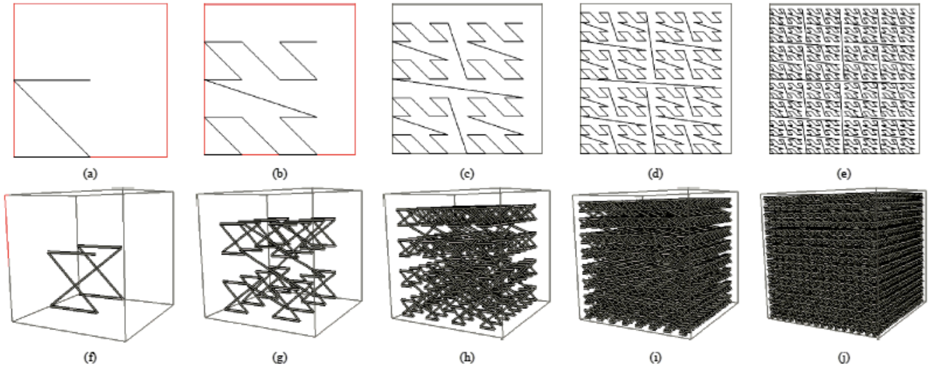
NetCDF-4

Parallel and Distributed File System

Visualization Client (VisIt, ParaView,...)

More up-coming features.

- Multi-resolution output/analysis. Pascucci, Frank, "Global Static Indexing for Real-time exploration of Very large regular Grids"
  - Idea is to re-order the data, using a Z-SFC, and to provide algorithms to "progressively analyze the output"
- Renew the focus on topological methods
  - Extract the feature in the data to reduce the amount of data touching the file system.
- ADIOS 1.4 will incorporate I/O compression + multi-resolution output formats (in BP).
- Query interface coming soon.



Questions and challenges.

1. How to run complex queries for large data saved from scientific data.
2. How to perform complex analysis with “plug-in” services created from users, with best numerical algorithms by analysis/visualization experts.
3. How do we minimize the I/O impact when reading and writing data, and allow file format to work on multitude of file systems.
4. Ensure a type of QoS while working with data.

---

1. Data Mining Techniques for performing fast queries.
2. Certificates, along with virtualizing “analysis/ visualization” clusters, allow scientist to move and reserve VM to move to data to work with “large” complex data and multiple locations.
3. Many approaches to handle this challenge. Our approach is with the ADIOS-BP file format.
4. Always a challenge with large data running on batch systems. We need “predictable” performance.