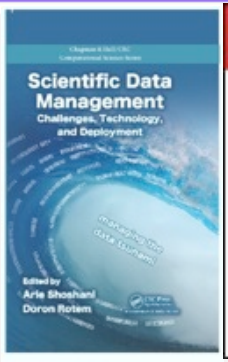
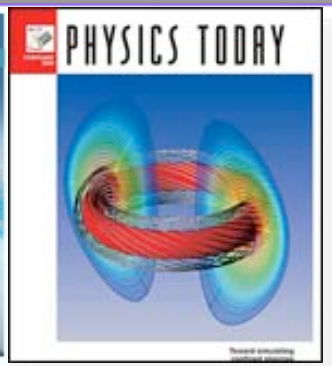
In situ data processing for extreme-scale computing





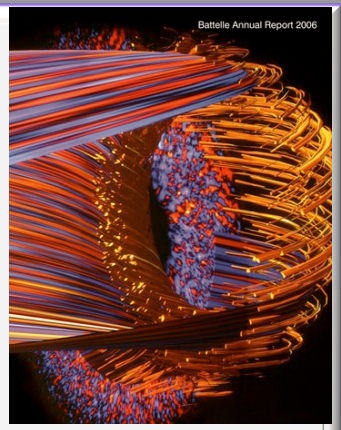
ICNSP 2011

9/8/2011

Scott A. Klasky

klasky@ornl.gov

H. Abbasi, Q. Liu, J. Logan, N. Podhorszki, R. Tchoua (ORNL)
K. Schwan, M. Wolf (Georgia Tech), M. Parashar (Rutgers)



Special Thanks to:

. Ahern[1], I.Altintas[9], L. Chacon[1], CS Chang[10], J. Chen[5], J. Cummings[13], C. Docan[6], G. Eisenhauer[4], S. Ethier[10], R. Grout[7], R. Kendall[1], J. Kim[3], T. Kordenbrock[5], Z. Lin[10], J. Lofstead[5], X. Ma[15], K. Moreland[5], R. Oldfield[5], J. Saltz, N. Samatova[15], A. Shoshani[3] , R. Tchoua[1], Y. Tian[14], M. Vouk[15], K. Wu[3], W. Yu[14], F. Zhang[6], F. Zheng[4]

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

Outline

Five steps for success in Exascale Scientific Data Management

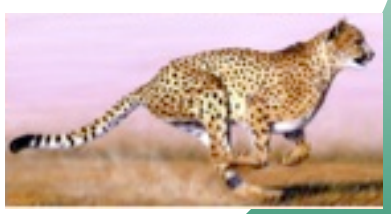
- Step 1: Identification of the problem: from CS, technology, and physics POV
- Step 2: Investigate related research
- Step 3: Solve the problem
- Step 4: Innovate
- Step 5: Impact and Future

Work supported under:

ASCR : CPES, SDM, Runtime Staging, SAP, OLCF, Co-design
FES : GPSC, GSEP
NSF : EAGER, RDAV
NASA : ROSES

We want our system to be so easy, even a chimp can use it

Sustainable
Fast
Scalable
Portable

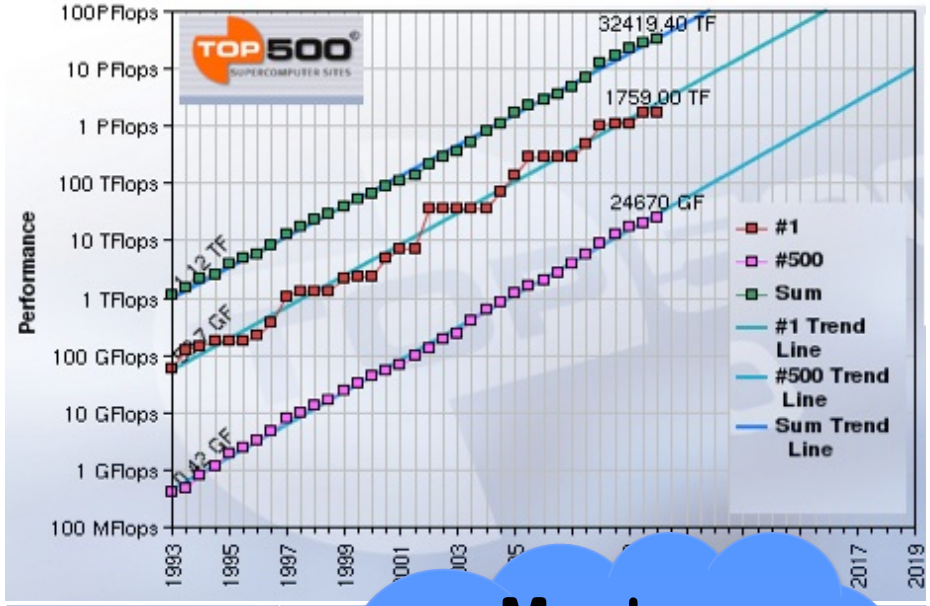


Step 1

Identify the problem

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!
- ASCI purple (49 TB/140 GB/s) – JaguarPF (300 TB/200 GB/s)



Systems	20			
System Peak Flops/s				
System Memory				
Node Performance				
Node Memory BW				/s
Node Concurrency				(1000)
Interconnect BW	1.5 GB			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

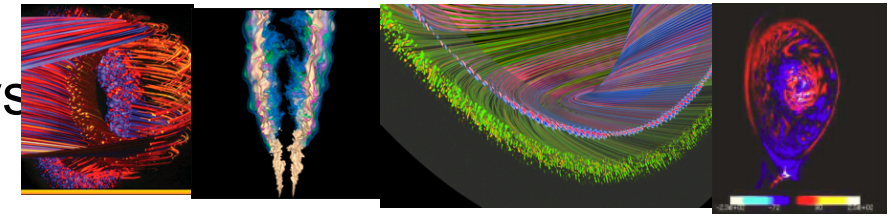
From J. Dongarra, "Impact of Architecture and Technology for Extreme Scale on Software and Algorithm Design," Cross-cutting Technologies for Computing at the Exascale, February 2-5, 2010.

Next generation I/O and file system challenges

- At the architecture or node level
  - Use increasingly deep memory hierarchies coupled with new memory properties
- At the system level
  - Cope with I/O rates and volumes that stress the interconnect and can severely limit application performance
  - Can consume unsustainable levels of power
- At the exascale
  - Immense aggregate I/O needs with potentially uneven loads placed on underlying resource
  - Can result in data hotspots, interconnect congestion and similar issues

Applications are the main drivers

- Our team works directly with
  - Fusion: XGC1, GTC, GTS, GTC-P, Pixie3D, M3D, GEM
  - Combustion: S3D
  - Relativity: PAMR
  - Nuclear Physics: Nuclear Phys
  - Astrophysics: Chimera, Flash
  - Many more application teams



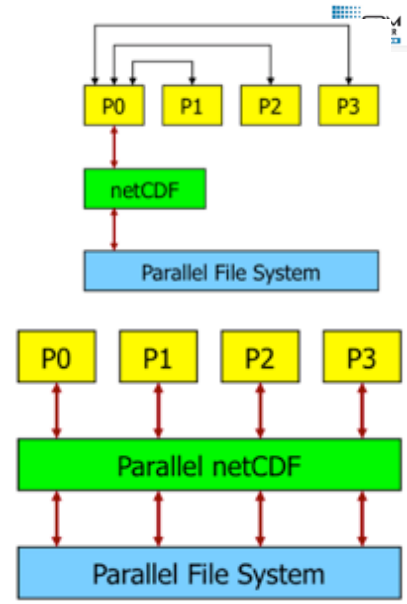
- Requirements (in order)
  - Simplicity in using I/O
  - Fast I/O with low overhead
  - Self Describing file formats
  - QoS
- Extras, if possible
  - Ability to visualize data from simulation
  - Couple multiple codes
  - Perform in situ workflows

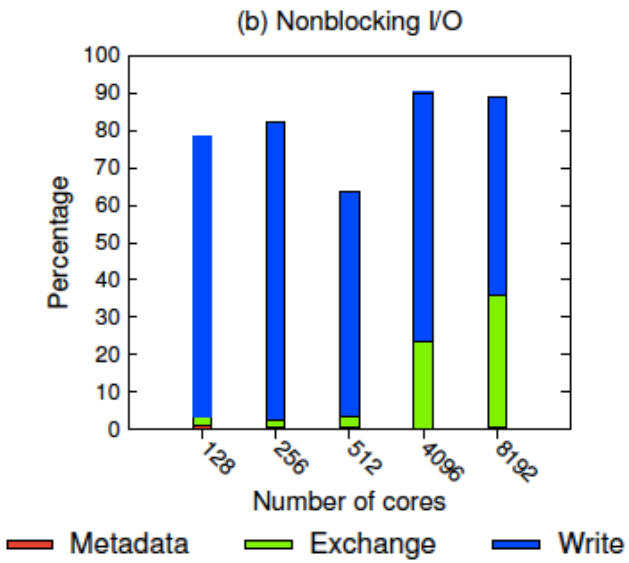
Step 2

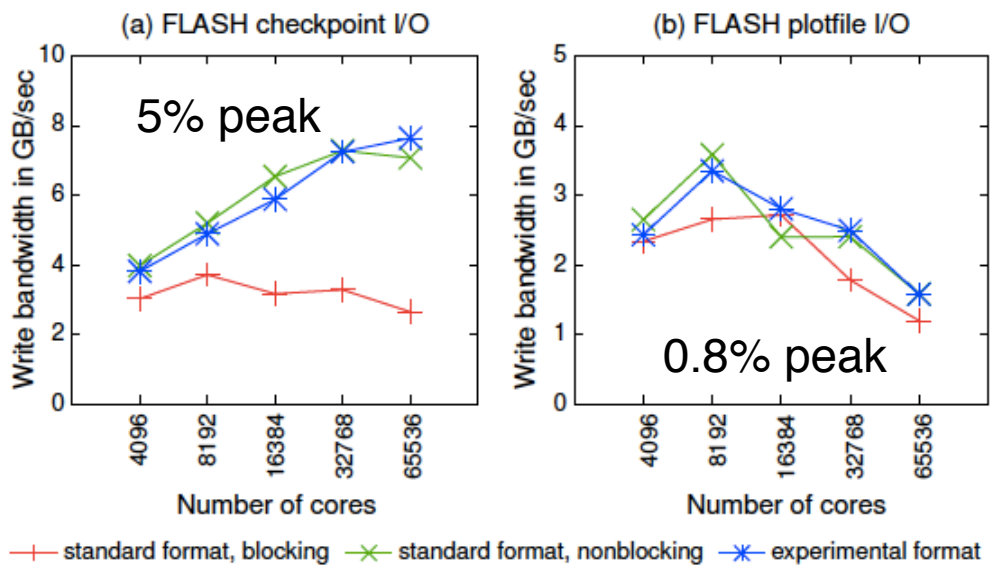
Investigate related research

Parallel netCDF

- New file format to allow for large array support
- New optimizations for non-blocking calls
- Idea is to allow netcdf to work in parallel and for large files and large arrays
- File format is not scalable, not sustainable



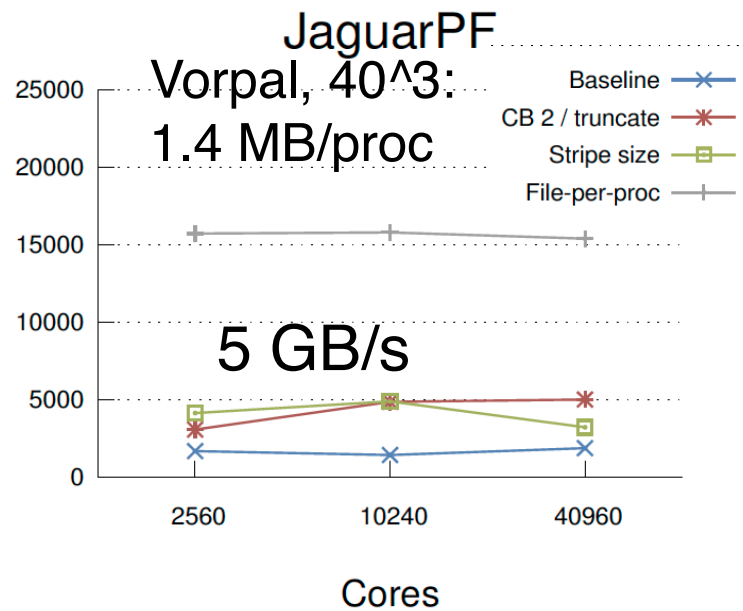




2011 paper: A Case Study for Scientic I/O: Improving the FLASH Astrophysics Code http://www.mcs.anl.gov/uploads/cels/papers/P1819.pdf

Parallel netCDF-4/ HDF5

- 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

Step 3

Solve the problem

The "early days": 2001, Reduce I/O overhead for 1 TB data from the GTC code and "real-time" monitoring

- 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 applicatio driven approach
  - Buffer Data, and combine all I/O reques 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**

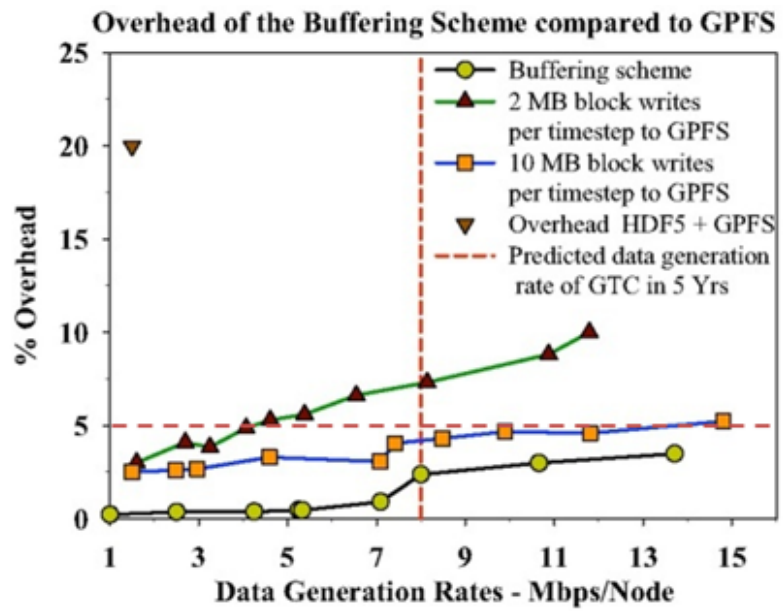
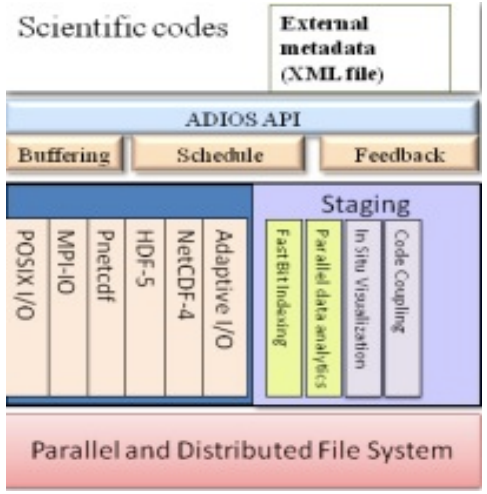


Figure: 8. Overhead with Buffering Scheme compared to GPFS (I/O).

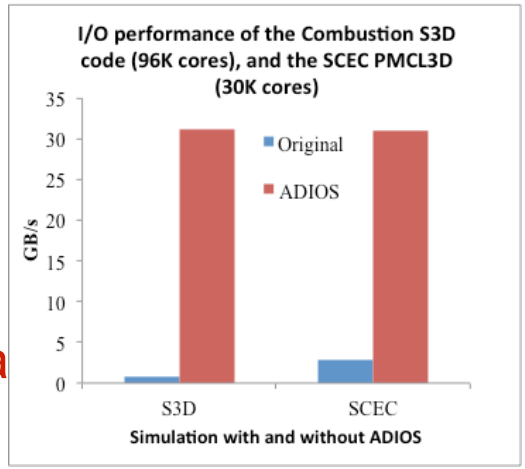
Adaptable I/O System

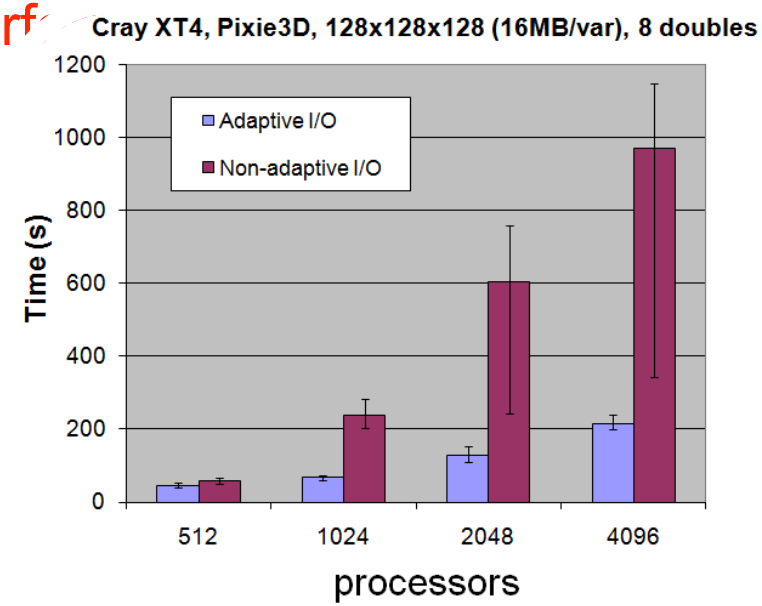
- Provides portable, fast, scalable, easy-to-use, metadata rich output with a simple API
- Change I/O method by changing XML input file
- Layered software architecture:
  - Allows plug-ins for different I/O implementations
  - Abstracts the API from the method used for I/O
- Open source:
  - http://www.nccs.gov/user-support/center-projects/adios/
- High Writing Performance
  - XGC1 code → 40 GB/s
  - GTC code → 40 GB/s, GTS code: 35 GB/s
  - S3D: 32 GB/s with 96K cores, 1.9MB/core: 0.6% I/O overhead with ADIOS, SCEC code 30 GB/s

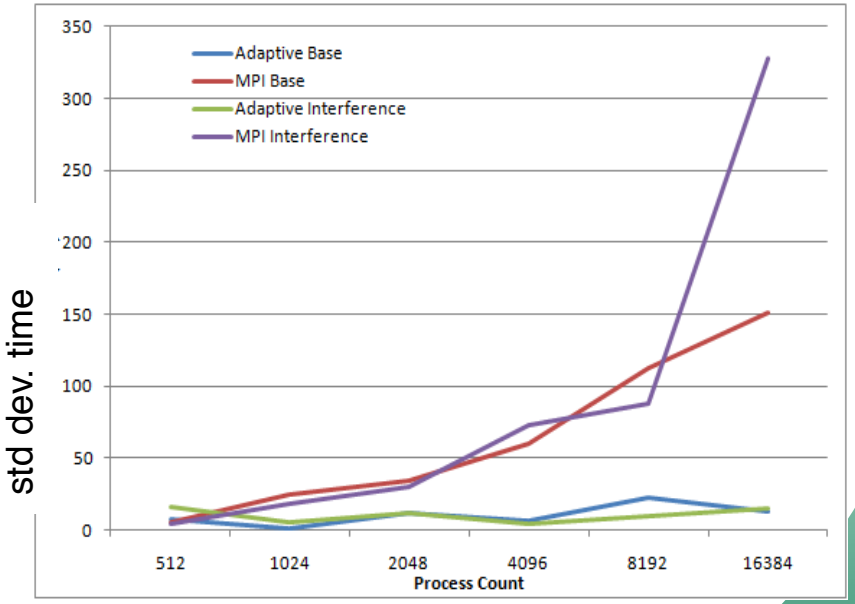


Performance Optimizations for Write

- New technologies are usually constrained by the lack of usability in extracting performance
- Next generation I/O frameworks must address this concern
  - Partitioning the task of optimizations from the actua description of the I/O
- Decrease high I/O variability, and increase "average" I/O perf

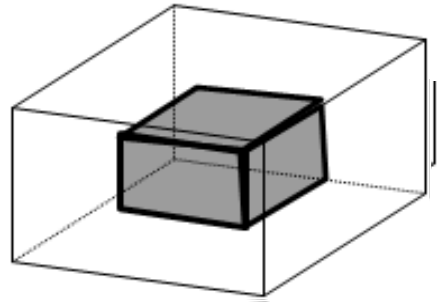


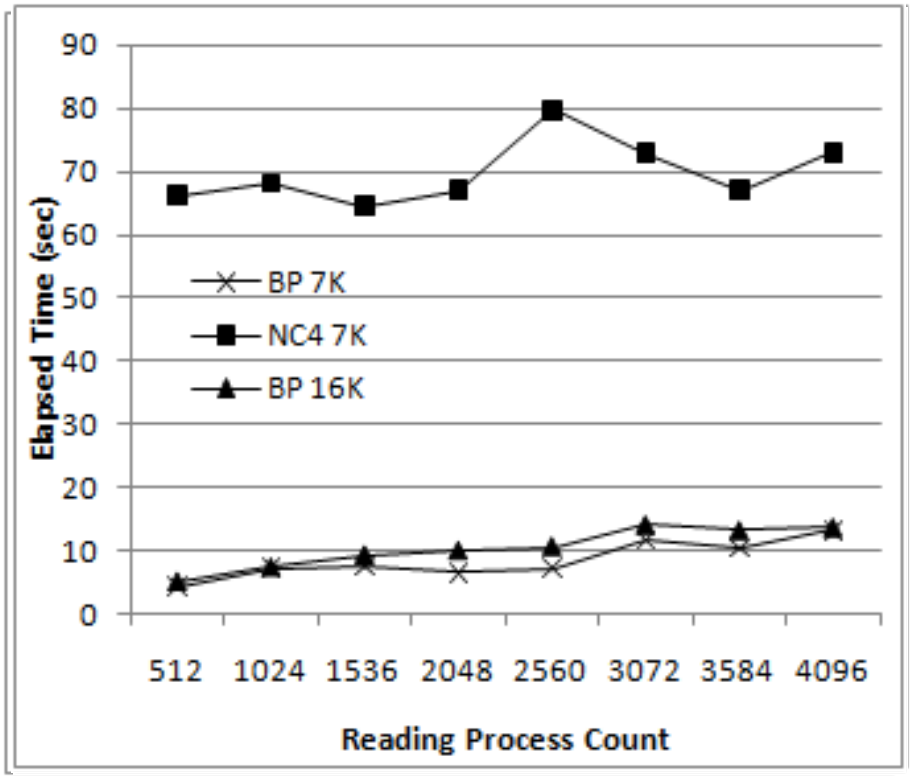




Understand common read access patterns

- Sub Volume

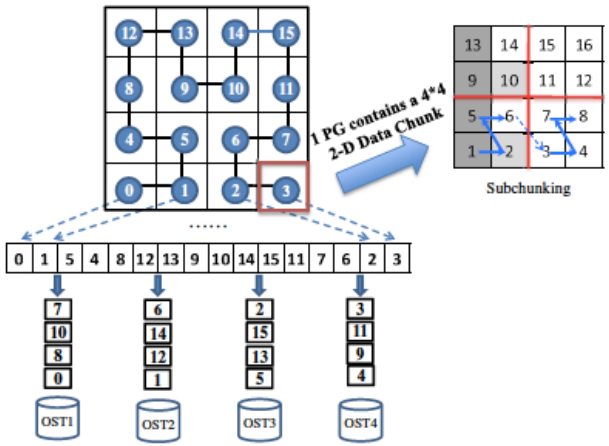




Conclusion: File formats that chunk the data routinely achieved better read performance than logically contiguous file formats

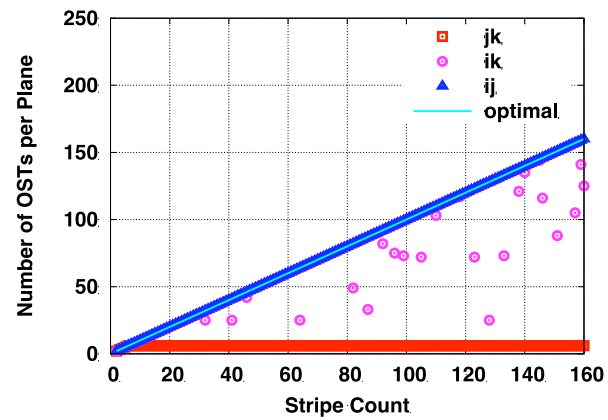
Understand why: (Examine reading a 2D plane from a 3D dataset)

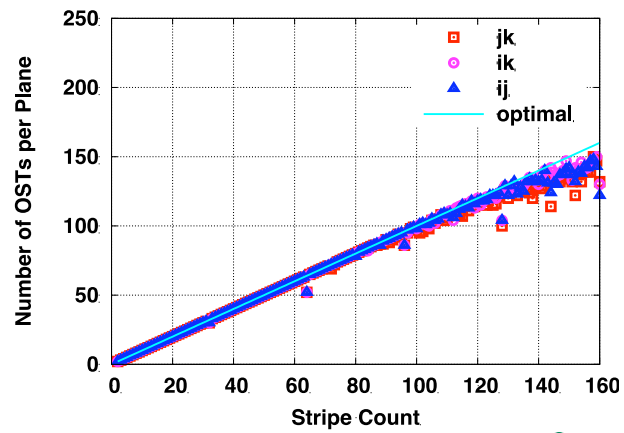
- Use Hilbert curve to place chunks on lustre file system with an Elastic Data Organization



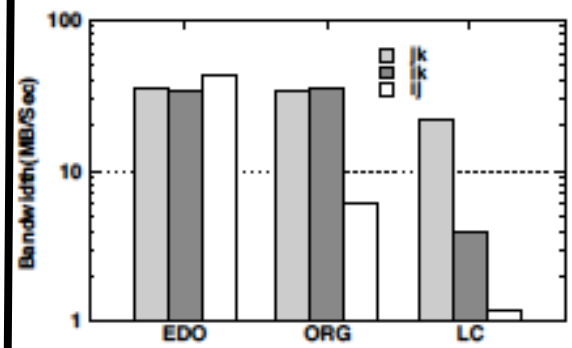
(b) SFC-based Placement with EDO

Theoretical concurrency

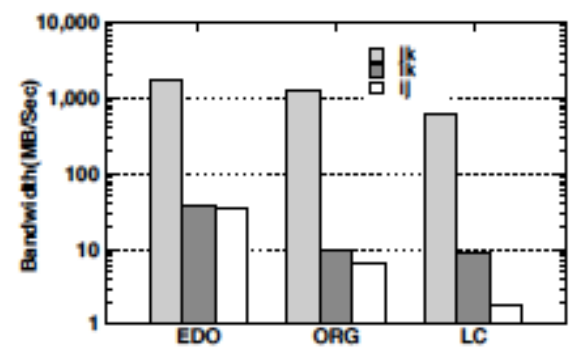




Observed Performance



(a) Case S (stripe=128)



(b) Case X (stripe=128)

Step 4

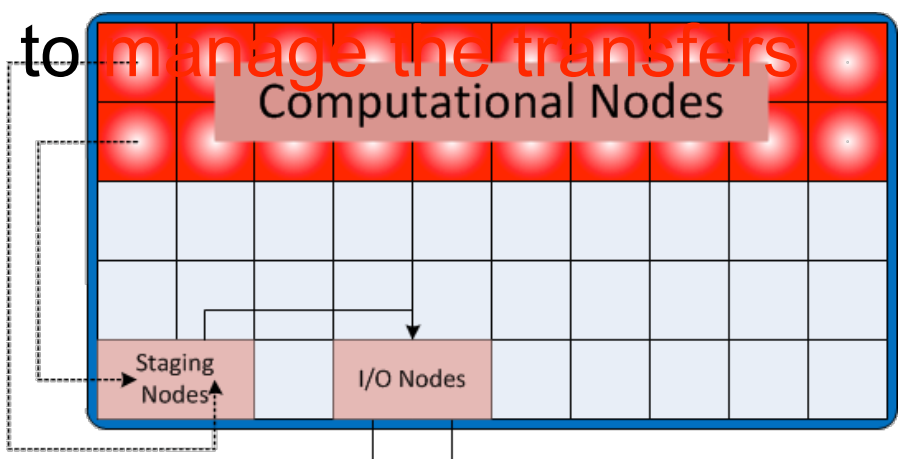
Innovate for *In situ* data processing for extreme-scale computing

SOA philosophy

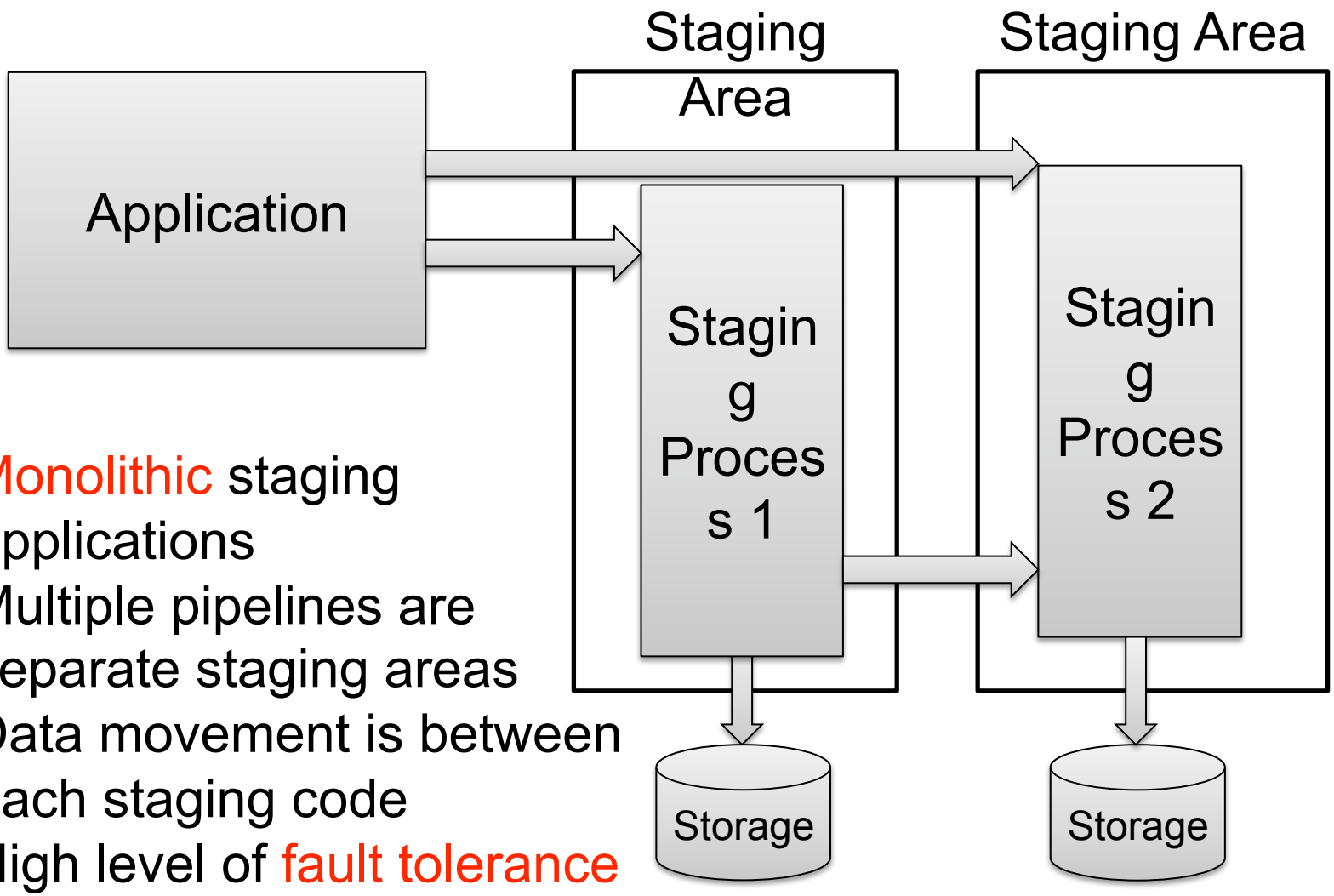
- The overarching design philosophy of our framework is based on the Service-Oriented Architecture
  - Used to deal with system/application complexity, rapidly changing requirements, evolving target platforms, and diverse teams
- Applications constructed by assembling services based on a universal view of their functionality using a well-defined API
- Service implementations can be changed easily
- Integrated simulation can be assembled using these services
- Manage complexity while maintaining performance/scalability
  - Complexity from the problem (complex physics)
  - Complexity from the codes and how they are
  - Complexity of underlying disruptive infrastructure
  - Complexity from coordination across codes and research teams
- Inspiration from industry/Yahoo data challenges: Google manages 24 PB/day.

Designing the system

- Why staging?
  - 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: RDMA
- Scalable data movement with shared resources requires us to manage the transfers



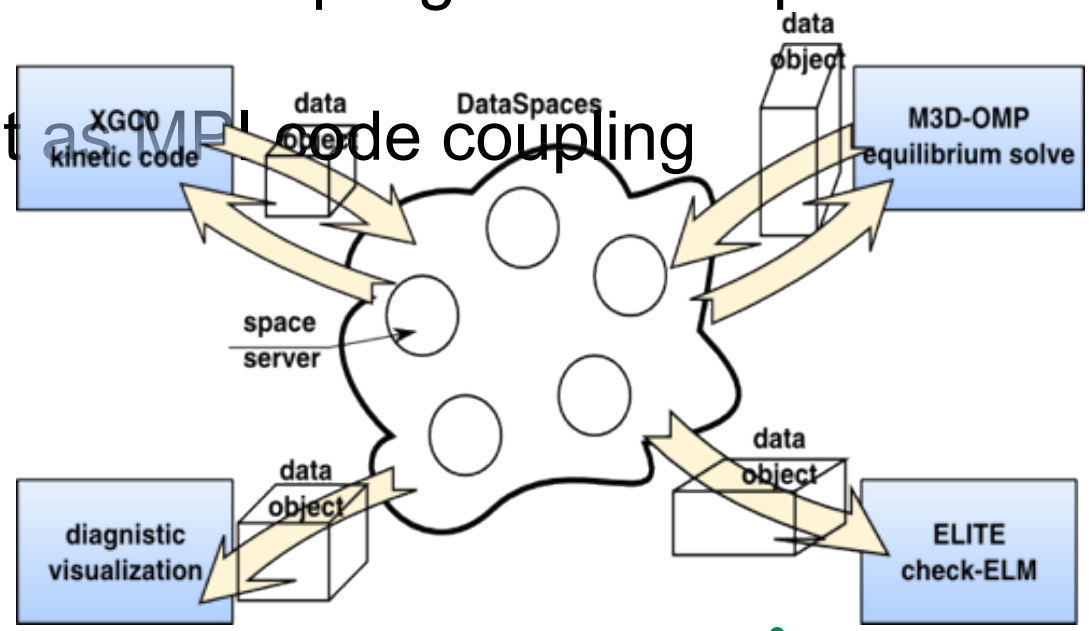
Staging Revolution: Phase 1



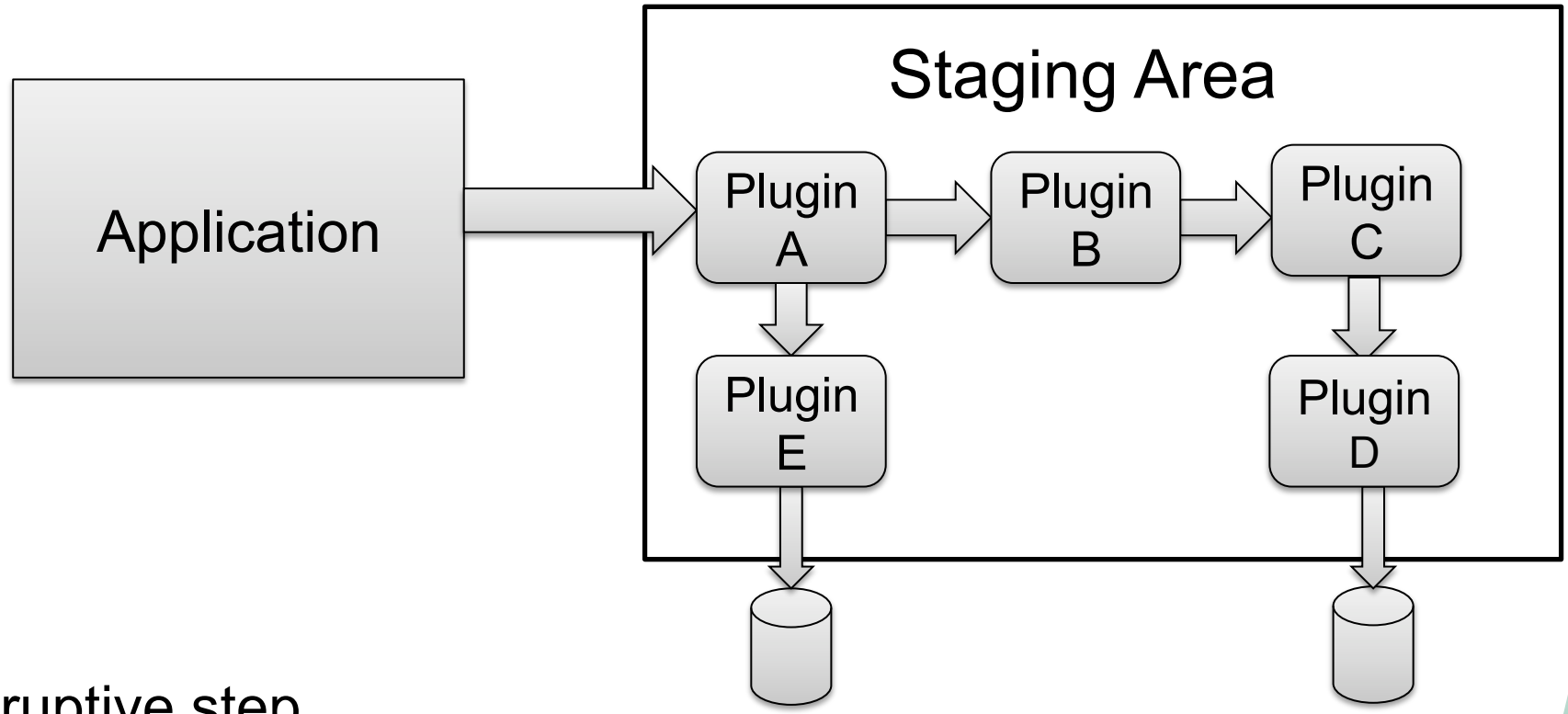
- Monolithic staging applications
- Multiple pipelines are separate staging areas
- Data movement is between each staging code
- High level of fault tolerance

Approach allows for in-memory code coupling

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

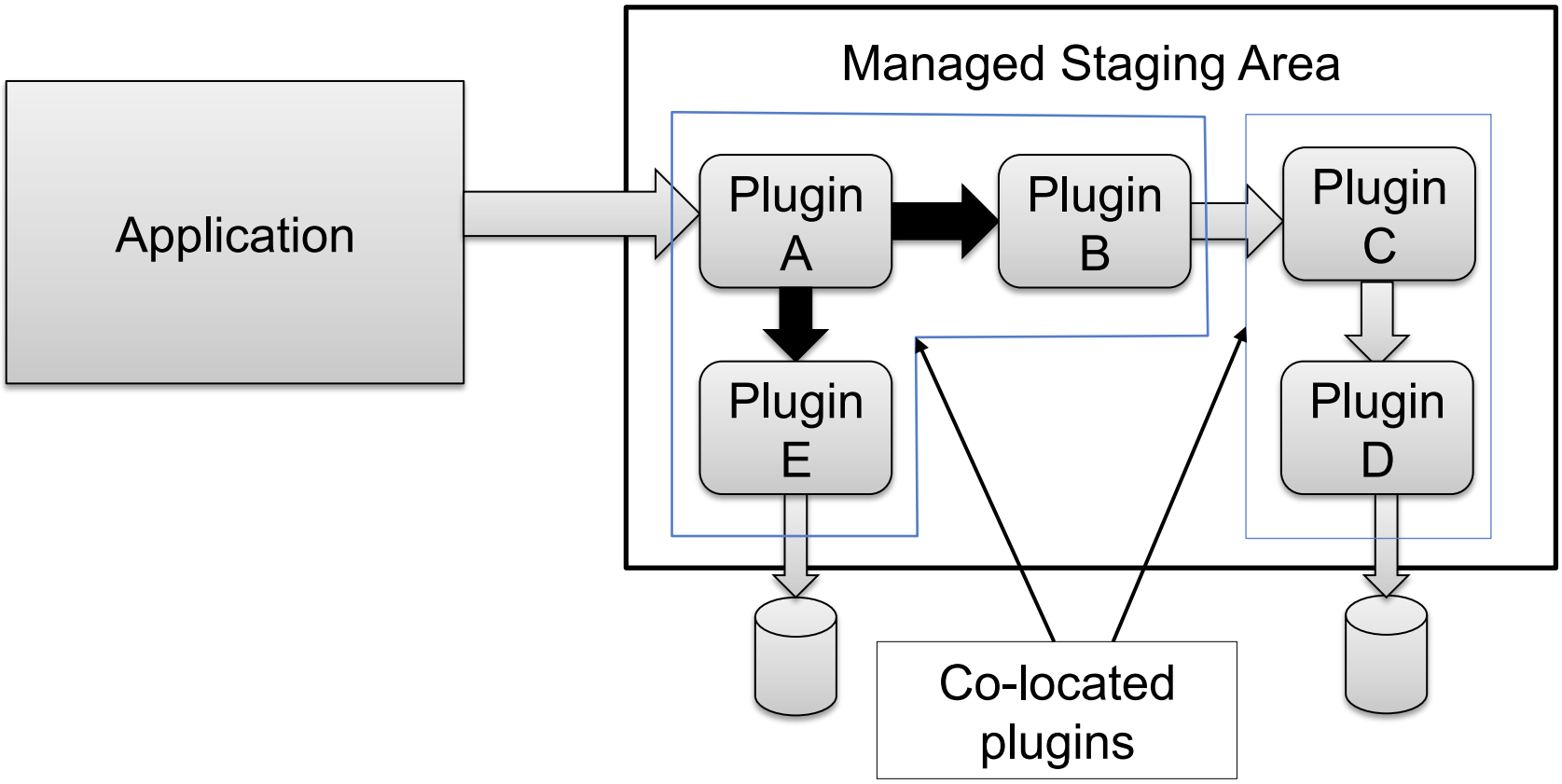


Phase 2: Introducing Plugins



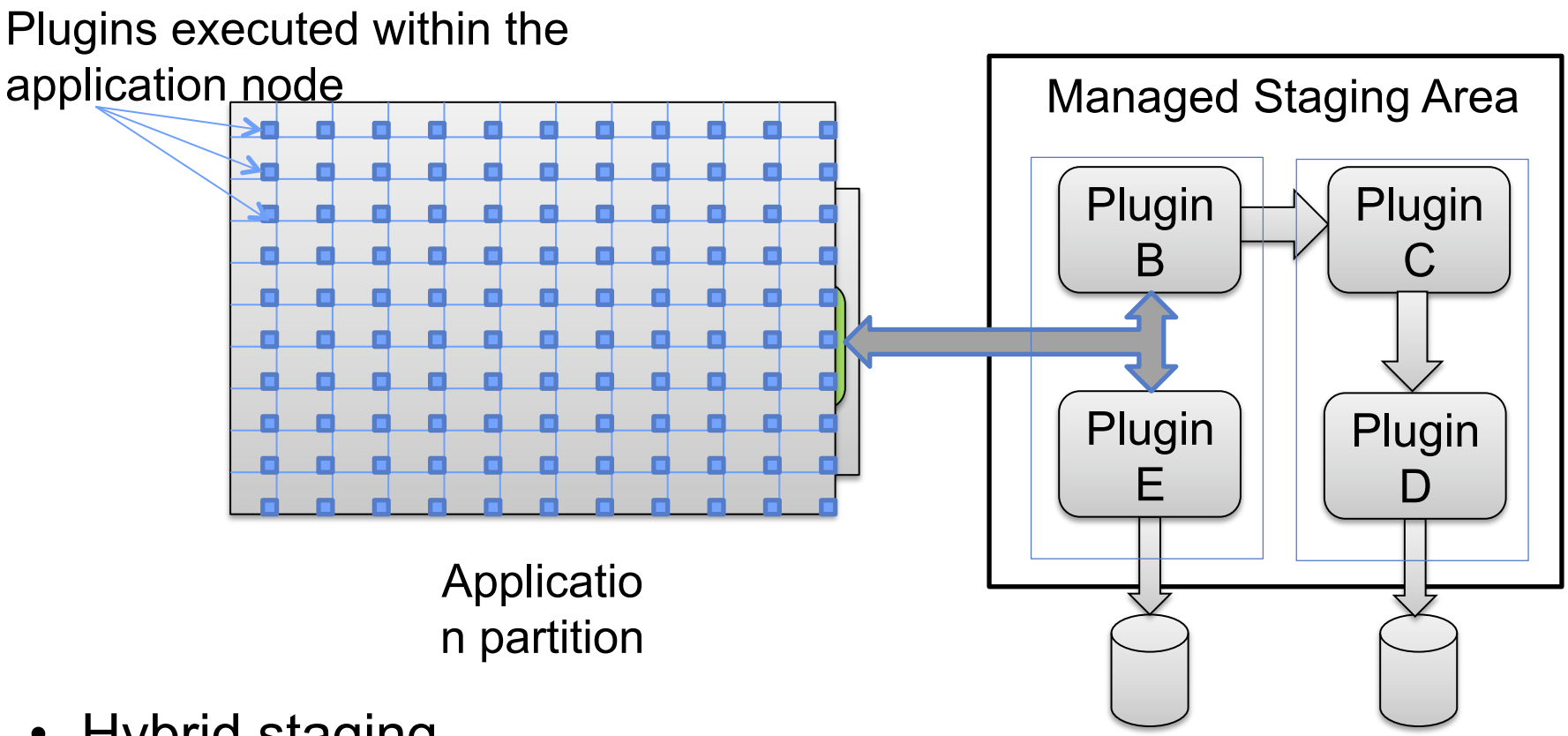
- Disruptive step
- Staging codes broken down into multiple plugins
- Staging area is launched as a framework to launch plugins
- Data movement occurs between all plugins
- Scheduler decides resource allocations

Phase 3: Managing the staging area



- Resources are co-allocated to plugins
- Plugins are executed consecutively, co-located plugins for no data movement
- Scheduler attempts to minimize data movement

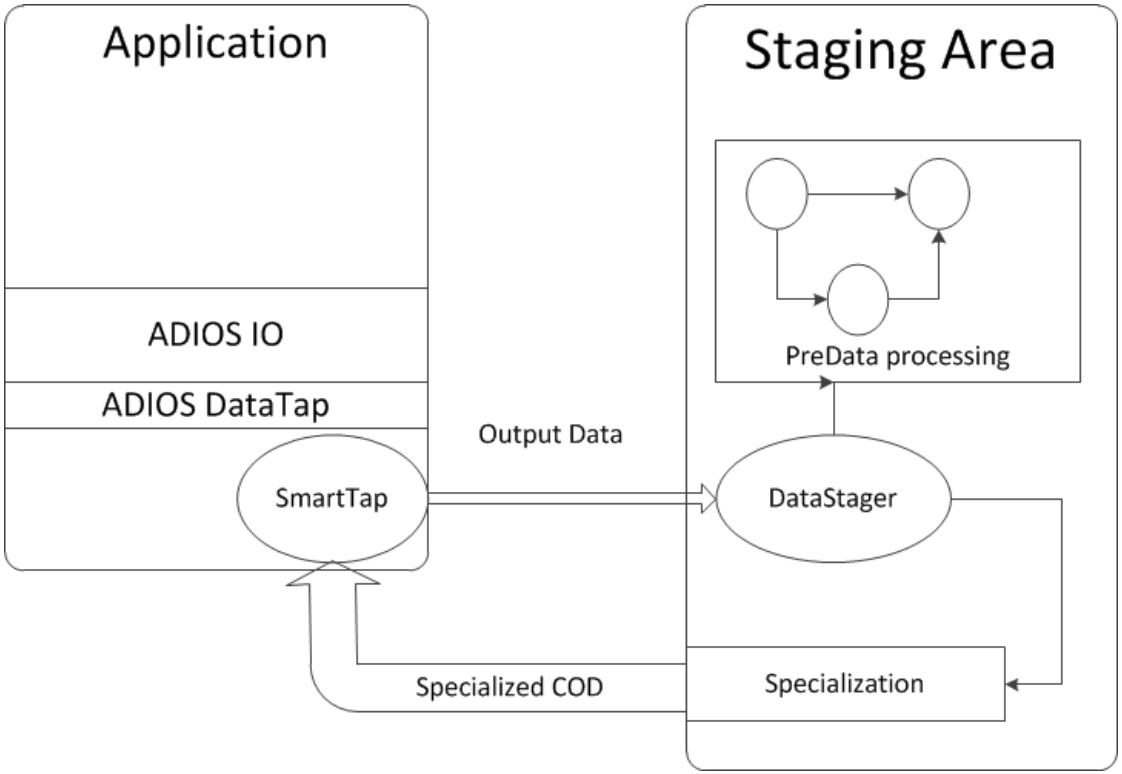
Phase 4: Hybrid Staging



- Hybrid staging
- Plugins may be executed in the application space
- Scheduler decides which plugin is migrated upstream to the application
- Maintain resiliency of the staging architecture

Move work to data (I/II)

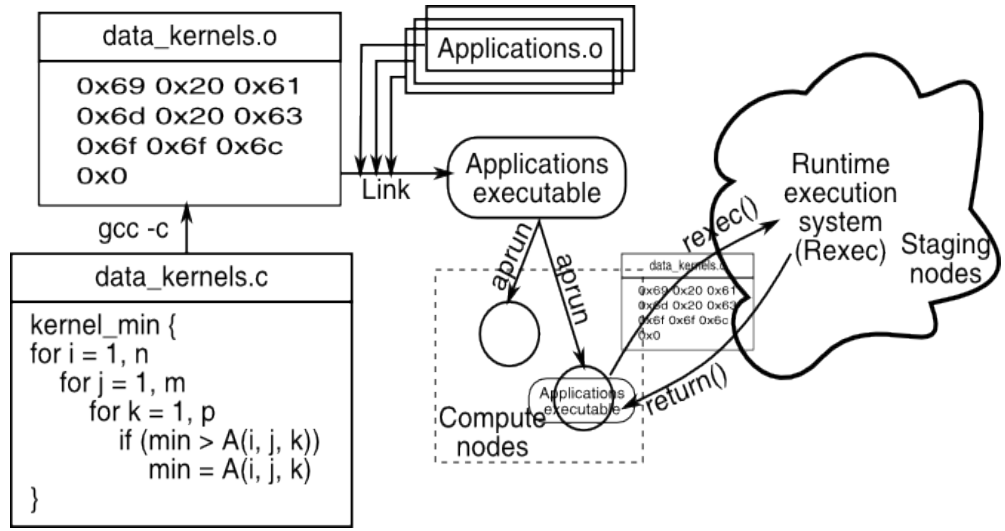
- Runtime placement decisions
- Dynamic code generation
- Filter specialization

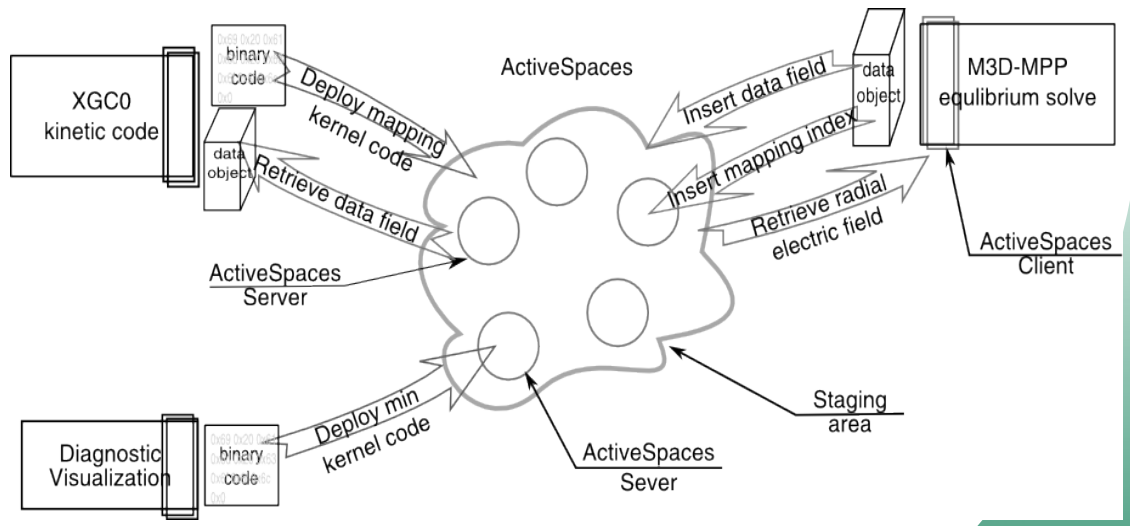


- Staging functions are implemented in C-on-Demand (CoD)
  - CoD can be transparently moved from staging nodes to application nodes or vice versa

Move work to data (II/II)

- *ActiveSpaces* – Dynamic ***binary code*** deployment and execution
- Programming support for defining custom data kernels using native programming language
  - Operates only on data of interest
- Runtime system for binary code transfer and execution in the staging area

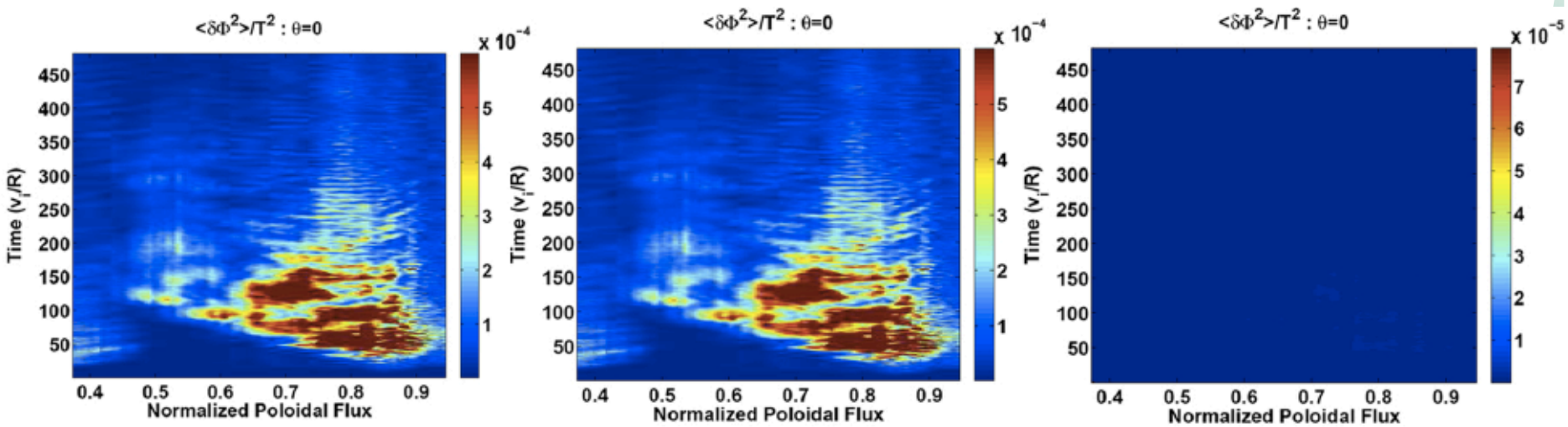




Data Management (Data Reduction/in situ)

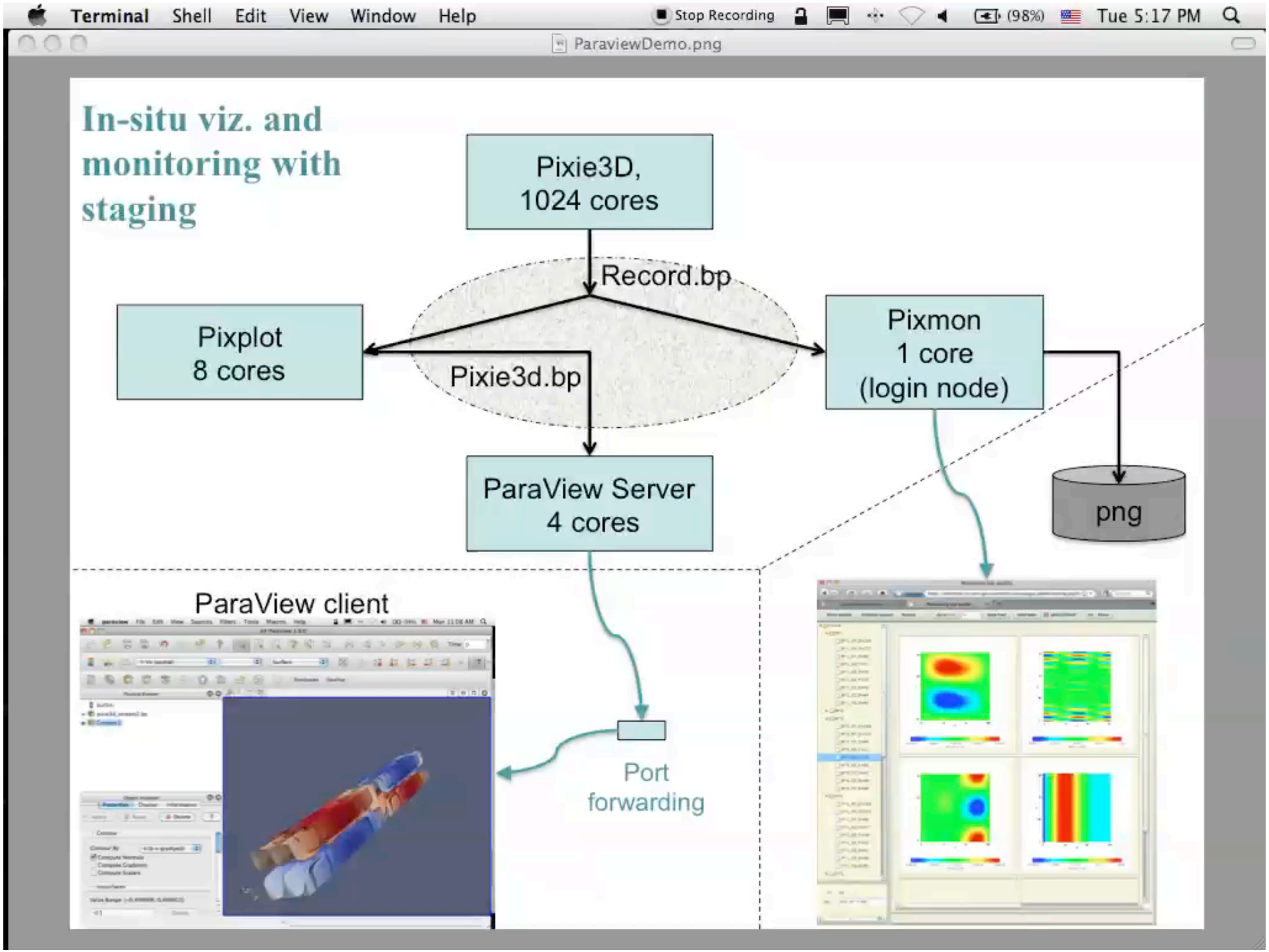
- Develop a lossy compression methodology to reduce "incompressible" spatio-temporal double precision scientific data
- Ensure high compression rates with ISABELA, while adhering to user-controlled accuracy bounds
- Provide fast and accurate approximate query processing technique directly over ISABELA-compressed scientific data
- Techniques using querying with FastBit, etc. included inside the compressed data.

ISABELA Compression Accuracy at 75% reduction in size



From research to innovation to impact principles

- Understand the important problems in the field
- Partner with the “best” researchers in the field
- Form teams based on each members ability to deliver
- Avoid overlap as much as possible
- Understand the field, and the GAPS as we move forward in terms of applications involvement, technology changes, and software issues, and grow the software
- Further develop the software so that it can be sustainable


Terminal Shell Edit View Window Help
Stop Recording
(98%)
Tue 5:17 PM
ParaviewDemo.png
In-situ viz. and monitoring with staging
Pixie3D, 1024 cores
Record.bp
Pixplot 8 cores
Pixie3d.bp
Pixmon 1 core (login node)
ParaView Server 4 cores
png
ParaView client
Port forwarding

Step 5: Impact since 2008

- HPC wire highlights
  - 5 Fusion, 4 Computer Science, 1 Combustion
- SOA
  - 4 Tier-1 conferences, 1 Tier-3 conference, 1 workshop, 1 SciDAC conference, and 1 Journal article.
- Data Formats
  - 1 Tier-1 conference, 1 First-place for ACM student research competition, 1 workshop, and 1 user-group meeting.
- Data Staging
  - 5 Tier-1 conferences, 3 journal publications
- Usability of optimizations
  - 3 Tier-1 conferences, 2 workshops
- Data Management
  - 3 Tier-1 conferences, 1 Tier-2 conference, 1 SciDAC conference, 1 Journal article
- Simulation Monitoring
  - 1 Tier-1 conference, 1 Tier-2 conference, 3 Tier-3 conferences, 1 workshop.
- In situ processing
  - 1 Tier-1 conference, 1 Book, 1 Workshop, 2 SciDAC conferences
- Misc.
  - 3 SciDAC conferences, 1 IAEA, 1 journal article.

Conclusions and Future Direction

- ADIOS, developed by many institutions, is a proven DOE framework which blends together
  - Self-describing file formats for high levels of I/O performance
  - Staging methods with visualization and analysis “plugins”
  - Advanced data management plugins, for data reduction, multi-resolution data reduction, and advanced workflow scheduling
  - Multiplexing using a novel SOA approach
  - Added levels of resiliency for next generation knowledge discover in HPC
- Future work should build on these successes to integrate HPC + cloud technology
- Our goal is to deliver breakthrough science and get the job done no matter what it takes