ADIOS 1.2

EU-US Workshop on Software Technologies for Integrated Modeling in Fusion

Sweden

12/2//2010

Scott A. Klasky

klasky@ornl.gov

Collaborators / :
Lofstead, Jay; Matthew Wolf; Schwan, Karsten; Liu, Qing; Podhorszki, Norbert; Todd H Kordenbrock; Oldfield, Ron A; Abbasi, Hasan; Fang Zheng; Ciprian Docan; Fan Zhang; Divya Dinakar; Xiaosong Ma; Mladen Vouk, Samatova, Nagiza F.; Alexander Romosan; Sriram Lakshminarasimhan; Michael Warren; Bing Xie; Arie Shoshani; Micah Beck; K. John Wu, Weikuan Yu, Yuan Tian + many more.

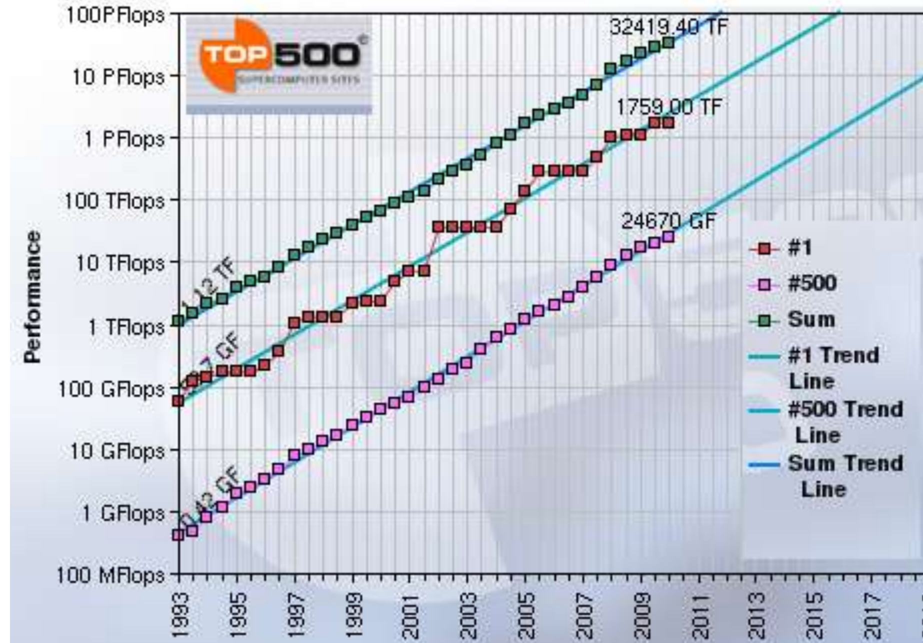
Outline

- High End Computing Trends.
- Motivation for ADIOS
- ADIOS features.
- ADIOS performance.
- ADIOS utilities.
- ADIOS demo.
- ADIOS future.

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	2009	2011	2015	2018
System Peak Flops/s	2 Peta	20 Peta	100-200 Peta	1 Exa
System Memory	0.3 PB	[illegible]	[illegible]	[illegible]
Node Performance	125 GF	[illegible]	[illegible]	[illegible]
Node Memory BW	25 [illegible]	[illegible]	[illegible]	[illegible]
Node Concurrency	12	[illegible]	[illegible]	[illegible]
Interconnect BW	1.5 GB/s	[illegible]	[illegible]	50 GB/s
System Size (Nodes)	18,700	100,000	500,000	O(Million)
Total Concurrency	225,000	3 Million	[illegible]	O(Billion)
Storage	15 PB	[illegible] 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

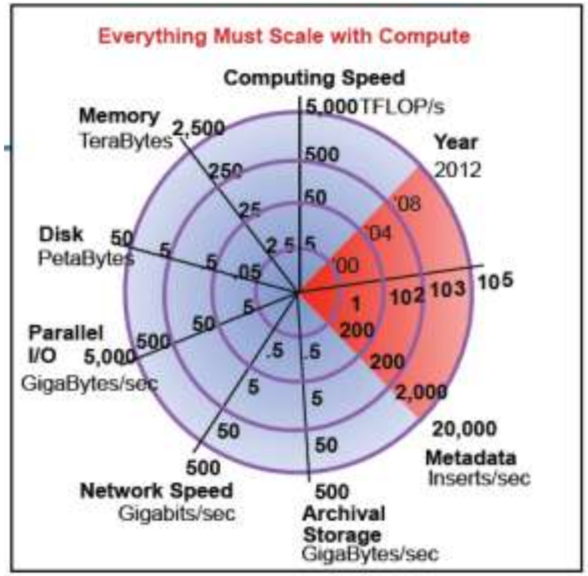
Most people get < 10 GB/s at scale

ILLUSTRATION: A. TOVEY

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.

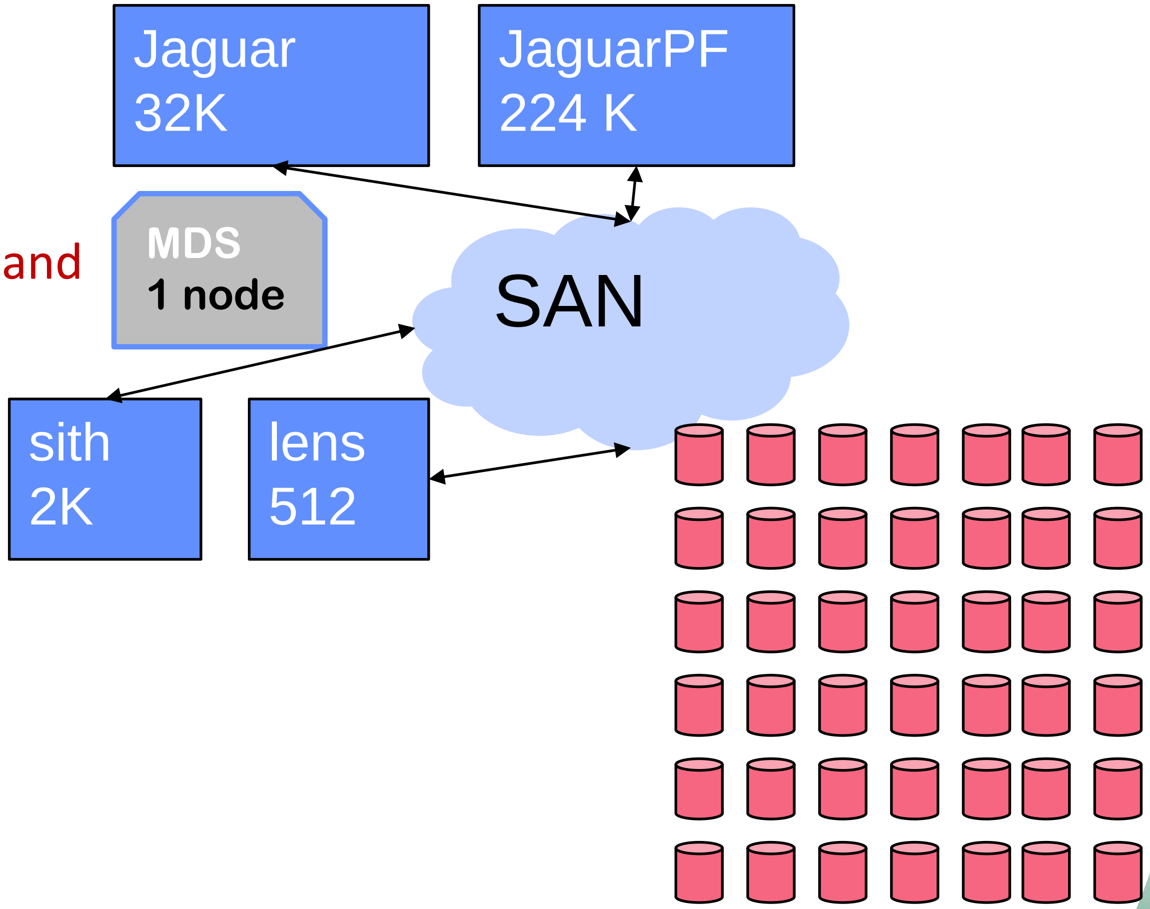


Garth Gibson 2010

- 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.

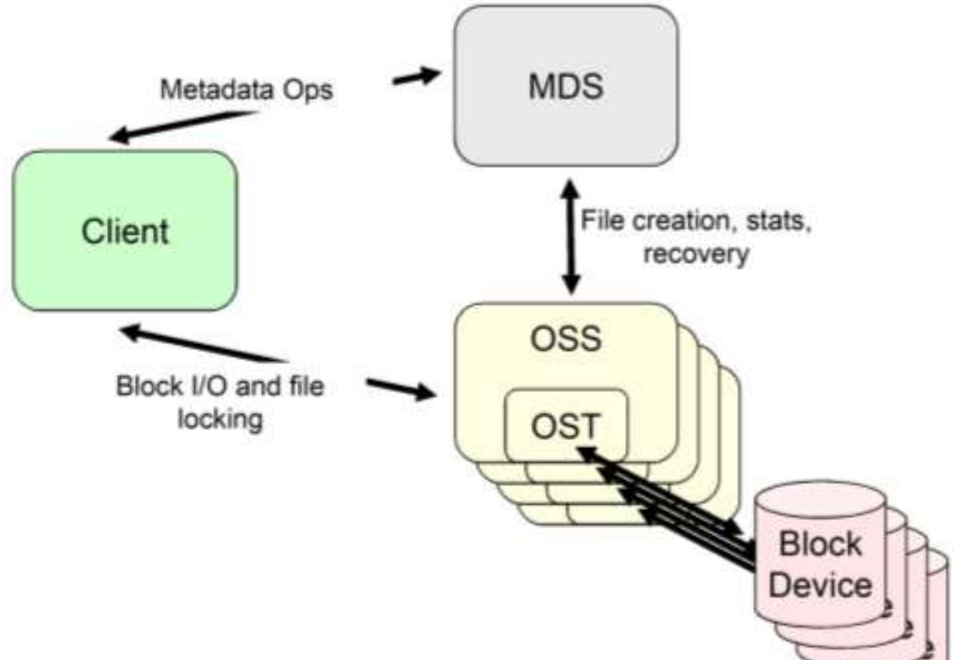
Trends in HPC Centers

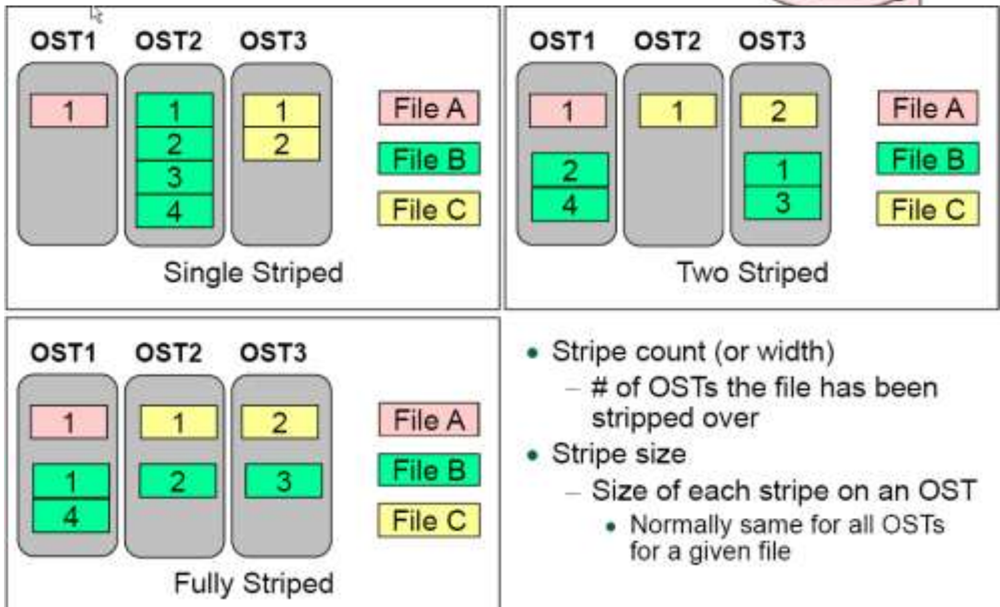
- Shared work-space
- Advantages
  - cheaper for total storage and bandwidth capacity
  - faster connection of resources to data
- Disadvantages
  - additional interference sources
  - potential single point of failure



LUSTRE

- Lustre consists of four major components
  - MetaData Server (MDS)
  - Object Storage Servers (OSSs)
  - Object Storage Targets (OSTs)
  - Clients
- MDS
- OSS
- OST
- Performance: Striping, alignment, placement
- GPFS works similar, but …





I/O Componentization: ADIOS Motivation

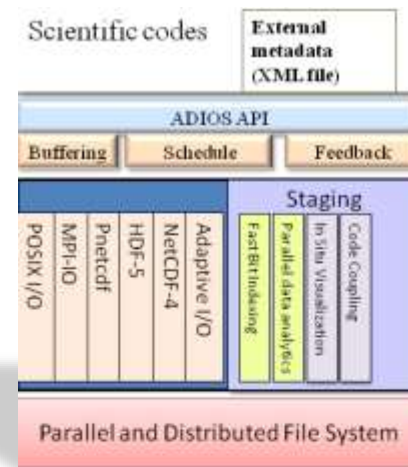
- End users should be able to select the most efficient I/O method for their code, with minimal effort in terms of code alternations/updates.
  - Systems today can have multiple file systems attached, and MPI I/O hints are difficult to use to get efficient I/O.
- Performance-driven choices should not prevent data from being stored in the desired file format, since this is crucial for later data analysis.
  - Make it easy for application scientist to achieve high performance/scalable I/O.
- Have efficient ways of identifying and selecting certain data for analysis, to help end users cope with the flood of data being produced by these codes.
- Make it easy to introduce new research transport methods into ADIOS.

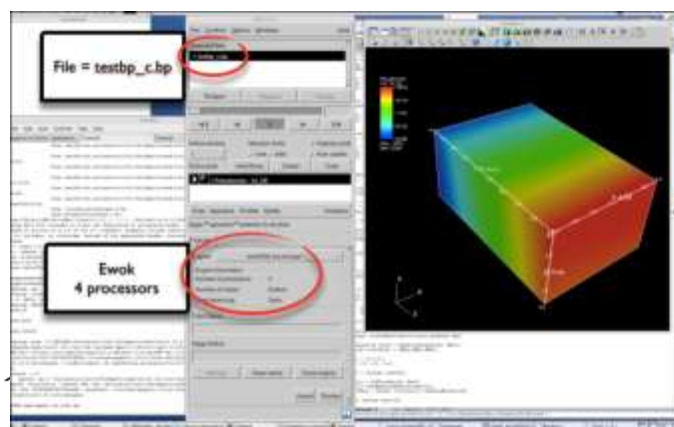
Common I/O Practices in Simulations

- Use netcdf for xy plot diagnostic data.
- Use HDF5, with a schema, for visualization data.
- Use F90 output for Checkpoint data.
  - But this can gives lots of files!
  - Can kill the metadata server!
- Move over to MPI-IO for Checkpoint restart Data.
  - But this can give terrible performance for most users.
  - Small writes can destroy the performance.
- Now there is parallel netcdf
  - What version? Netcdf-4 (from Unicar)? Pnetcdf from Argonne?

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



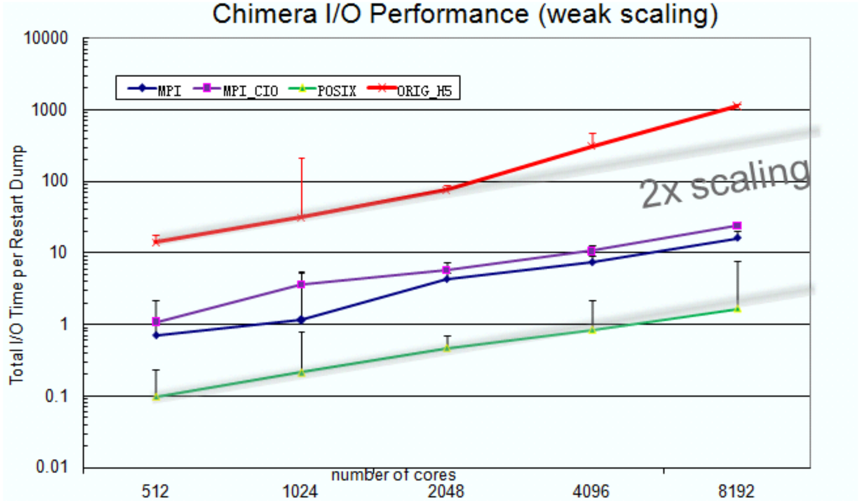


GOAL 1: Make writing FAST and EASY

- Easy because
  - You can define all of your variables in a file, and write many different file formats without knowing netcdf, Posix, MPI-IO, HDF5, etc.
  - You can write attributes, global arrays (across variables) without complex code (aka HDF5, netcdf).
  - You can change I/O methods without reading new manuals.
- FAST
  - ADIOS 1.0 was released on the Cray XT4
    - Speed up the Chimera code over 1000x
    - Speed up the GTC code. (3x for restarts, 25x for analysis output).
    - Speed up the XGC1 code by 5x.

ADIOS 1.0: Write Performance

- Introduce ADIOS.
- GTC: over 35 GB/s on Cray XT4 (peak = 40 GB/s).
- XGC1: over 30 GB/s on XT4
- S3D: over 20 GB/s on XT4.
- Chimera 1000x better than apps first attempt.



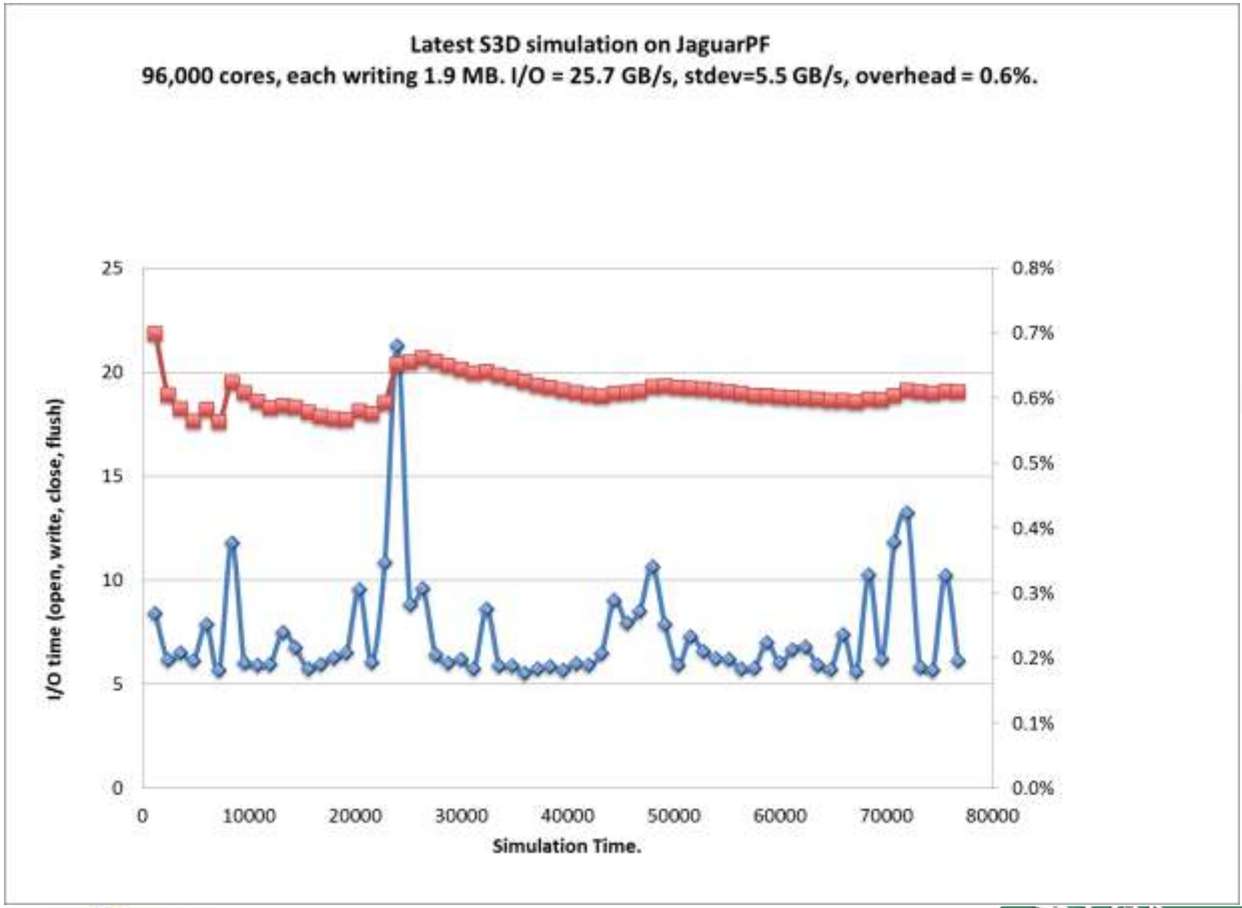
- Plot minimum value from 5 runs with 9 restarts/run
- Error bars show maximum time for the method.
- Parallel HDF5 was hand coded by Chimera team

ADIOS Independent MPI-IO		
Function	*# calls*	*Total Time (sec)*
write	2560	2218.28
MPI_Recv	2555	24.68
MPI_File_open	2560	95.80
other	-	65

Parallel HDF-5		
Function	*# calls*	*Total Time (sec)*
write	144065	33109.67
MPI_Bcast	314800	12259.30
MPI_File_open	2560	325.17
other	-	68.71

ADIOS_AMR Method

- Targets codes which lots of small writes/mpi process.
- In AMR code, each processor can output varied amount of (possibly small) data.
    - Hence, dynamic aggregation technique is needed to achieve good I/O performance.
- Initial results on Cray XT5, 96,000 procs with 1.8MB/proc.
- Total overhead for I/O for S3D = 0.6%.



Some Other Key Enhancements

- File open threaded.
  - Reduce the total I/O time significantly especially for large-scale runs.
- Data is written out into multiple files to overcome Lustre striping limitations.
- Subfiles are transparent to users, and is chosen for optimal performance for HPC.

restart.bp

.restart.sub_dir/restart.bp.0

.restart.sub_dir/restart.bp.1

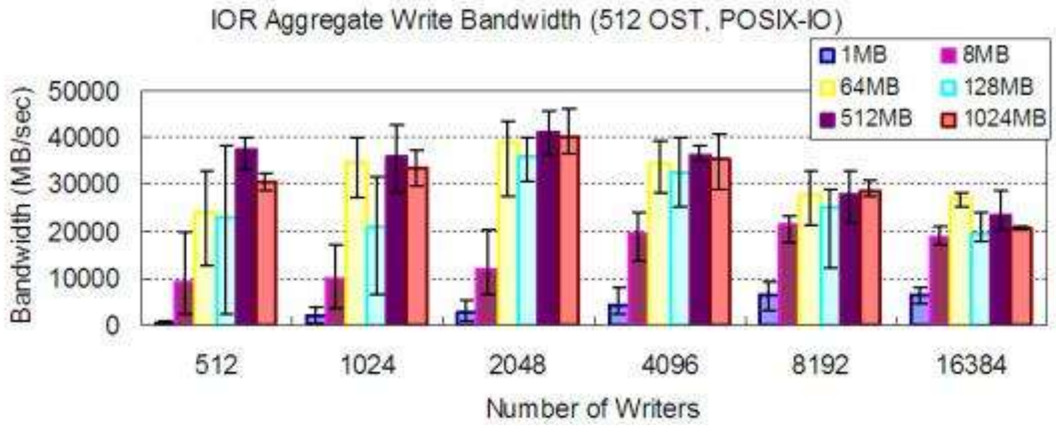
.restart.sub_dir/restart.bp.2

- To read the data back

bpls restart.bp

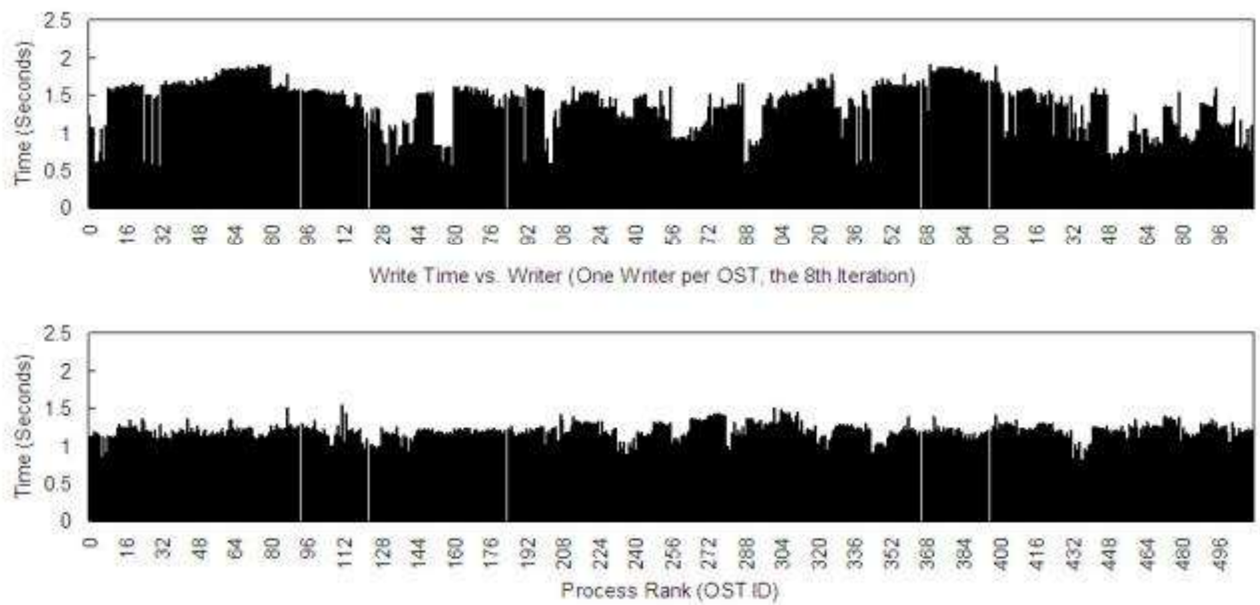
I/O interference

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

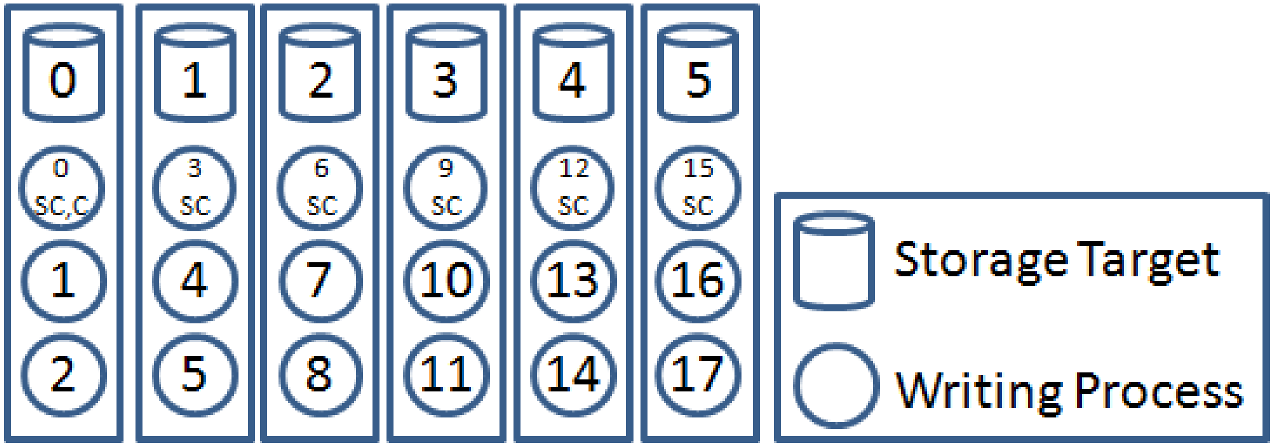


3.44 vs. 1.86 imbalance factor

128 MB/process, 3 minutes apart



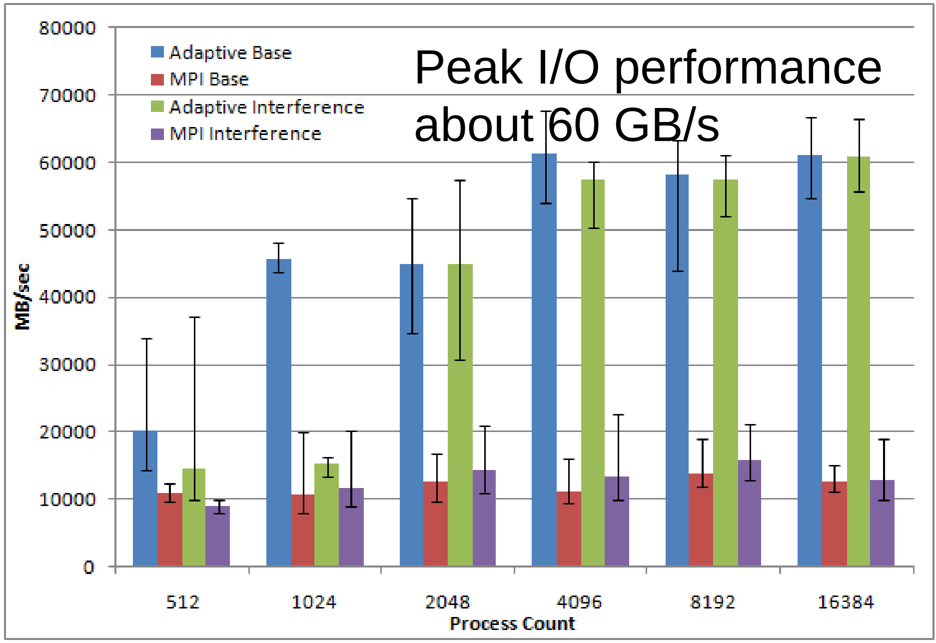
Adaptive Algorithm

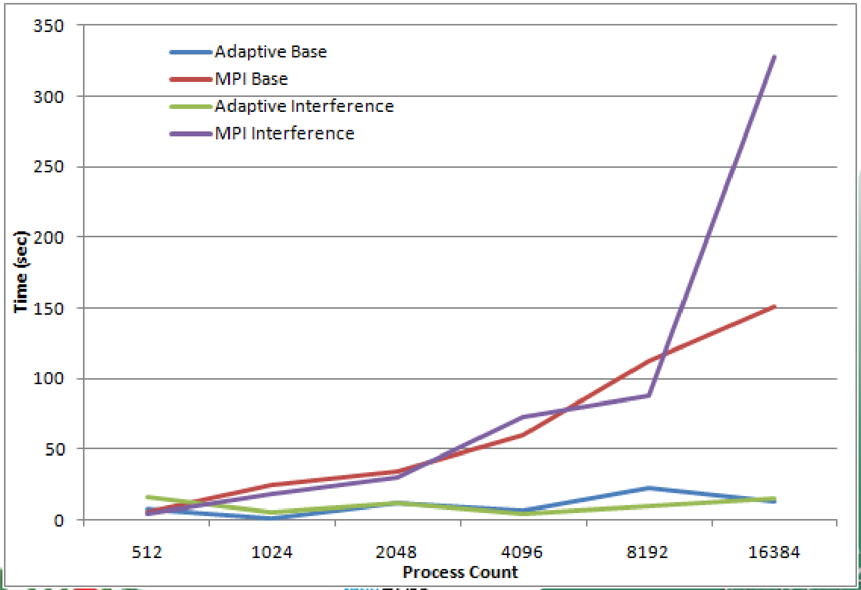


- Break processes into groups, one group per each storage target
- Schedule processes to write individually per group
- Track completion of groups to enable shifting work

Adaptive Method

- New adaptive method meant to handle the variability of the writes.
- Now codes can achieve almost 60 GB/s at scale!
- Peak IOR (gold standard of I/O benchmarking) gets 60 GB/s on a loaded system.

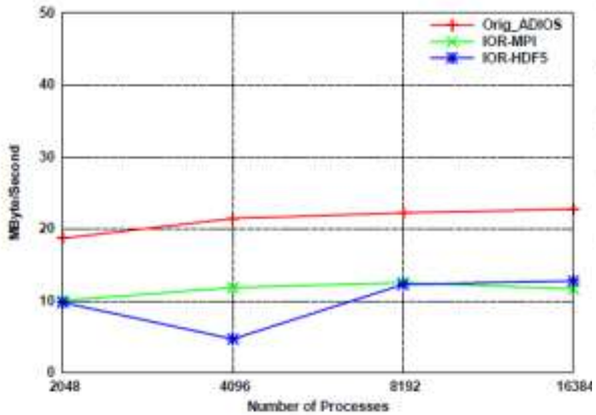




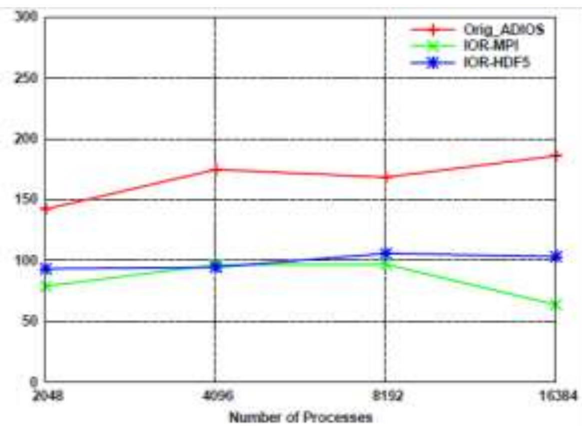
IBM BGP (Intrepid)

- No Changes in ADIOS…
- Write data from a 3D domain decomposition.
- Small = 128 KB, Medium = 1MB, Large = 8 MB (per mpi process)

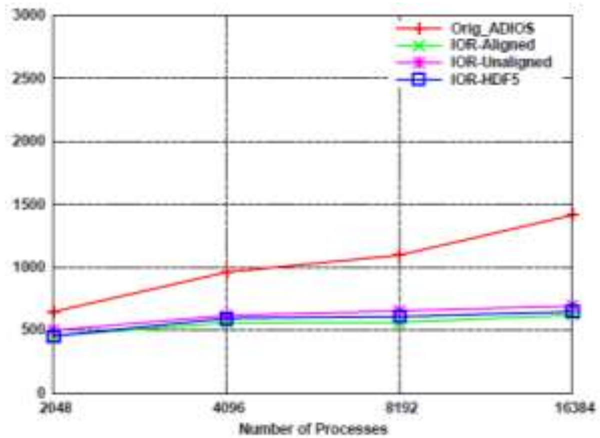
PVFS



(a) Small Message

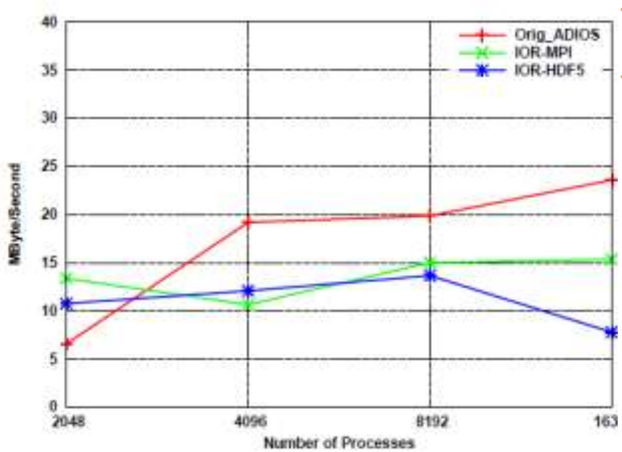


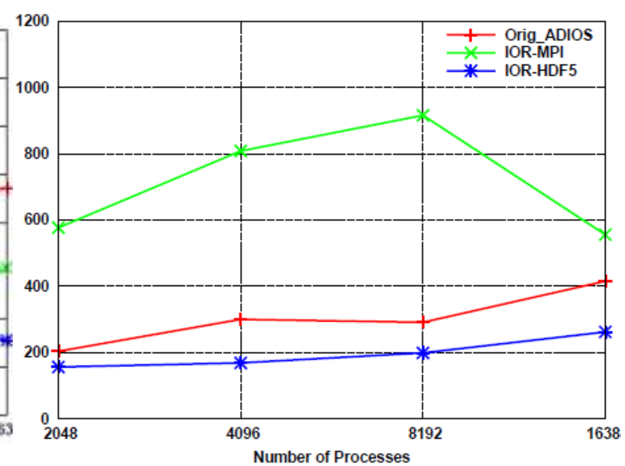
(b) Medium Message

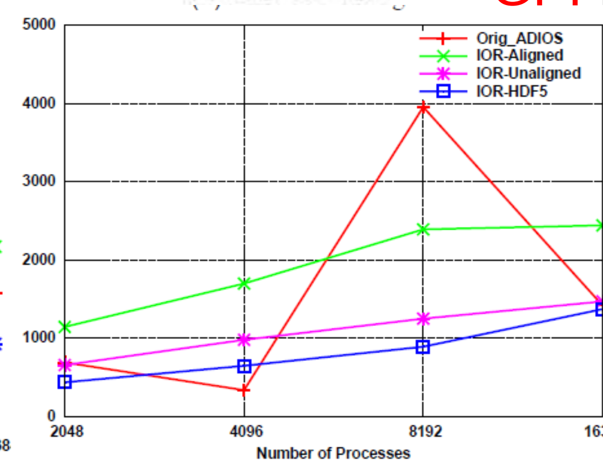


(c) Large Message

GPFS







But what about the read performance?

- ADIOS 1.0 contained a new file format (ADIOS-BP) to help it achieve excellent performance on the Cray XT4.
- The file format was an Application Log file Format (ALF)
  - Has semantic knowledge of the data. (So better than a file system interface).
  - Has knowledge of the file system.
- The feedback from some in the CS community:

- “You will never be able to read in a subsection of a 3d global array
- “Your read performance will be abysmal compared to a logically contiguous file format!”

i.e. write to parallel netcdf, parallel hdf5!

SO………

File formats

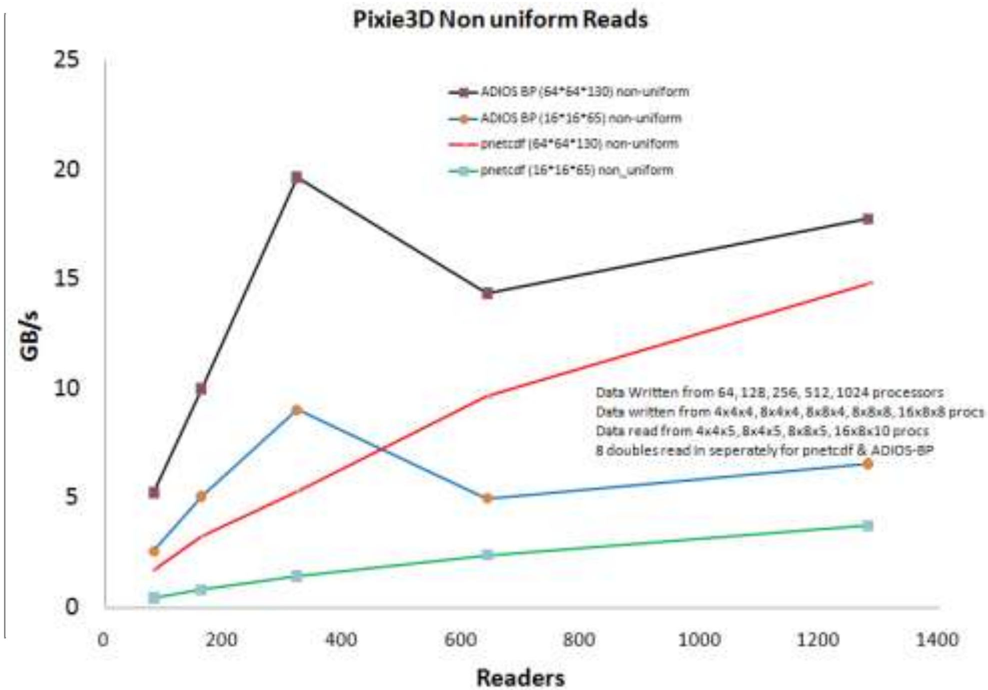
- pNetCDF
  - "right sized" header
  - coordination for each data declaration
  - data stored as logically described
- HDF-5
  - b-tree format
  - coordination for each data declaration
  - single metadata store vulnerable to corruption.
- ADIOS-BP (Binary metadata rich Packed).
  - Individual outputs into "process group" segments.
  - Headers in each process group segment.
  - Metadata indices next
  - Characteristics Integrated into the file format.
  - Index offsets and version flag at end (footer, no header). (Redundant)

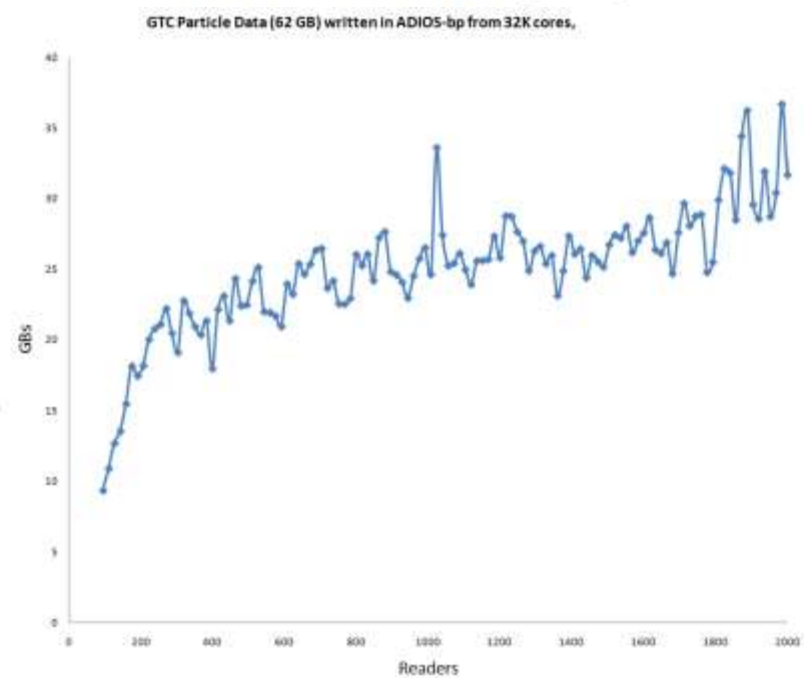
Process Group 1	Process Group 2	...	Process Group n	Process Group Index	Vars Index	Attributes Index	Index Offsets and Version #

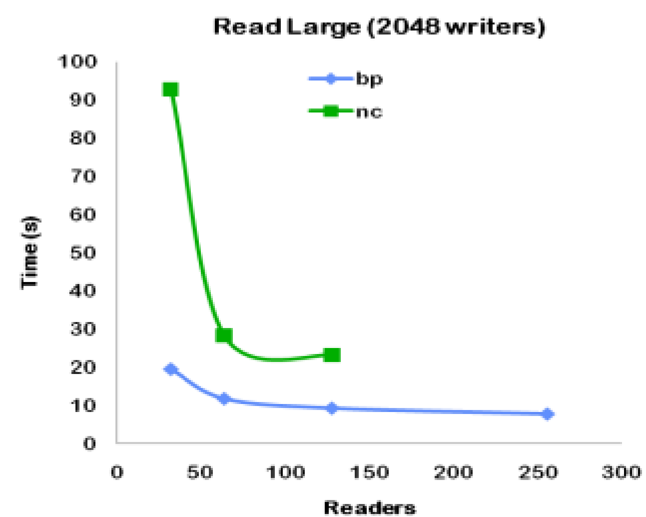
Understand the "typical" read access patterns

- 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 Performance (PDSW 2009)

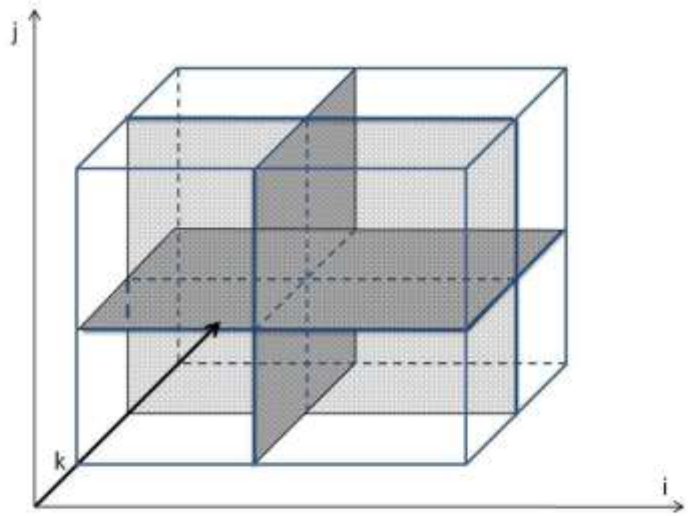


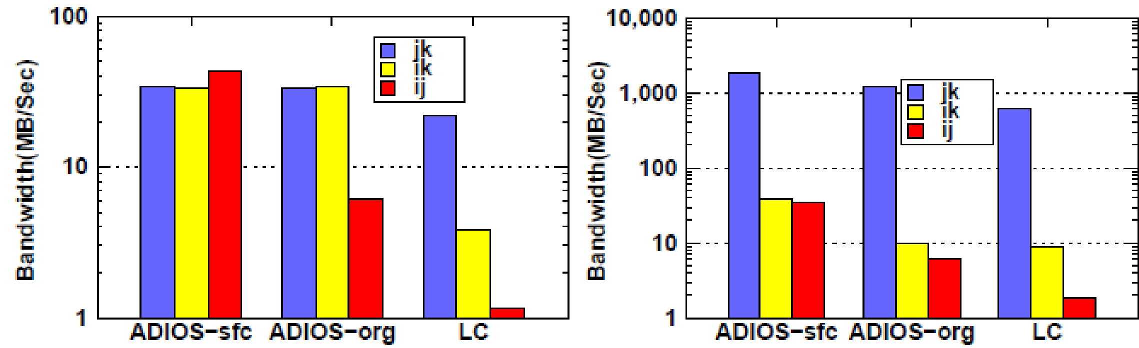




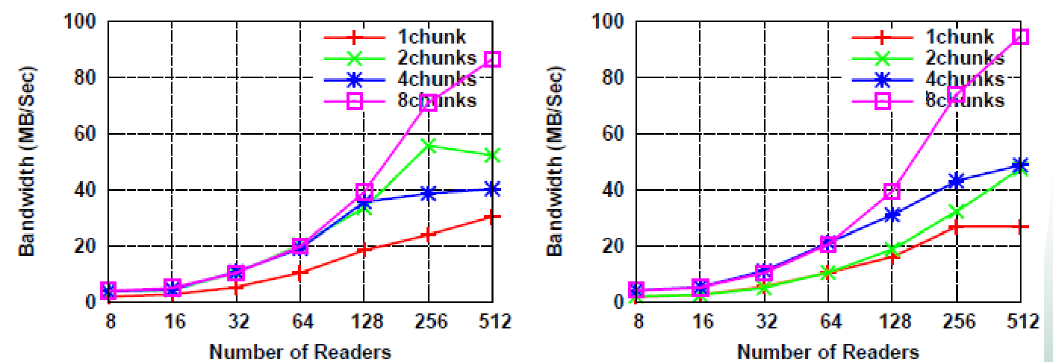
- Read results are quite promising for restarts and analysis data.
- Restarts for small/medium small Pixie3D data always better for BP than pnetcdf.
- But why?

Problem of reading in 2D data from 3D dataset

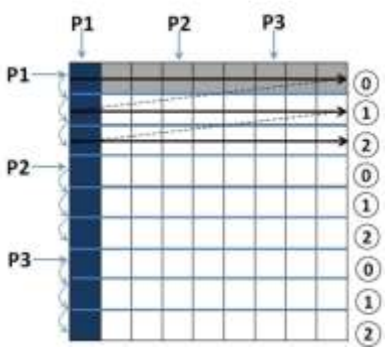


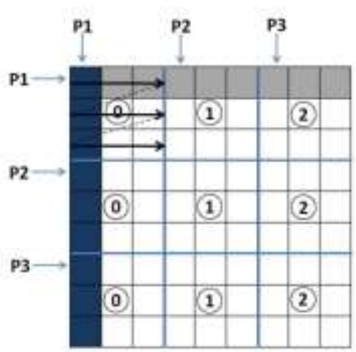


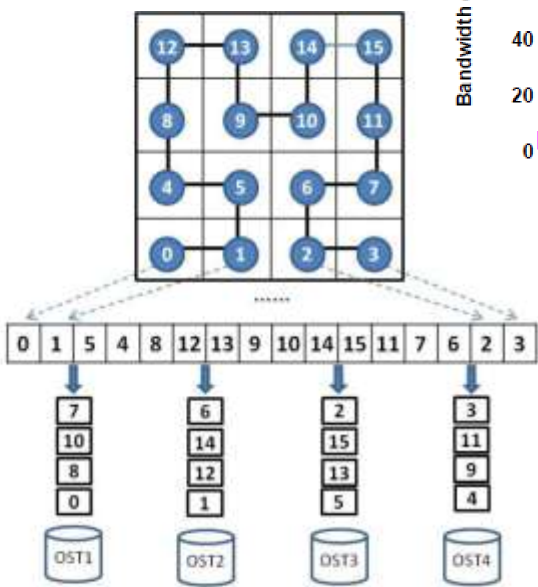
(a) Small(stripes=128) (b) Extra Large(stripes=128)



(a) ik planes (b) ij planes







The BP file format

File info:

of groups: 1

of variables: 32

of attributes: 16

time steps: 102 starting from 1

file size: 132 MB

bp version: 513

endianness: Little Endian

Group record:

double /time {102} = 0 / 0.2 / 0.0990196 / 0.0588536 {MIN / MAX / AVG / STD_DEV}

integer /itime {102} = 0 / 100 / 49.5098 / 29.4268 {MIN / MAX / AVG / STD_DEV}

double /dt {102} = 0.002 / 0.002 / 0.002 / nan {MIN / MAX / AVG / STD_DEV}

integer /nvar scalar = 8

integer /dimensions/nxd+2 scalar = 102

integer /dimensions/nyd+2 scalar = 66

integer /dimensions/nzd+2 scalar = 3

integer /aux/zsize scalar = 3

double /var/v1 {102, 3, 66, 102} = 1 / 1 / 1 / 0 {MIN / MAX / AVG / STD_DEV}

double /var/v2 {102, 3, 66, 102} = -2.07959e-06 / 4.86488e-08 / -8.25872e-07 / 5.53109e-07 {MIN / MAX / AVG / STD_DEV}

double /var/v3 {102, 3, 66, 102} = -1.48595e-06 / 1.48595e-06 / 2.0967e-10 / 2.30334e-07 {MIN / MAX / AVG / STD_DEV}

double /var/v4 {102, 3, 66, 102} = -1.66745e-08 / 1.66745e-08 / -1.32551e-12 / 2.71603e-09 {MIN / MAX / AVG / STD_DEV}

string /dimensions/nxd+2/description attr = "3D array size in X direction including two ghost cells on the faces"

New characteristics into ADIOS-BP

- Histograms can be automatically generated, in the footer (no added cost in writing).
  - <analysis group="temperature" var="temperature" break-points="0, 100, 200, 300" />
  - <analysis group="temperature" var="temperature" min="0" max="300" count="3"/>
  - Both the above inputs create bins [0, 100), [100, 200), [200, 300)
- Min/max over time steps.
- Averages.
- Easy to add new characteristics.

Example ADIOS code. (XML)

```c
MPI_Init (&argc, &argv);
    MPI_Comm_rank (comm, &rank);
    MPI_Comm_size (comm, &size);
    for (i = 0; i < NX; i++)
        t[i] = rank*NX + i;
    strcpy (filename, "adios_global.bp");
    adios_init ("adios_global.xml");
    adios_open (&adios_handle, "temperature", filename, "w", &comm);
    #include "gwrite_temperature.ch"
    adios_close (adios_handle);
    MPI_Barrier (comm);
    adios_finalize (rank);
    MPI_Finalize ();
}
```

```xml
<?xml version="1.0"?>
<adios-config host-language="C">
  <adios-group name="temperature" coordination-communicator="comm" >
    <var name="NX" type="integer"/>
    <var name="size" type="integer"/>
    <var name="rank" type="integer"/>
    <global-bounds dimensions="size,NX" offsets="rank,0">
      <var name="temperature" gwrite="t" type="double" dimensions="1,NX"/>
    </global-bounds>
    <attribute name="description"  value="Global array" type="string"/>
</adios-group>
<method group="temperature" method="MPI“/>
<buffer size-MB="2" allocate-time="now"/>
</adios-config>
```

1 more Write example (No XML)

```
MPI_Init (&argc, &argv);
MPI_Comm_rank (comm, &rank);
MPI_Comm_size (comm, &size);
Gbounds = sub_blocks * NX * size;
strcpy (filename, "adios_global_no_xml.bp");
adios_init_noxml ();
adios_allocate_buffer (ADIOS_BUFFER_ALLOC_NOW, 10);
adios_declare_group (&m_adios_group, "restart", "iter", adios_flag_yes);
adios_select_method (m_adios_group, "MPI", "", "");
adios_define_var (m_adios_group, "NX","", adios_integer,0, 0, 0);
adios_define_var (m_adios_group, "Gbounds","", adios_integer,0, 0, 0);
for (i=0;i<sub_blocks;i++) {
  adios_define_var (m_adios_group, "Offs","", adios_integer,0,0,0);
  adios_define_var (m_adios_group, "temp","",adios_double,"NX","Gbounds","Offs");
}
adios_open (&m_adios_file, "restart", filename, "w", &comm);
```

1 more Write example (No XML)

```
adios_groupsize = sub_blocks * (4 + 4 + 4 + NX * 8);
adios_group_size (m_adios_file, adios_groupsize, &adios_totalsize);
adios_write(m_adios_file, "NX", (void *) &NX);
adios_write(m_adios_file, "Gbounds", (void *) &Gbounds);
for (block=0;block<sub_blocks;block++) {
 Offs = rank * sub_blocks * NX + block*NX;
 adios_write(m_adios_file, "Offs", (void *) &Offs);
 for (i = 0; i < NX; i++)
  t[i] = Offs + i;
 adios_write(m_adios_file, "temp", t);
}
adios_close (m_adios_file);
MPI_Barrier (comm);
adios_finalize (rank);
MPI_Finalize ();
return 0;
}
```

Read Example

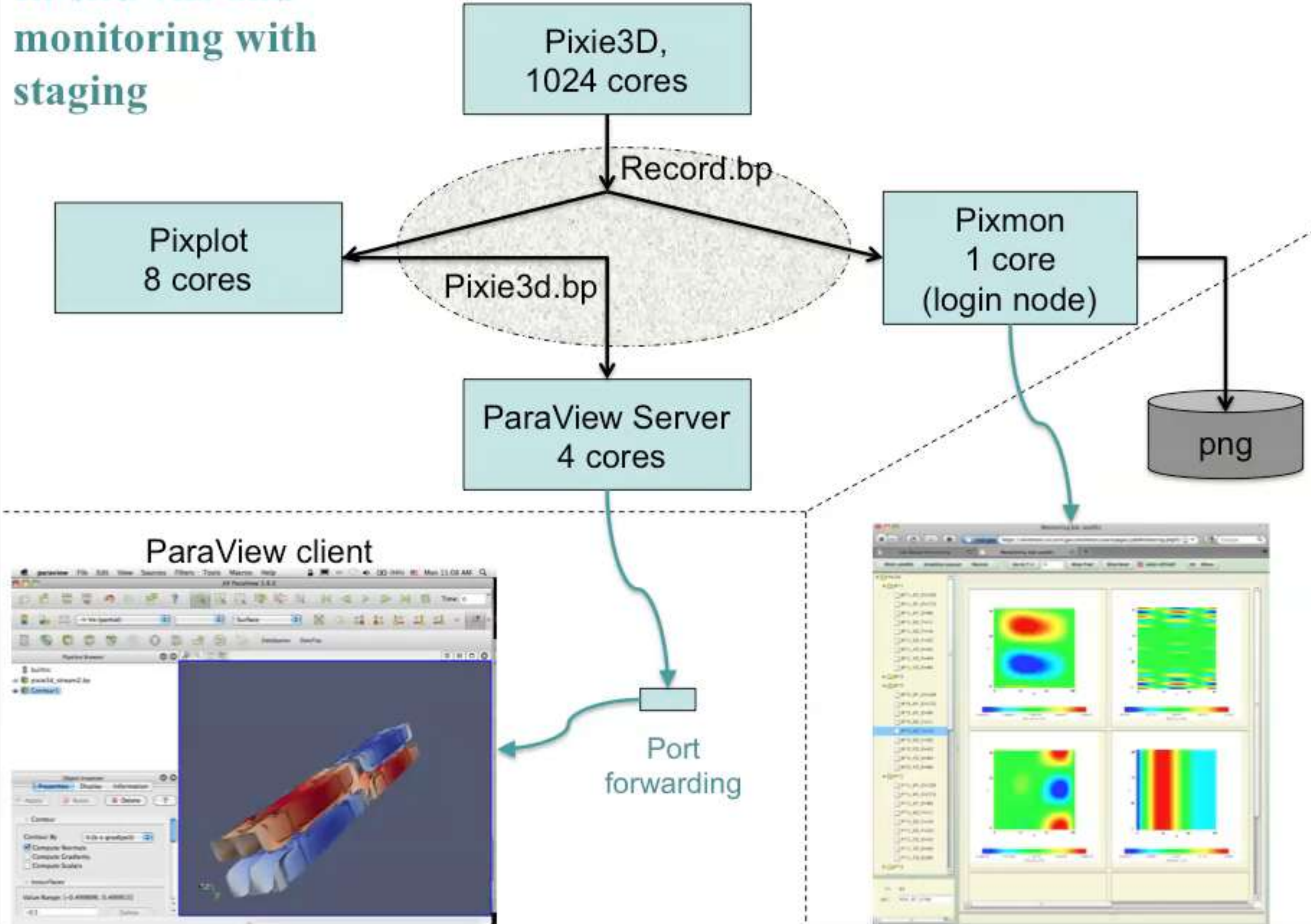
```
char        filename [256];
   int         rank, size, i, j;
   MPI_Comm    comm = MPI_COMM_WORLD;
   void * data = NULL;
   uint64_t start[2], count[2], bytes_read = 0;
   MPI_Init (&argc, &argv);
   MPI_Comm_rank (comm, &rank);
   MPI_Comm_size (comm, &size);
   ADIOS_FILE * f = adios_fopen
("adios_global.bp", comm);
   ADIOS_GROUP * g = adios_gopen (f,
"temperature");
   ADIOS_VARINFO * v = adios_inq_var (g,
"temperature");
   /* Using less readers to read the global array
back, i.e., non-uniform */
   uint64_t slice_size = v->dims[0]/size;
   start[0] = slice_size * rank;
```

```
if (rank == size-1)
slice_size = slice_size + v->dims[0]%size;
start[1] = 0;  count[1] = v->dims[1]; count[0] = slice_size;
   data = malloc (slice_size * v->dims[1] * sizeof
(double));
   bytes_read = adios_read_var (g, "temperature", start,
count, data);
   for (i = 0; i < slice_size; i++) {
      for (j = 0; j < v->dims[1]; j++)
         printf (" %6.2g\n", * (double *)data + i * v-
>dims[1] + j);
   }
   free (data);
   adios_gclose (g);
   adios_fclose (f);
   MPI_Barrier (comm);
   MPI_Finalize ();
   return 0;
}
```

QuickTime Player File Edit View Share Window Help Stop Recording (98%) Tue 5:16 PM

ParaviewDemo.png

In-situ viz. and monitoring with staging



Conclusions

- I/O has been a major problem of many codes at the OLCF, ALCF, and NERSC.
- Business as usual has not been working.
- ADIOS 1.2 has proven to solve most of the I/O methods by combining “state-of-the-art” computer science research with hardened solutions delivered in an open source package.
  - Can save valuable time reading data, writing data, understanding data.
- The philosophy has allowed many institutions to develop for ADIOS independently.
- Please email klasky@ornl.gov if you are interested in using ORNL.
  - Contact help@nccs.gov if you have any problems.

EFFIS

EU-US Workshop on Software Technologies for Integrated Modeling in Fusion

Sweden

12/2//2010

Scott A. Klasky

klasky@ornl.gov

For the CPES team

Physics requirements for a FSP framework

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

Specific FSP needs addressed

- Couple multiple services:
  - Coupling physics services on the same platform,
  - “real-time” analysis, visualization on multiple/remote platforms.
  - Coupling physics services on different platforms. (file-based)
- Coupling can be memory-to-memory or via files depending on frequency of coupling and platforms where codes are running
  - Coupling may involve transformation of data
  - Tool needed: fast memory-based coupling capability. (ADIOS)
  - Tool needed: workflow system to coordinate file movement and processing between multiple platforms. (Kepler)

Specific FSP needs addressed

- Monitor large-scale simulations while they are running
  - Process and visualize timestep/checkpoint data as soon as they are generated
  - Move data between platforms
  - Tool needed: modular high-performance I/O to extract data quickly and efficiently. **(ADIOS synchronous and asynchronous methods)**
  - Tool needed: a workflow system to manage data movement, processing, and generating graphs and images. **(Kepler)**
    - Workflow system must be robust, run for days, and recover from transient failures.
  - Tool needed: a web-based display capability that is fast and effective. **(eSimMon dashboard)**

Why SOA

- **Data Challenges at Yahoo! Ricardo Baeza-Yates & Raghu Ramakrishnan, Yahoo! Research**
  - Data diversity –them: text, streams, structured data, 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- them: 350K requests/sec! and growing via asynchrony, loose coupling, weak consistency; us: decoupling via ADIOS, data staging, …
- Leverage file system's high bandwidth- them: DFS++; us: Lustre
- Use multiple ways to represent data- them: row/column stores, DHTs us: BP, tuple spaces, …
- Deal with reliability- them: DFS based replication/recoverability; us: robust data format , checkpointing
- Make it easy to use- them: self-management, self-tuning; us: adaptive I/O
- Make it easy to change- them: adaptability, i.e., new analyses readily added (us: that's the whole point of the EFFIS SOA)

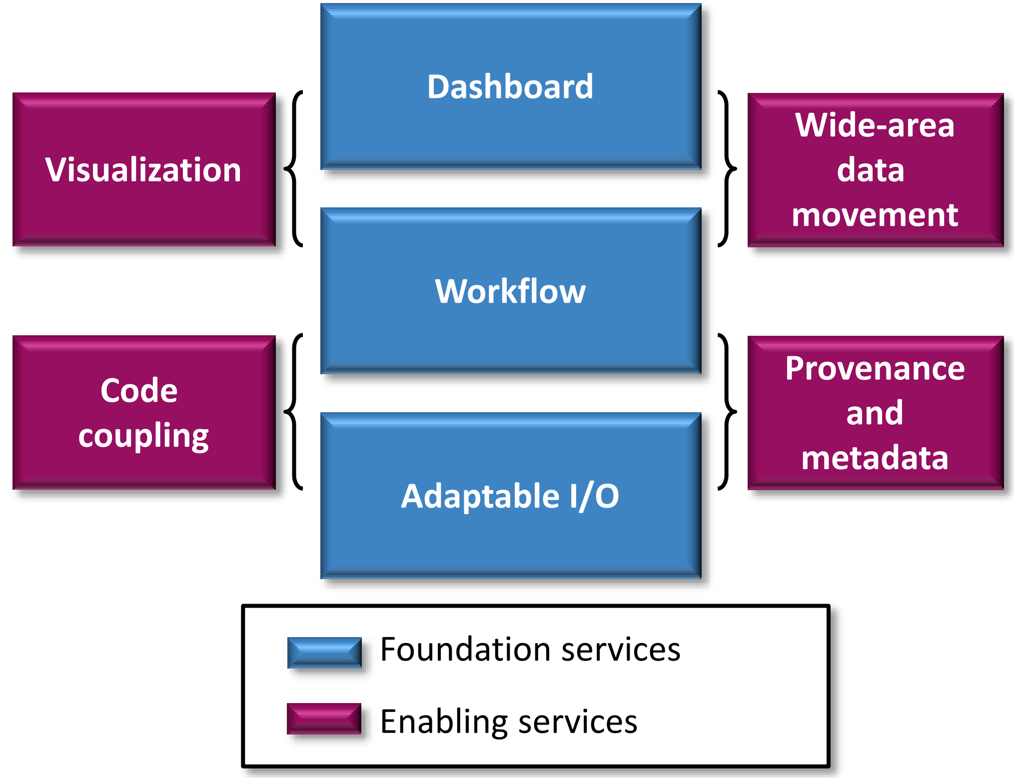
If Yahoo and Google can do it, so can we!

Complexity leads to a SOA approach

- Concept develop for the enterprise
- 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 from coordination across codes and research teams
- 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
- EFFIS Innovation:
  - Minimizing performance impact
  - Addressing unique requirements of FSP specifically and scientific computing in general

EFFIS Services

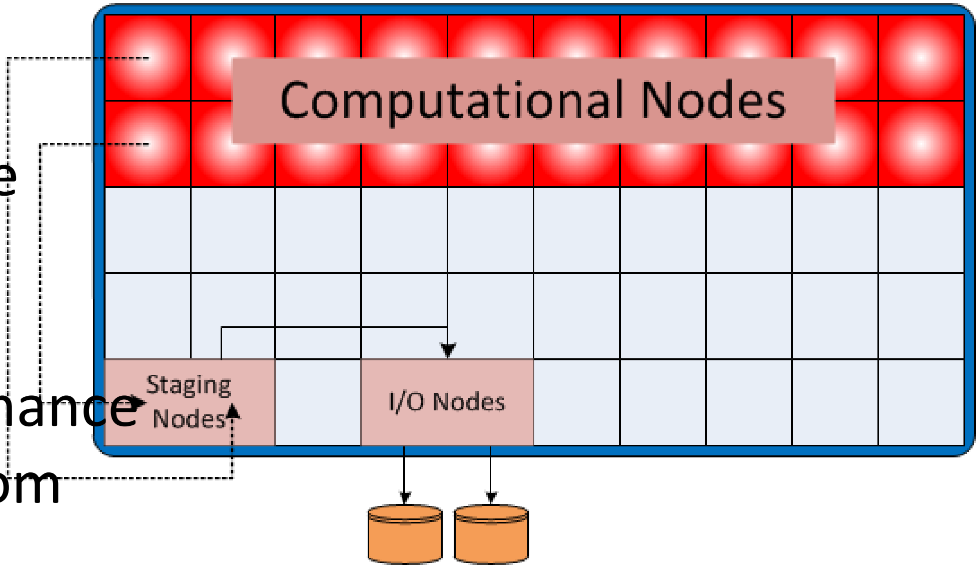
- Adaptable I/O
- Workflows
- Dashboard
- Provenance
- Code coupling
- WAN data movement
- Visualization



Approach: Place highly annotated, fast, easy-to-use I/O methods in the code, which can be monitored and controlled; have a workflow engine record all of the information; visualize this on a dashboard; move desired data to the user's site; and have everything reported to a database.

Benefit: automate complex tasks, and allow users to interact through simple interfaces that expose physics products remotely over the web.

ADIOS DataTap method for asynchronous 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
- ADIOS includes 3 methods for I/O staging
  - DataTap
  - DataSpaces
  - NSSI

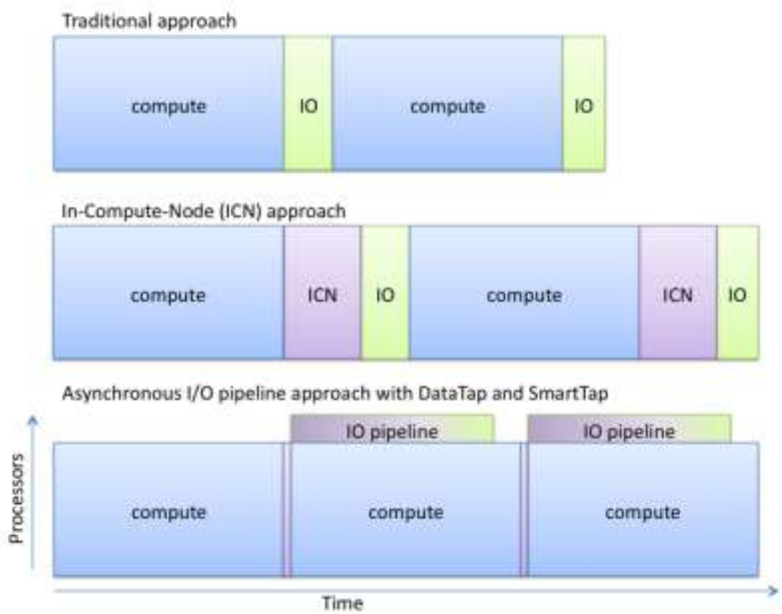
When do you need I/O staging on a large scale machine?

- Poor data layout (from a file system POV) from the code writing to disk.
- Very bad balance of I/O bandwidth and system speed.
  - Currently no production codes have needed this on the Cray XT4, and XT5 and NERSC, ORNL.
- When the data is very large, but is not frequent
  - Example: A code wants to write 54 TB of data from 130K cores.
  - On XT5 with I/O speed at 60 GB/s (system MAX), 25% of time is spent in I/O.
  - But: You can't make a staging area large enough: 42% of the processes would just be used for staging.
- SO

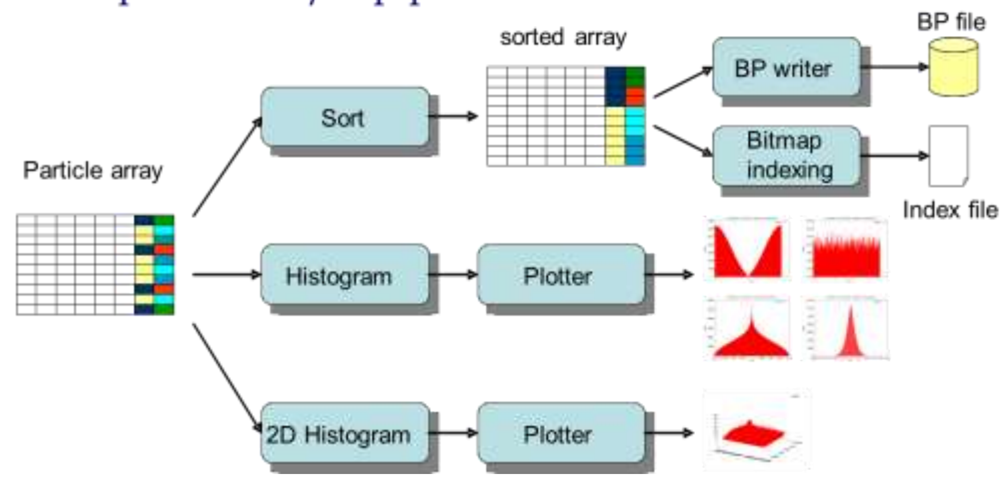
Staging is good for either

- Asynchronous data movement from simulation to a small staging area.
  - We will not be able to stage all of the data, and we can't buffer all of the data on the compute nodes.
  - Need to use asynchronous movement, and this must be scheduled with the MPI communication in your code.
    - Adios_start_calculation, adios_stop_calculation, adios_end_iteration
    - Tell the data movement scheduler when to move the data so it doesn't interfere with the communication in the simulation.
- The creation of I/O pipelines.

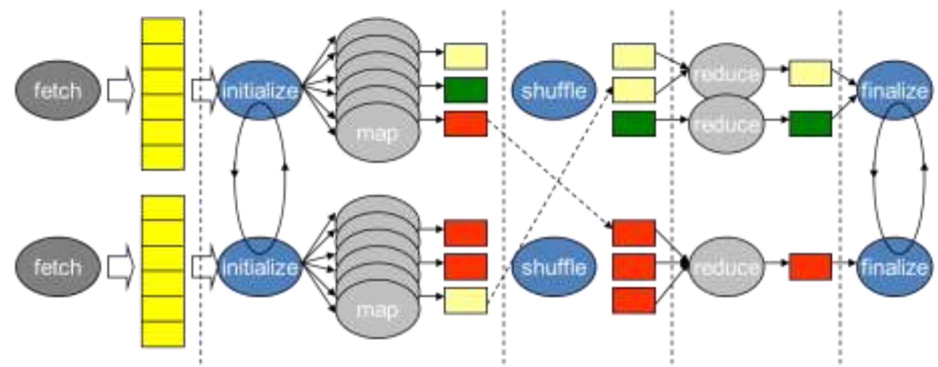
Creation of I/O pipelines to reduce file activity



Example of an I/O pipeline



Streaming Processing in Staging Area

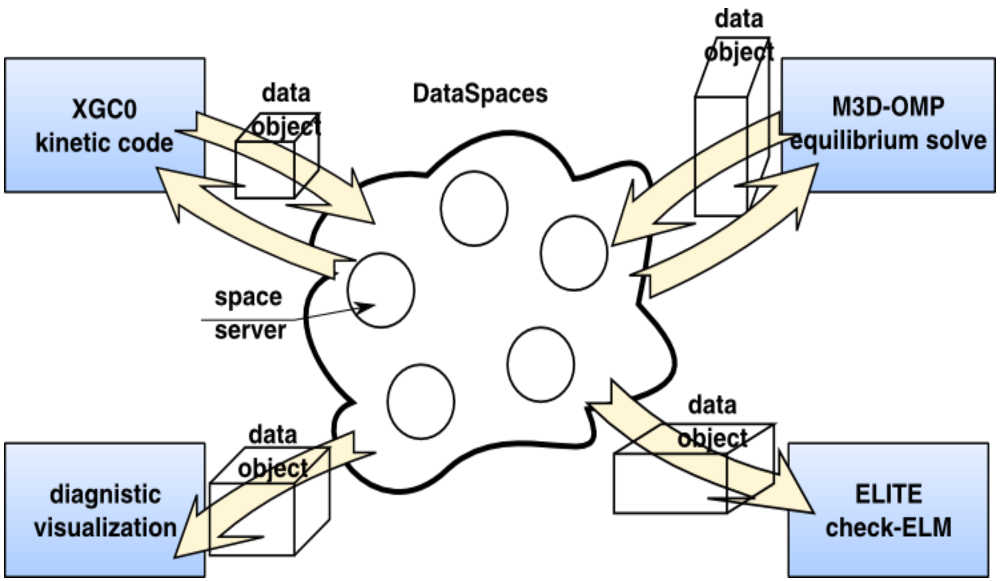


Differences with MapReduce:

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

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



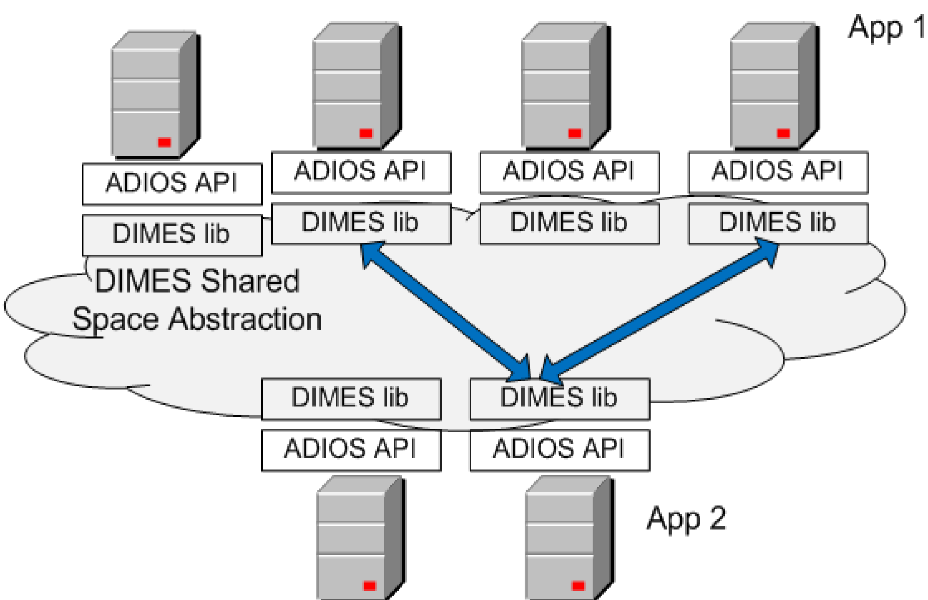
- Supports complex geometry-based queries
- In-space (online) data transformation and manipulations
- Robust decentralized data analysis in-the-space

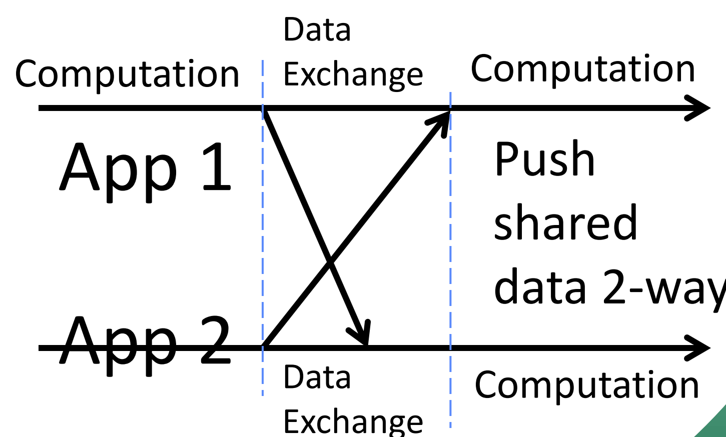
ADIOS with DIMES for in-memory latency-sensitive coupling (DIstributed MEmory Space)

- Motivation:
  - In ***tightly-coupled*** simulation workflow, synchronous data exchange has a strong requirement for low latency.
  - ***DIMES*** enables RDMA-based direct process-to-process data transfer between coupled applications.

data flow: Sender App → Receiver App

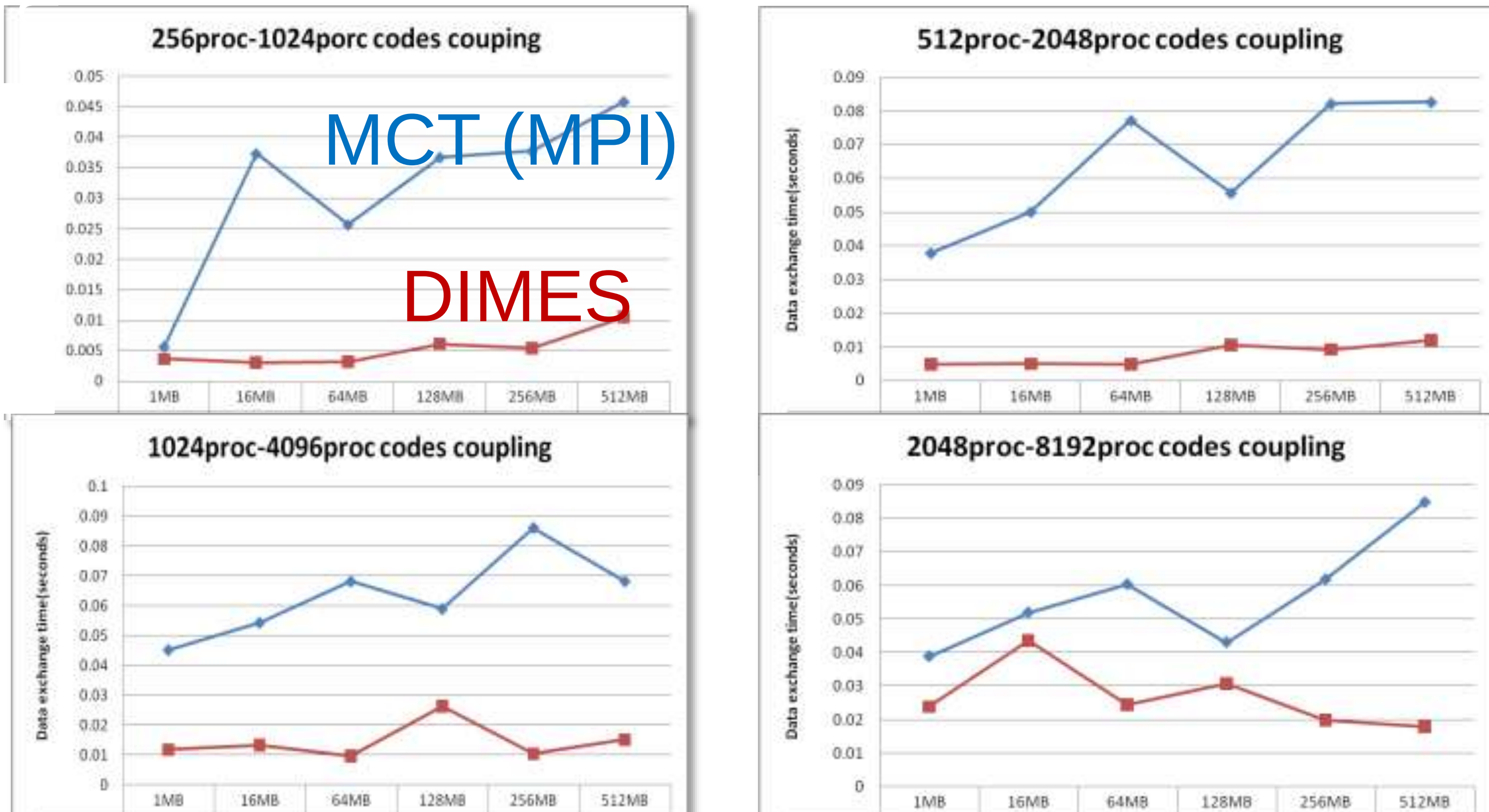
- **DIMES System Architecture:**





DIMES: Distributed Memory Space for code coupling

- Performance evaluation (DIMES vs MCT): Jaguar
- A global 2D array of size *M* is redistributed from app1 (runs on *N1* cores) to app2 (runs on *N2* cores), and both apps have (block,block) data distribution.



How does it work?

sends

```
call adios_open (adios_handle,
 "writer2D", fn, "w", group_comm,
 adios_err)
#include "gwrite_writer2D.fh"
call adios_close (adios_handle,
 adios_err)
```

- Generate the XML file to map F90/C variables to names.

```
<adios-group name="writer2D" >
 <global-bounds
 dimensions="dim_x_global,dim_y_global"
 offsets="offs_x,offs_y">
 <var name="xy" type="real"
 dimensions="dim_x_local,dim_y_local"/>
</global-bounds>
</adios-group>
<transport group="writer2D" method "DART" />
```

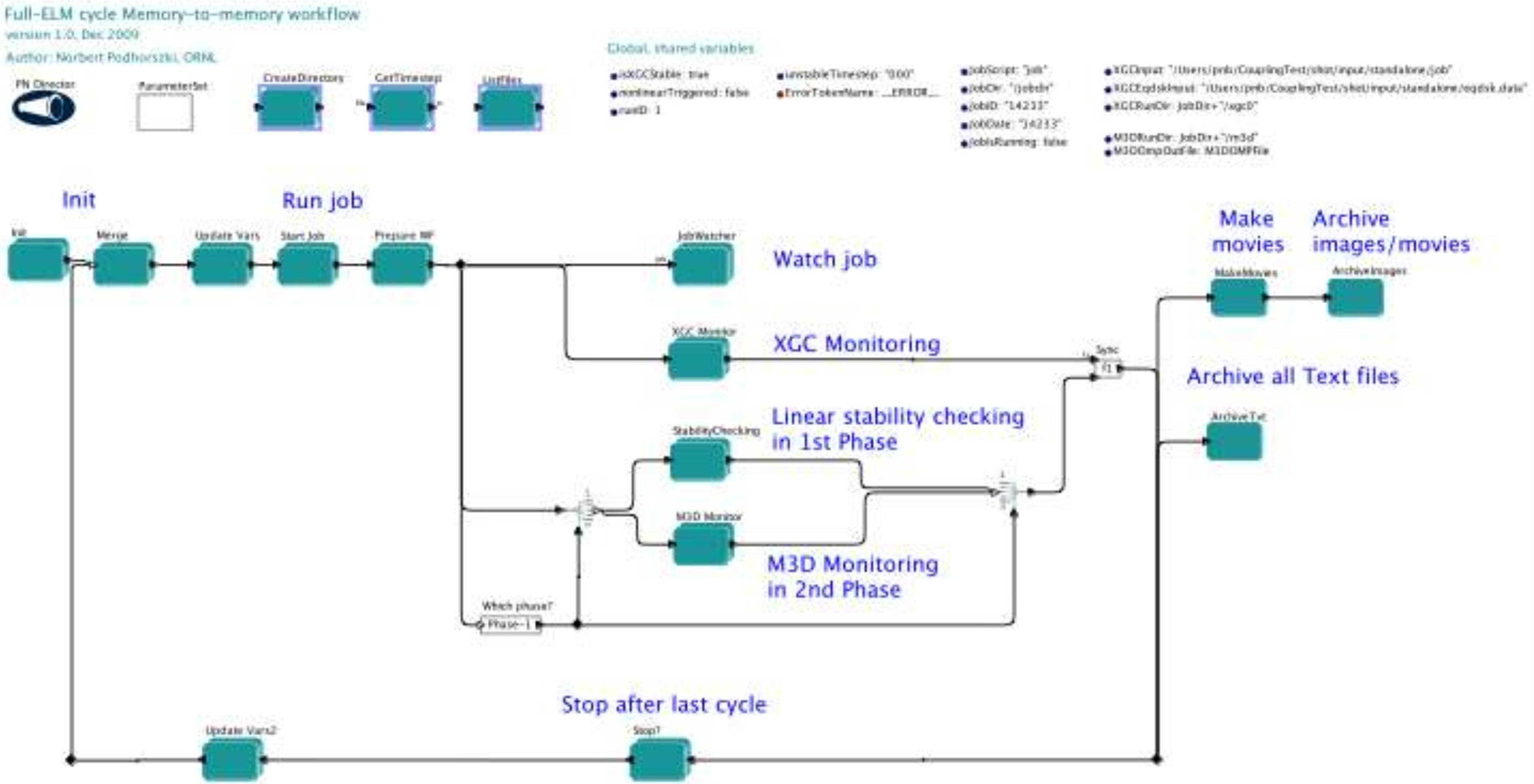
receives

```
call adios_set_read_method ( DART ,ierr)
call adios_read_init (group_comm, ierr)
call adios_fopen (fh, fn, group_comm, gcnt,
    adios_err)
call adios_gopen (fh, gh, "writer2D", vcnt,
    acnt, adios_err)
call adios_read_var (gh, "dim_x_global",
    offset, readsize, dim_x_local,
    read_bytes)
call adios_read_var (gh, "dim_y_global",
    offset, readsize, dim_y_local,
    read_bytes)
call adios_read_var (gh, "xy", offset,
    readsize, xy, read_bytes)
call adios_gclose (gh, adios_err)
call adios_fclose (fh, adios_err)
```

Now we have memory-to-memory coupling

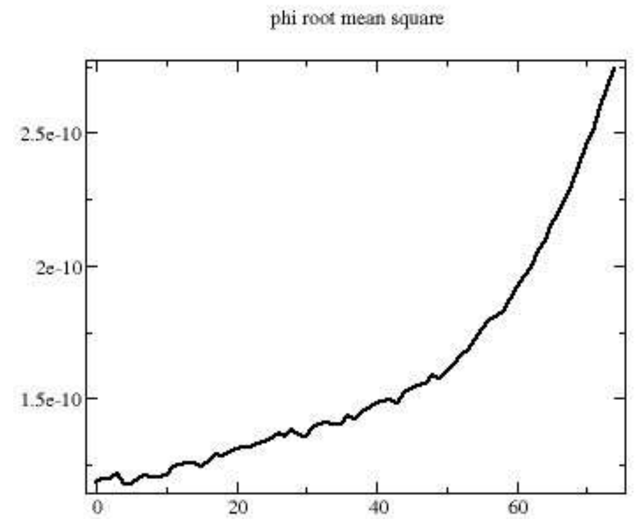
This can also be done with just APIs (no XML)

Example: Coupling workflow (memory-to-memory)



We want the workflow to

- get information about what data is exchanged and when between the codes
  - record data lineage of exchanged data objects (variables)
  - make plots from statistical values
    - e.g. min/max of variables
  - record data lineage of statistical values to image files to allow analysis on those values

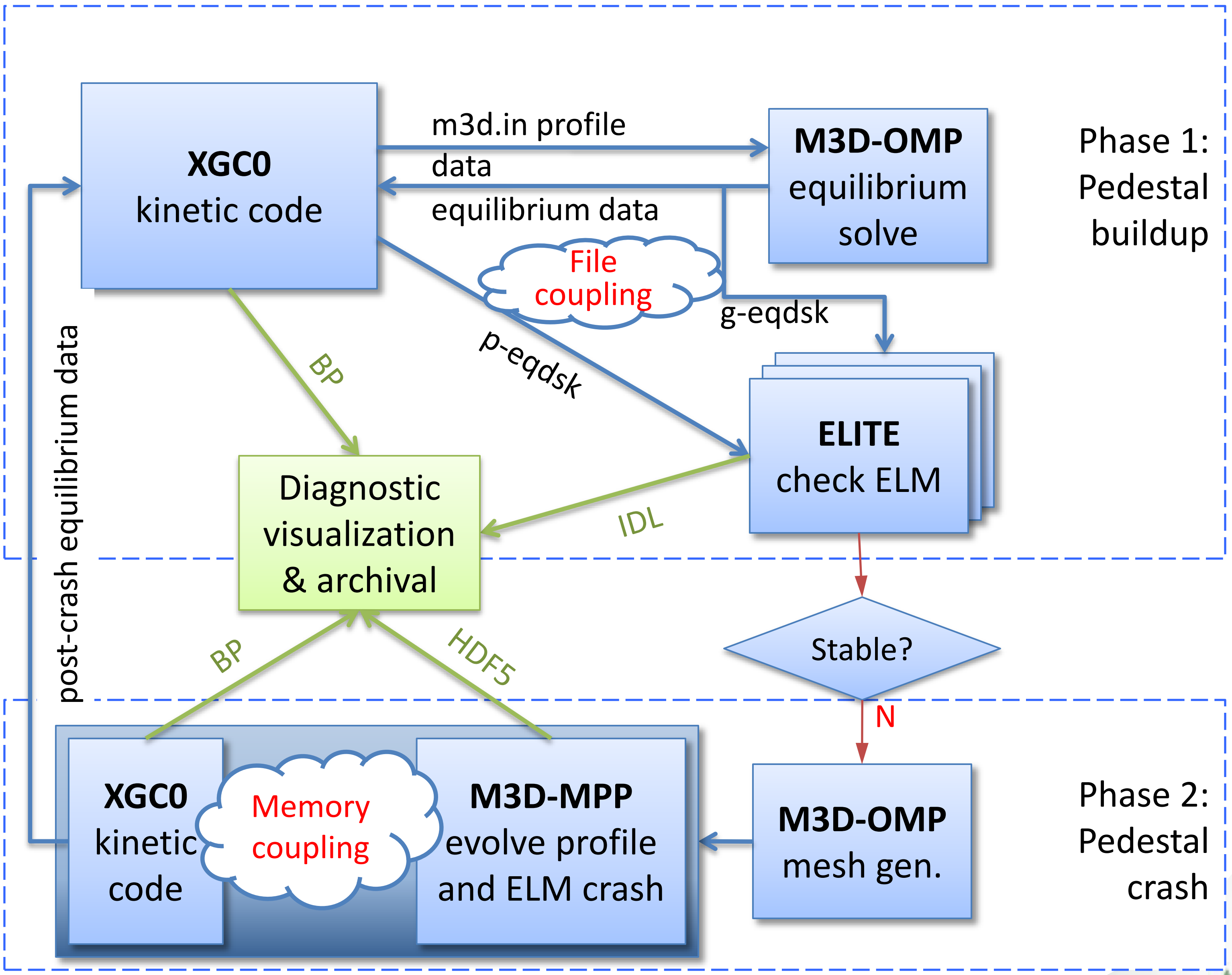


- switch from memory-to-memory coupling to a slower file-based coupling and generate more detailed diagnosis on specific conditions

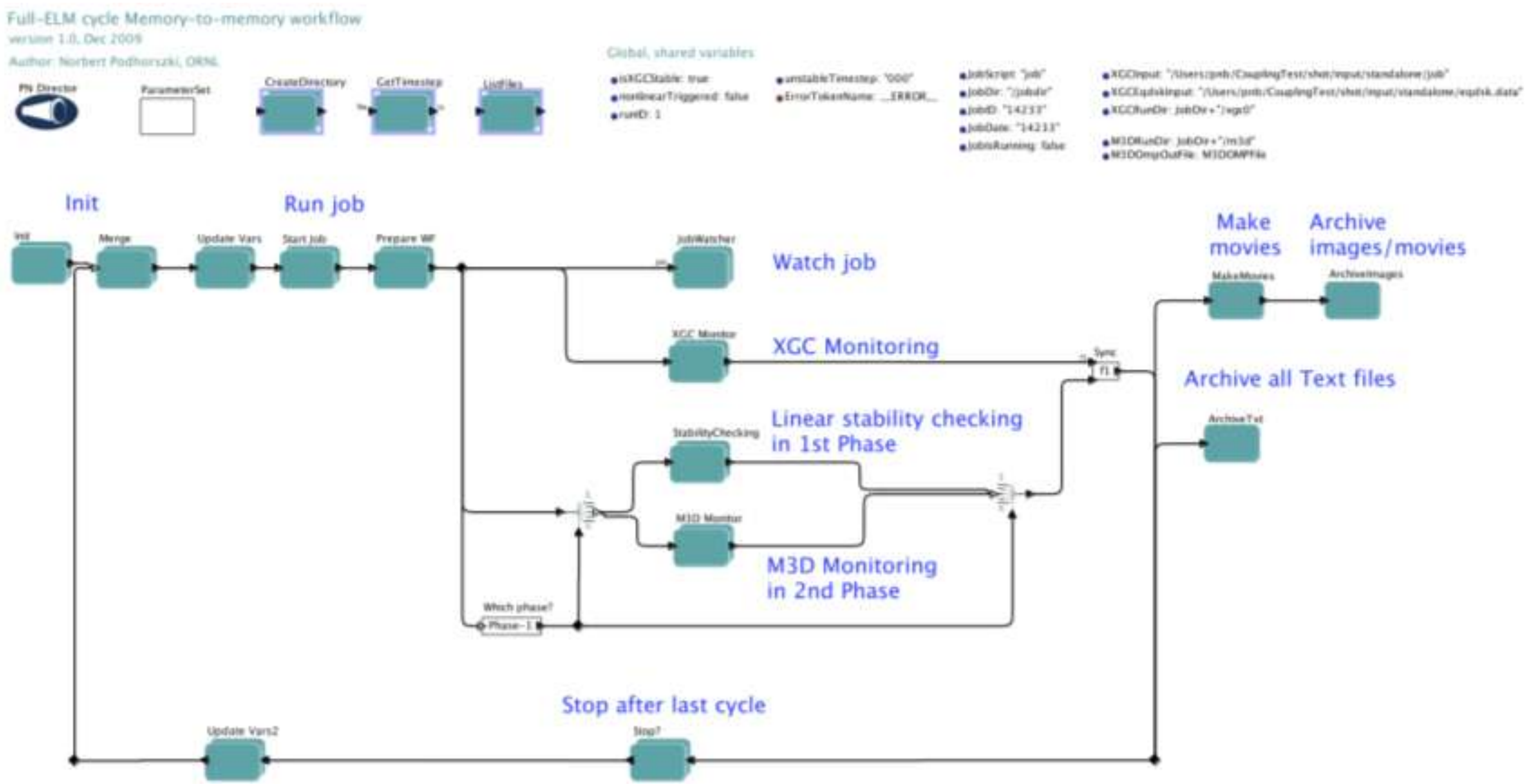
Kepler directors necessary in CPES/FSP

- Support for pipeline parallelism, Process Network (PN)
  - Each actor of the workflow can perform concurrently with other actors. This enables the same workflow to be used repeatedly for a stream of inputs.
- Support for dynamic firing of actors: Dynamic Dataflow (DDF)
  - DDF models enable branching and looping (conditionals). The workflow is sequential.

Full-ELM coupling scenario (divertor heat-load study)



Coupling workflow (memory-to-memory)



Use of GSI Certificates

- PNNL provided GSI extension of org.kepler.ssh
  - Kepler 2.0 supports these certificates
- ORNL installed GSI servers on Jaguar/Ewok and a specialized MyProxy server
  - Jaguar/Ewok are now accessible from NCCS machines, using a DOE certificate
- Full-ELM coupling workflow and XGC monitoring workflow now runs with Kepler 2.0
  - > 2 days coupling is possible if user has DOE certificate

Key features of ADIOS for the coupling

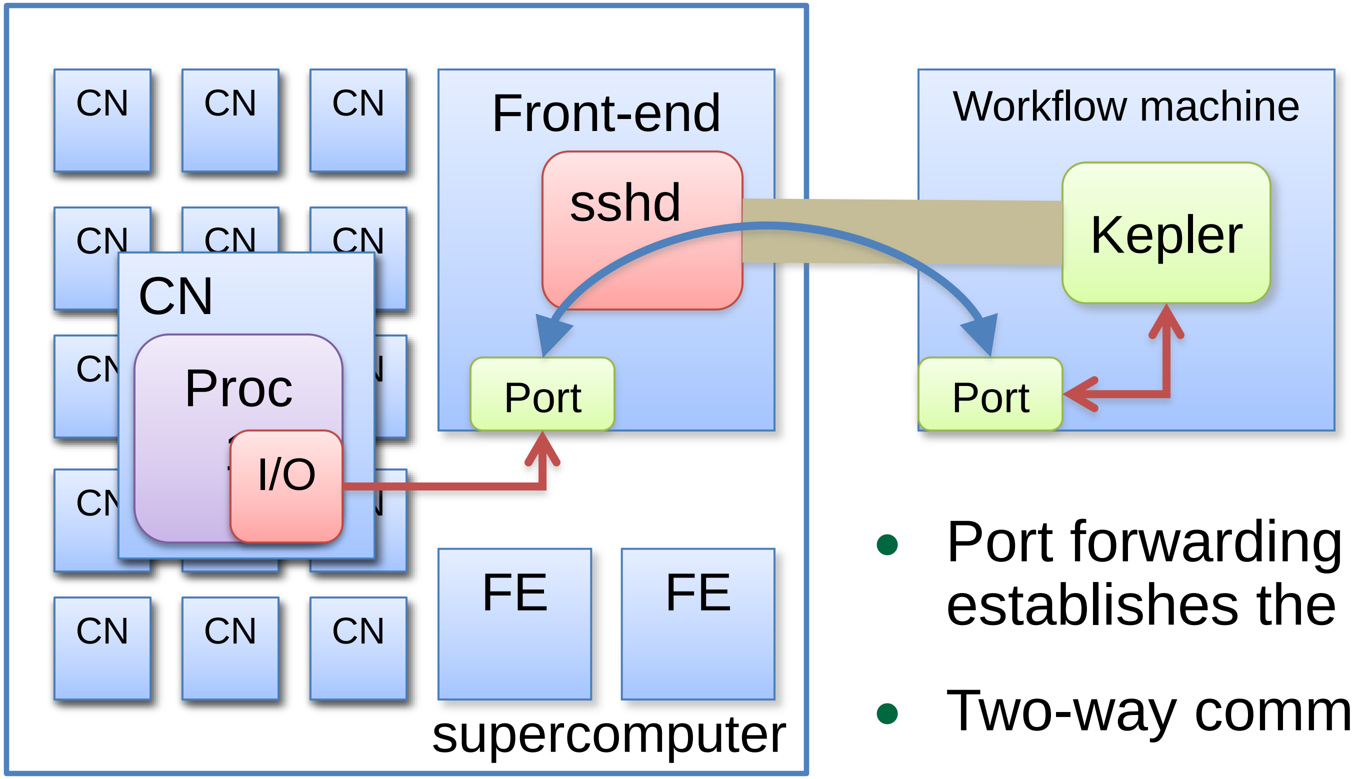
- Data exchange between codes is I/O, although via memory
  - ADIOS plugin can be developed for that (DART)
  - one application code for both modes of coupling
- Switching from memory-to-memory to file-based coupling requires behavior change of the applications
  - ADIOS can switch between plugins at runtime without the knowledge of the applications
- Getting information about the data exchanged through memory
  - ADIOS allows to use 2 plugins at once for each I/O operation

Getting that provenance

- Adios-provenance is a plugin to be used as secondary I/O method
  - It gathers only the metadata about the data (small) on one of the processes
  - It can send the metadata over a socket
- Metadata
  - variable names, types, dimensions
  - attributes (with values or reference to variables)
  - characteristics automatically calculated by ADIOS for each variable, currently
    - min/max of an array (per processor)
    - value of a scalar variable

Connecting the simulation and the workflow

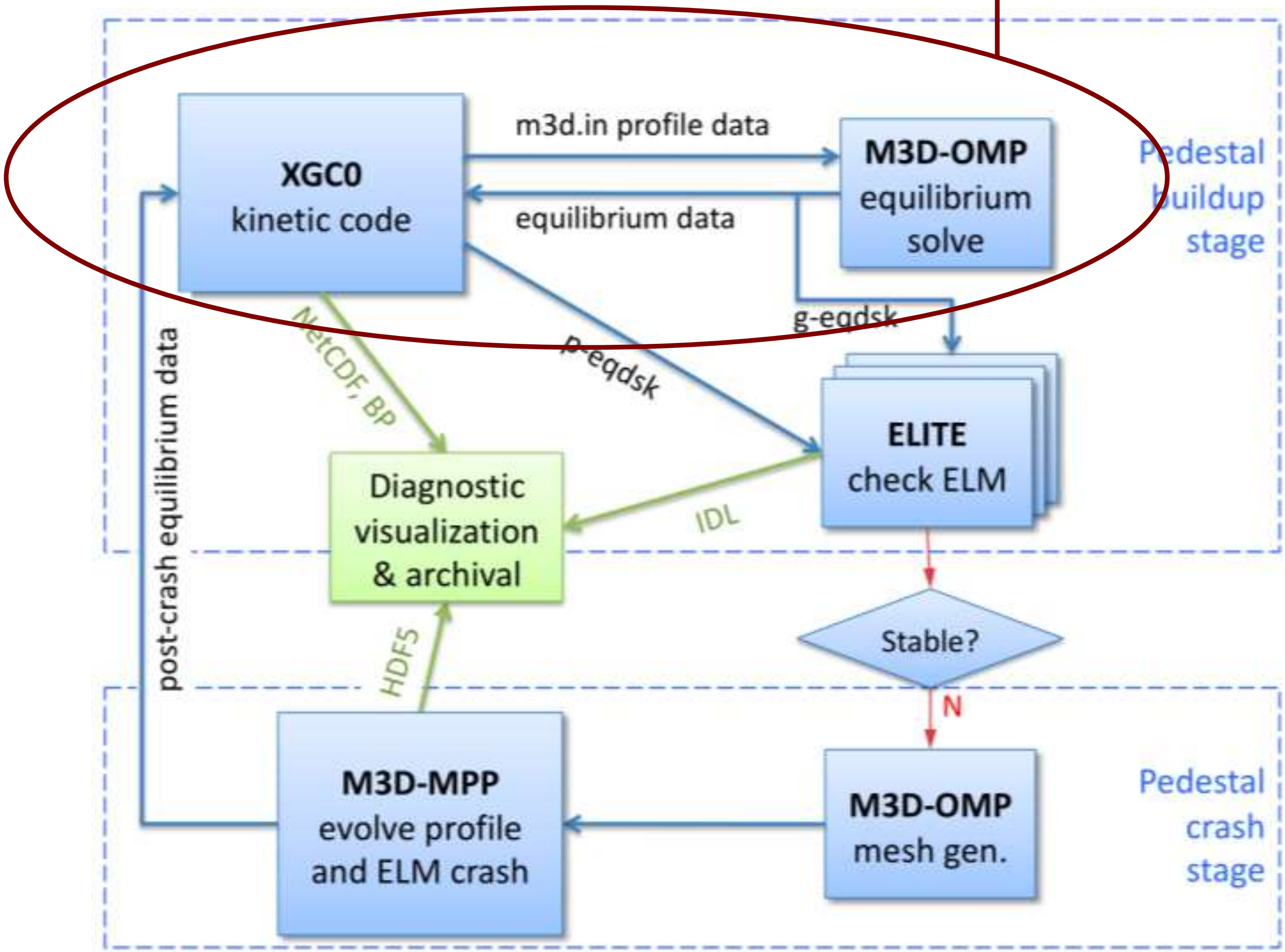
- Workflow has an SSH connection to the front-end
  - to look for data files and execute commands
- Simulation can connect to front-end



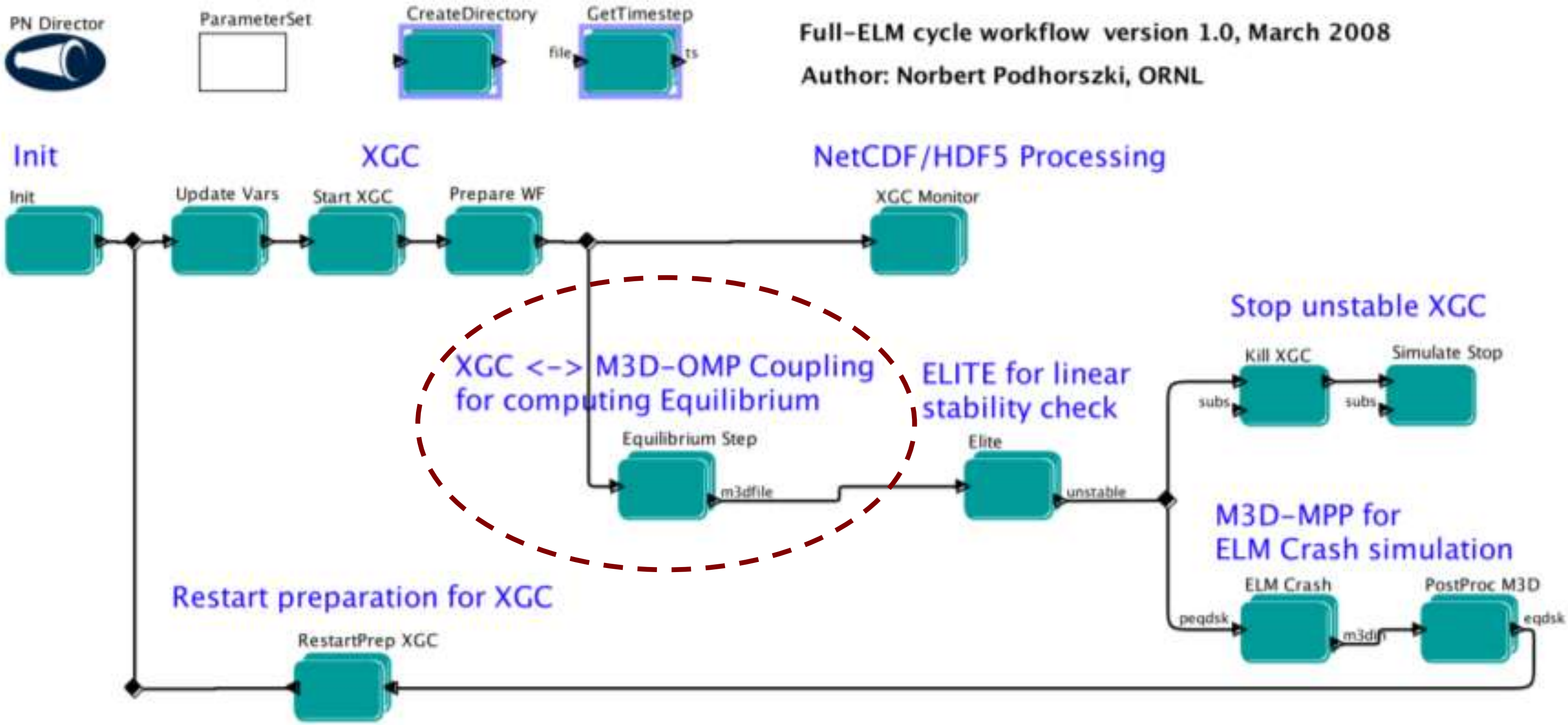
- Port forwarding in SSH daemon establishes the path to outside
- Two-way communication

Another coupling example

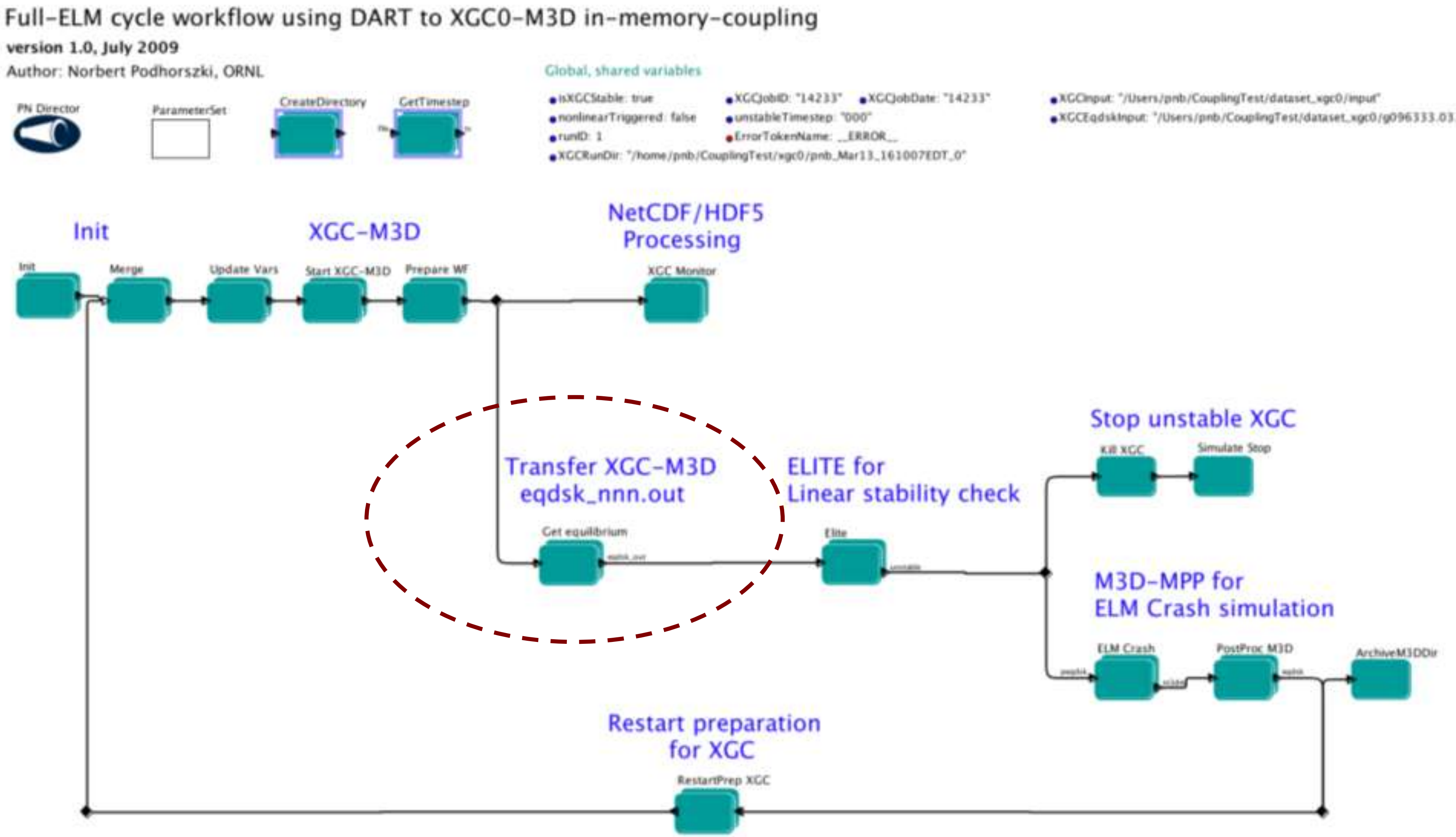
File-based or mem-to-mem



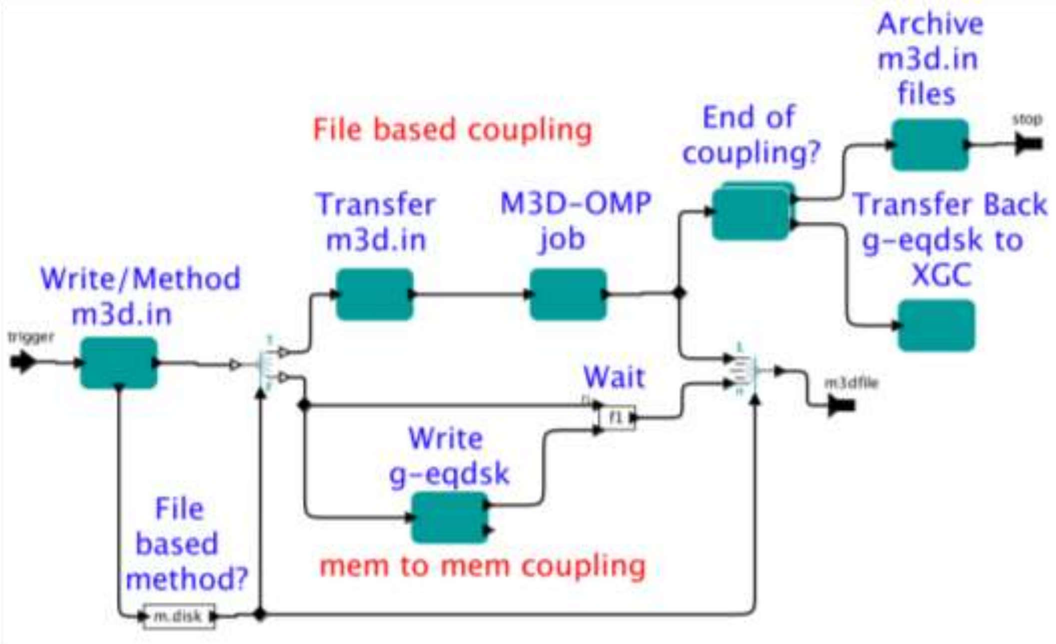
Coupling workflow (file-based)



Coupling workflow (memory-to-memory)



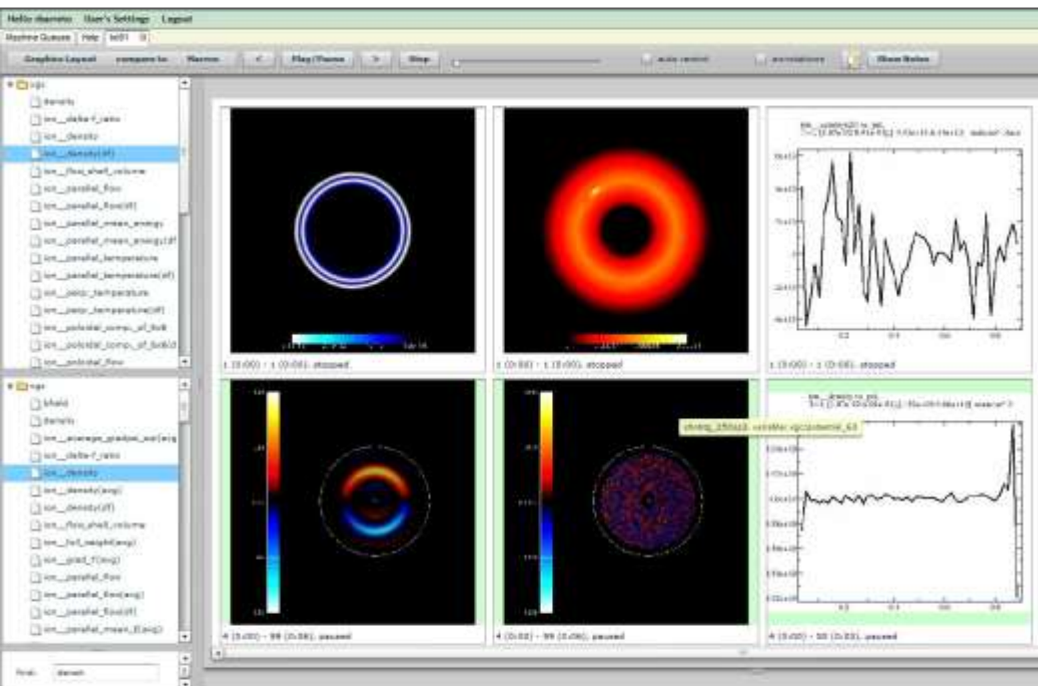
Single coupling workflow for both modes



- M3D-OMP step should handle both cases
  - get data file and run M3D-OMP job or
  - wait for the result data to appear
- Actor “Write/Method m3d.in”
  - fire once on trigger; output whenever m3d.in file is written
  - also tell which transport method was used
- Actor “Write g-eqdsk”
  - fire on trigger; output when g-eqdsk file is written

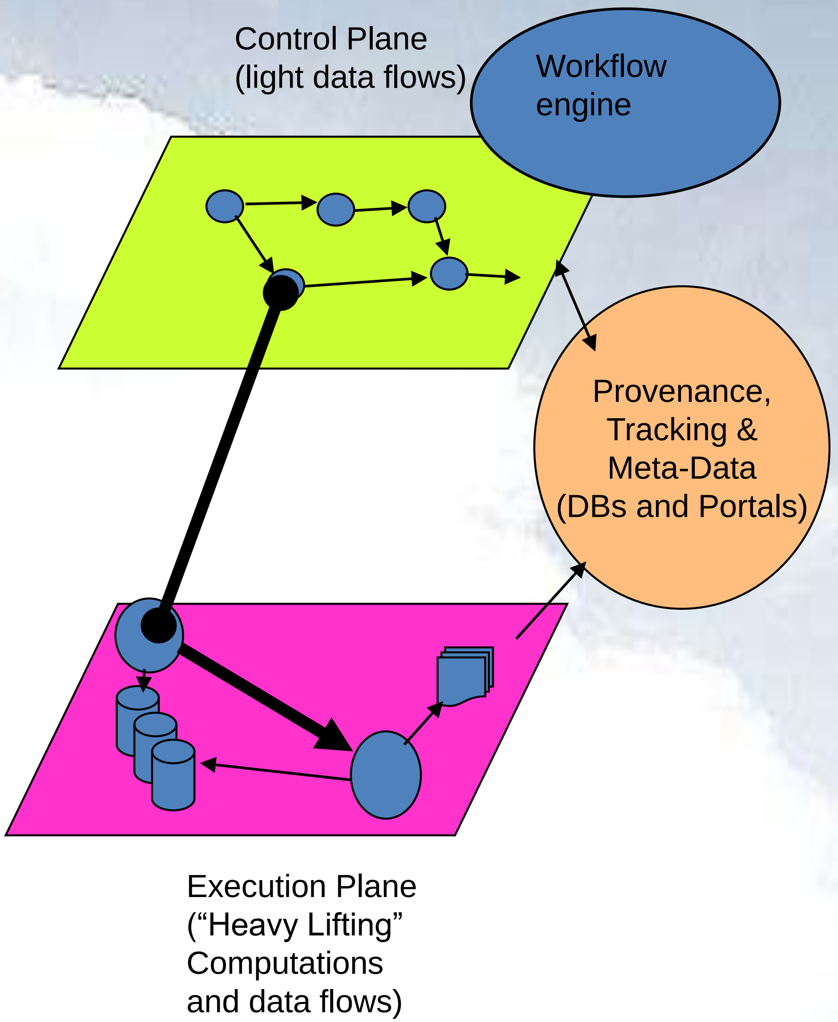
Other uses of the provenance

- Eliminate polling
  - "Write" event informs about a data file to be processed by the workflow
  - no need to regularly list sim. directory for data
- Make plots from exchanged data and record data lineage provenance for the produced images
  - later an analysis can get the data from the provenance database (as series of data values) instead of from a (non-existing) file



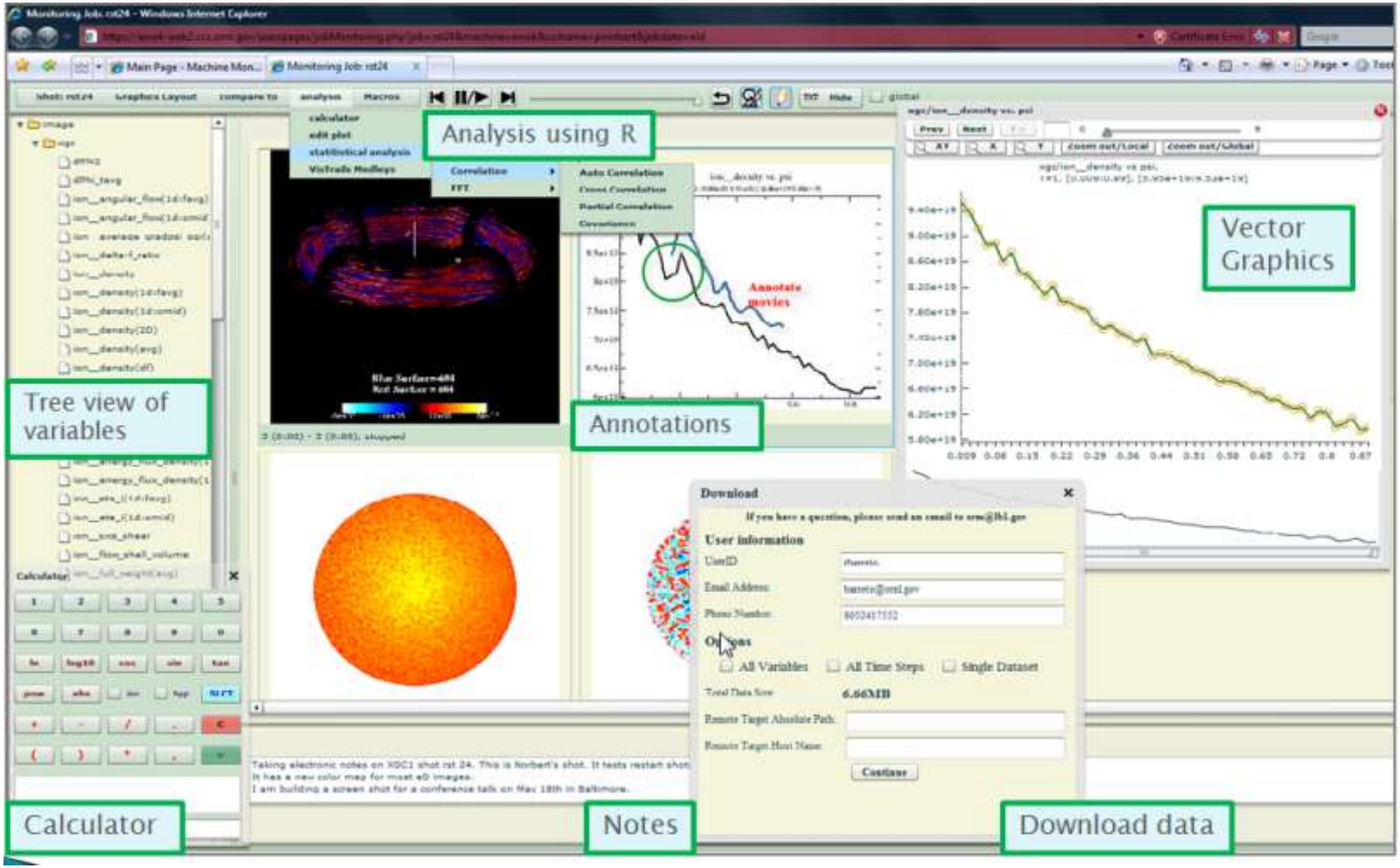
Essential ingredient to creating a data analysis facility

- **Process provenance**
  - **the steps performed in the workflow, the progress through the workflow control flow, etc.**
- **Data provenance**
  - **history and lineage of each data item associated with the actual simulation (inputs, outputs, intermediate states, etc.)**
- **Workflow provenance**
  - **history of the workflow evolution and structure**
- **System provenance**
  - **Machine and environment information**
  - **compilation history of the codes**
  - **information about the libraries**
  - **source code**
  - **run-time environment settings**



Tracking Files in the Kepler Provenance Framework(Citations: 1)
Pierre Mouallem, Roselyne Barreto, Scott Klasky, Norbert Podhorszki, Mladen A. Vouk
Conference: Statistical and Scientific Database Management - SSDBM2009

eSimMon dashboard for collaborative data management, analysis, and visualization



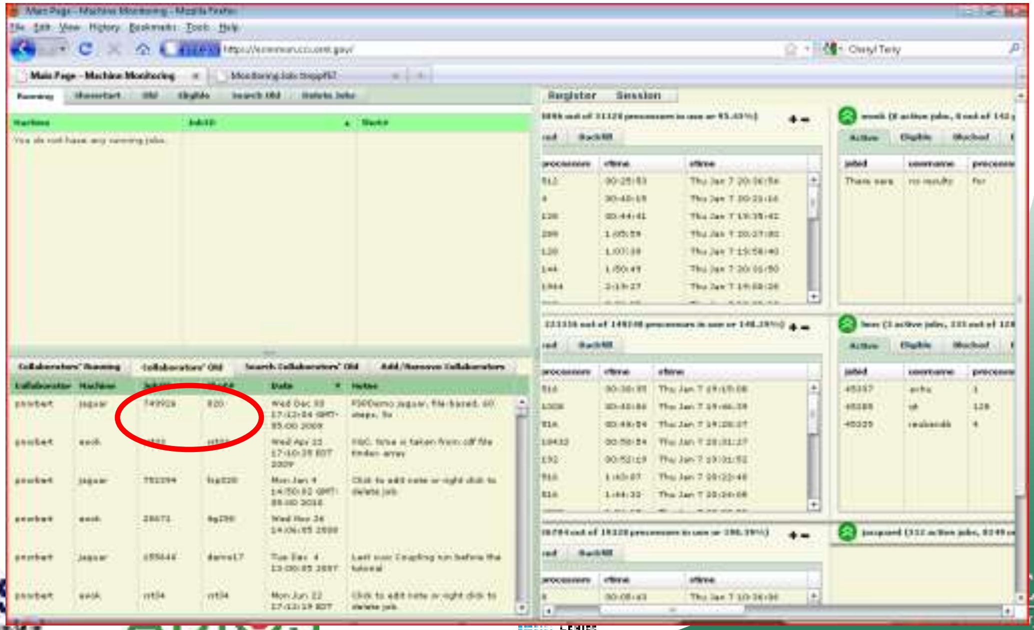
eSimMon 1.0 will be released this year.

eSimMon

- Goal
  - Abstract post processing services (Analysis with IDL, MatLab, Visit, Paraview, R) away from the interface.
- The **eSimMon** dashboard allows scientists with different backgrounds and levels of expertise to work together using one single online tool for **analysis**, **visualization**, and **data movement**
- Uses data, web, and workflow **service infrastructure** for flexibility and portability
- Keeping track of the **provenance information** (complete data lineage) is key for ease of use and efficiency.
  - It raises the focus from low IT details directly to the science by presenting researchers with simulation variables instead of files and directories.

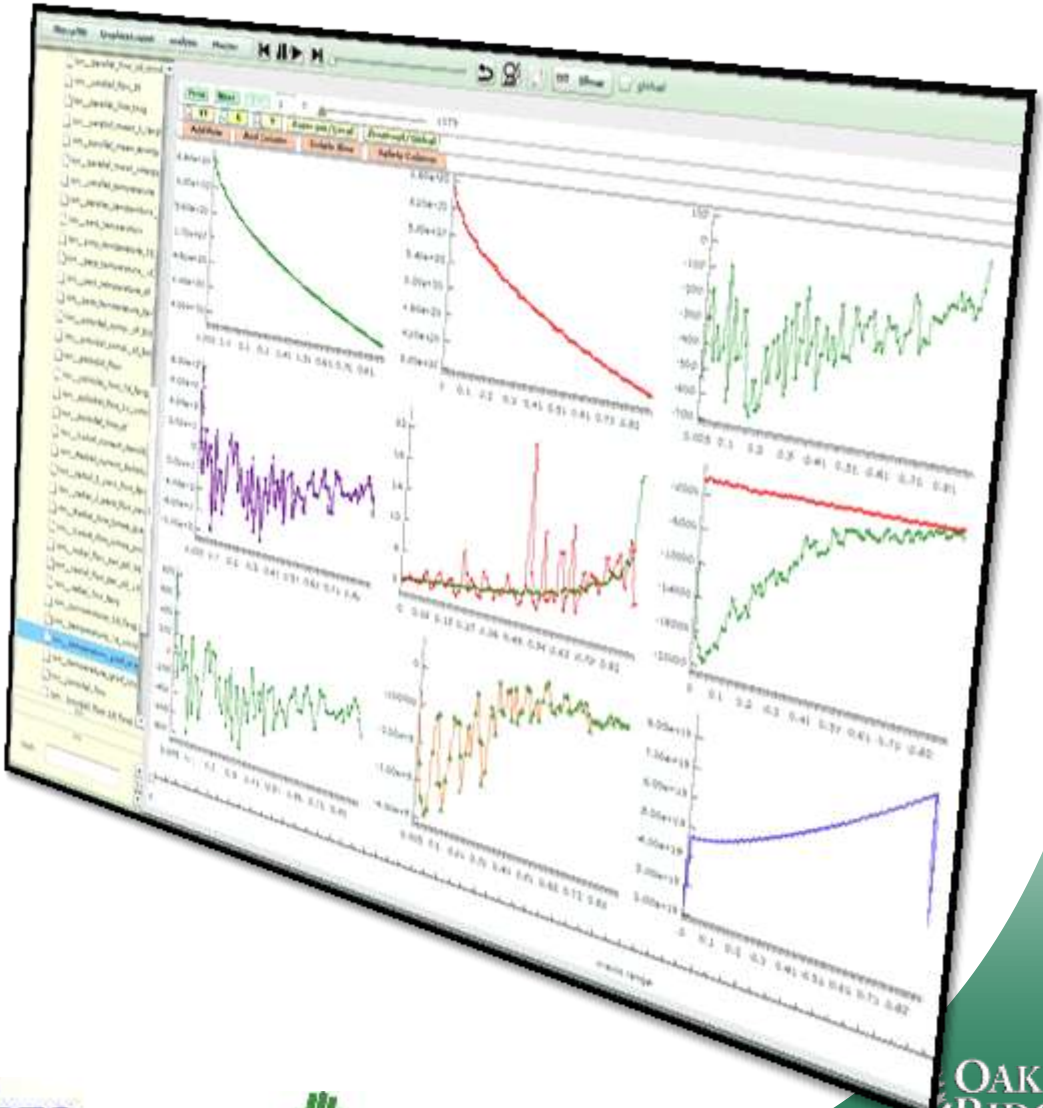
eSimMon Technology

- **Client Technology**: **Flash** which is a popular choice for responsive and event-driven Rich Internet Applications (RIAs)
- **Server Technology:** PHP/MySQL. The back-end creates and accesses a "data store" that contains user preferences and activities and information stored by the workflow during the simulation monitoring



Analysis

- We have integrated the following analysis
  - Vector graphics
  - Calculator
  - 3D module
  - Matlab
  - R
- We are looking at built-in tools as well as external plugin tools



by UT-Battelle
partment of Energy

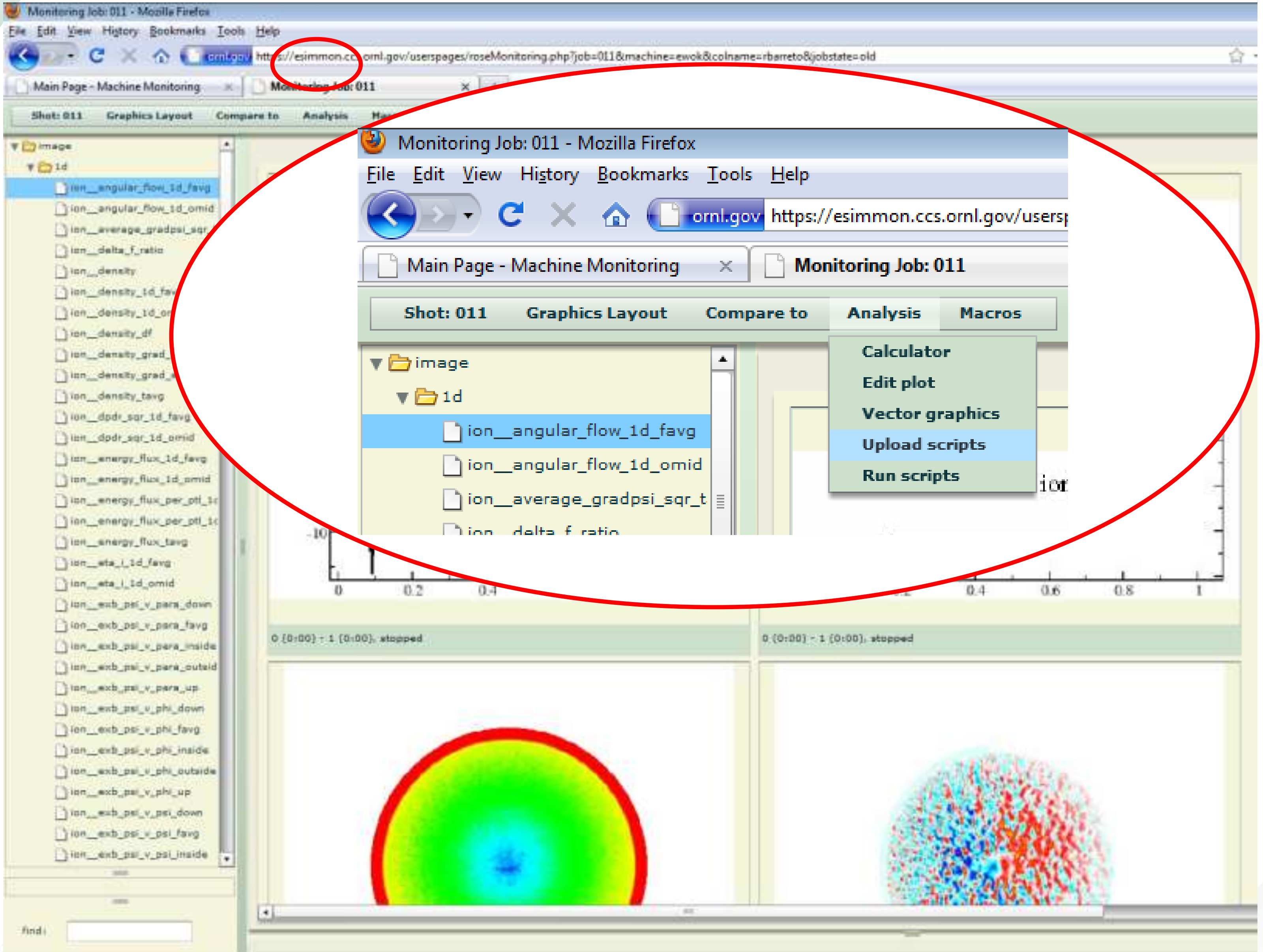
Matlab Jobs

- Developed GUIs to upload and run Matlab scripts from the dashboard
- Tested this GUIs with the GSI infrastructure at ORNL and Matlab scripts from Seung-Hoe Ku
- Next step are:
  - Explain to users how to use the GUI
  - Get their feedback for next implementation
- In the section we address the following questions:
  - What users need to do to use the GUI?
  - What are the modifications to their scripts?
  - What are the assumptions made by the dashboard?

Matlab Jobs

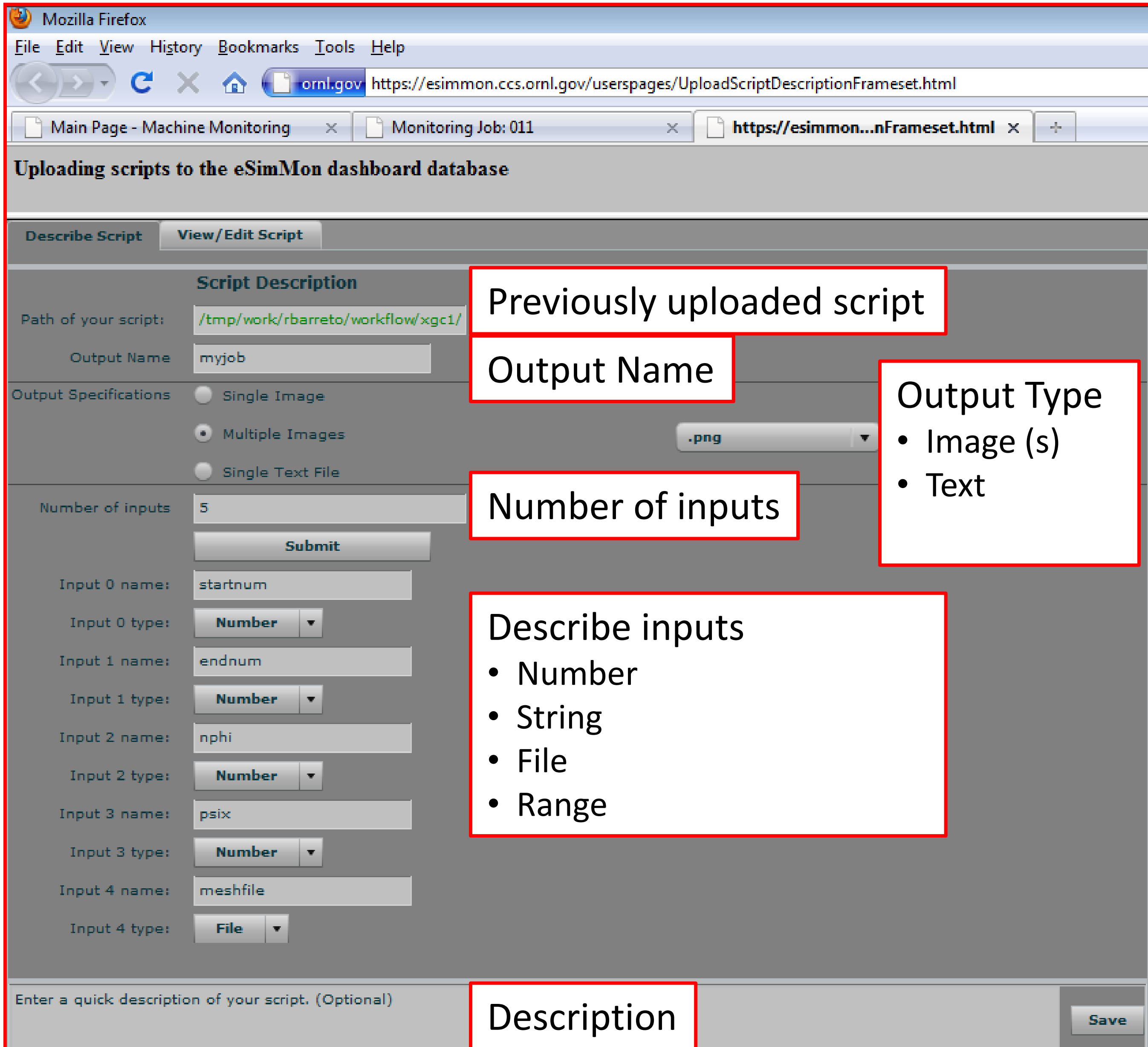
- Get a DOE grid certificate at: https://pki1.doegrids.org:443
- Execute the initializing command on any machine that supports GSI certificates
  - Run: ***myproxy-init –n*** to store a credential on the myproxy server
  - Enter your pass phrase ➔ credential is valid for 7 days
- Log on to the dashboard

Matlab Jobs


Monitoring Job: 011 - Mozilla Firefox
File Edit View History Bookmarks Tools Help
ornl.gov
https://esimmon.ccs.ornl.gov/usersp
Main Page - Machine Monitoring
Monitoring Job: 011
Shot: 011
Graphics Layout
Compare to
Analysis
Macros
Calculator
Edit plot
Vector graphics
Upload scripts
Run scripts
image
1d
ion__angular_flow_1d_favg
ion__angular_flow_1d_omid
ion__average_gradpsi_sqr_t
0 (0:00) - 1 (0:00), stopped
0 (0:00) - 1 (0:00), stopped
find:

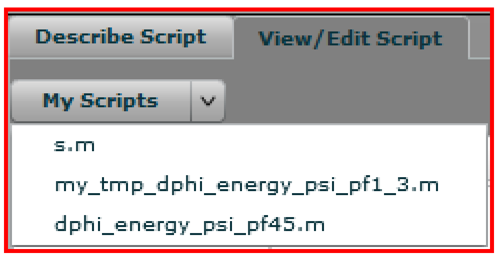
Matlab Jobs

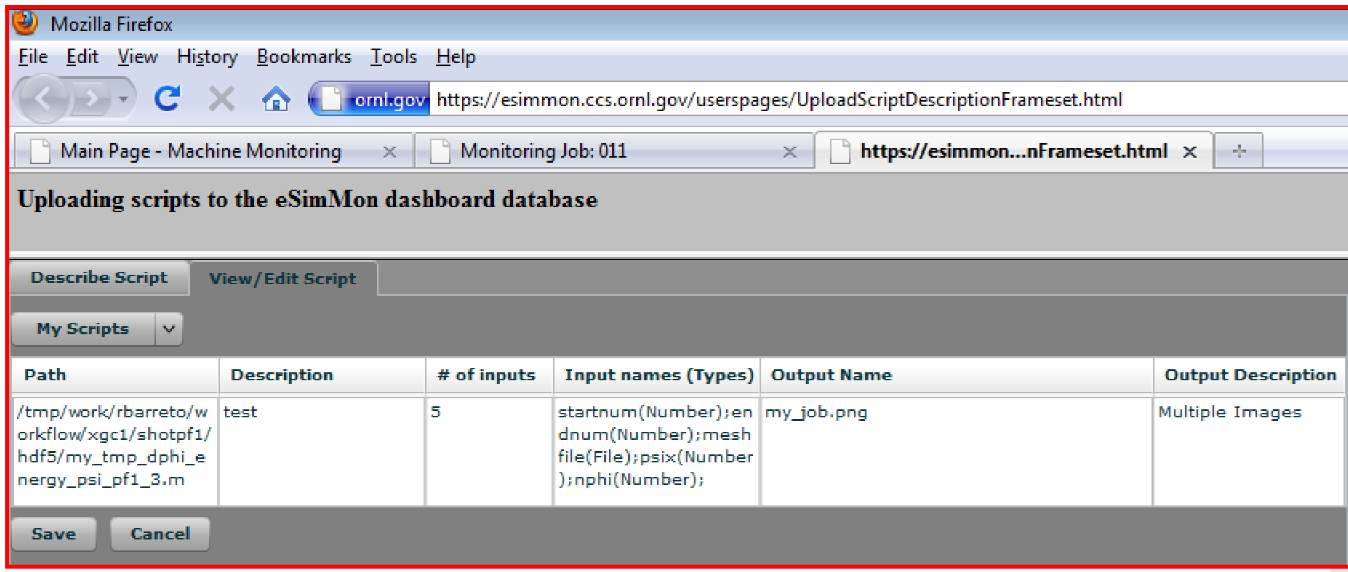
Describe the script to the dashboard



Matlab Jobs

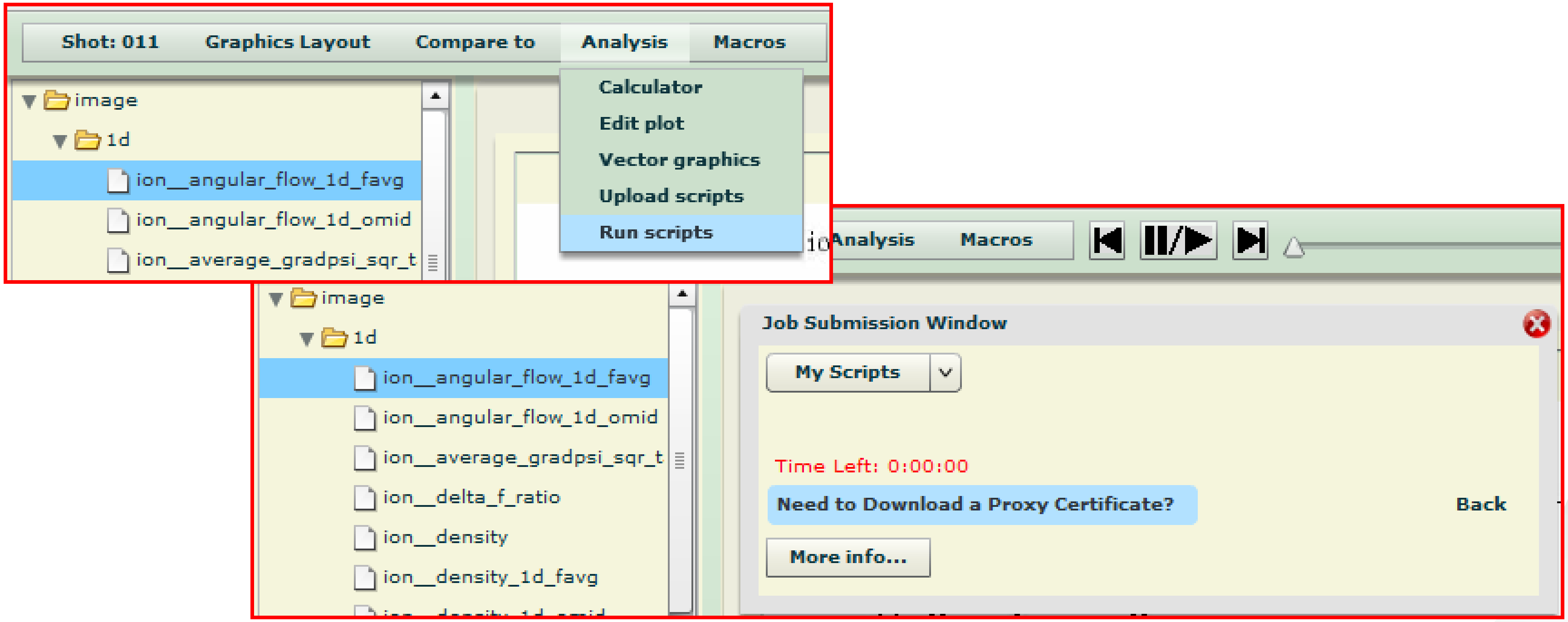
View and edit uploaded scripts





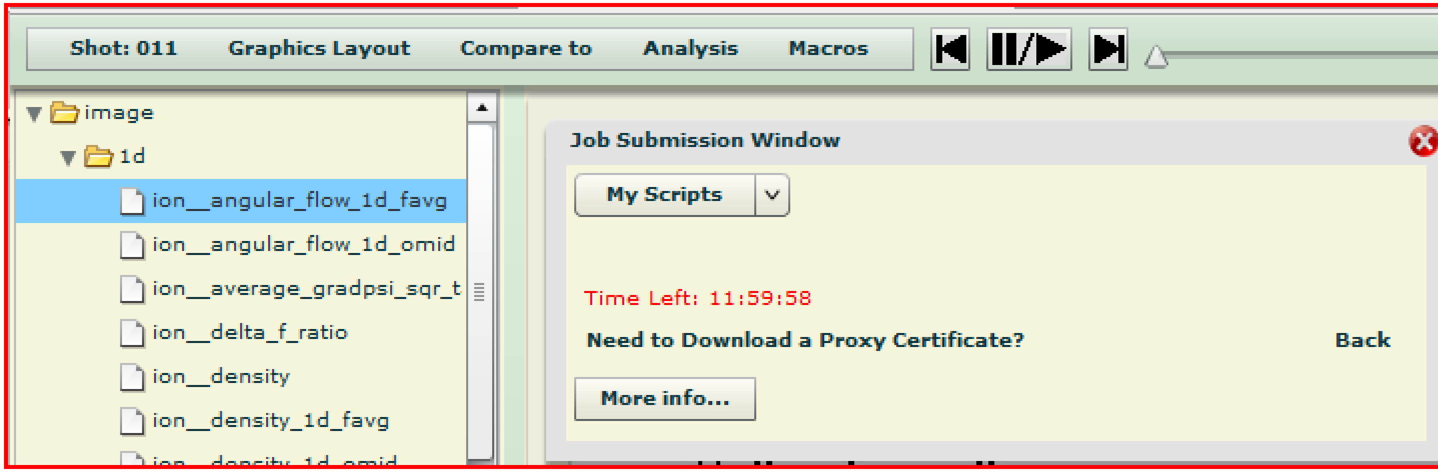
Matlab Jobs

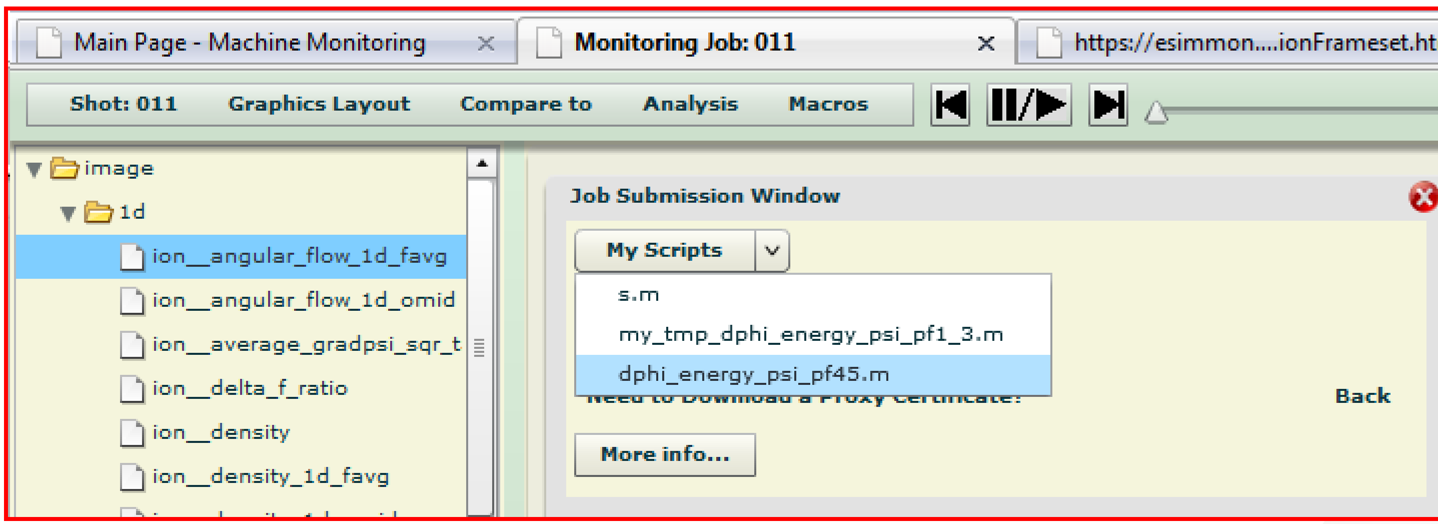
- Obtain your certificate from the server through the dashboard. The GUI run ***myproxy-logon*** using your one time password.



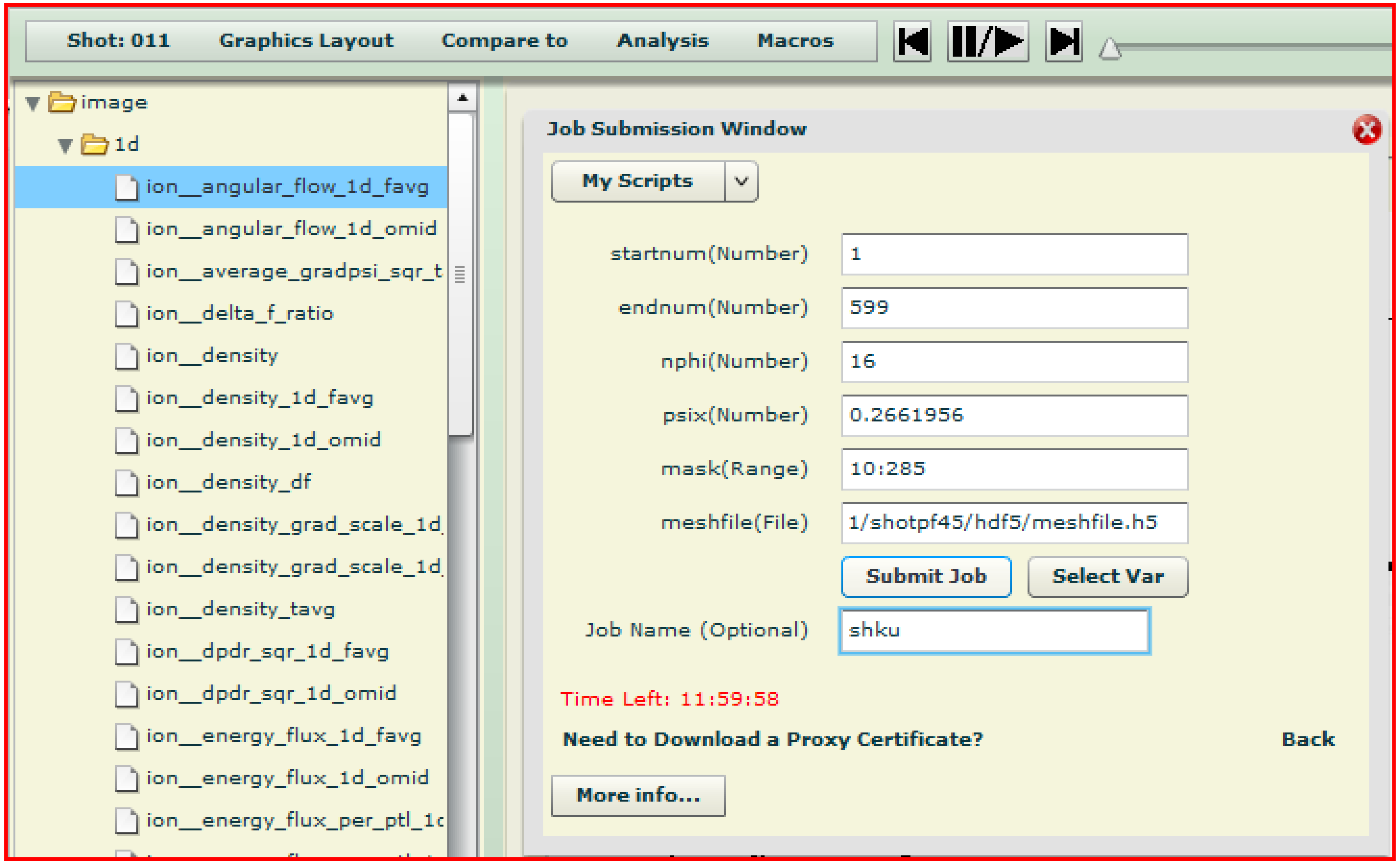
Matlab Jobs

- The certificate is valid 12 hours. You will see the time remaining on the GUI
- Select a script to run from a list



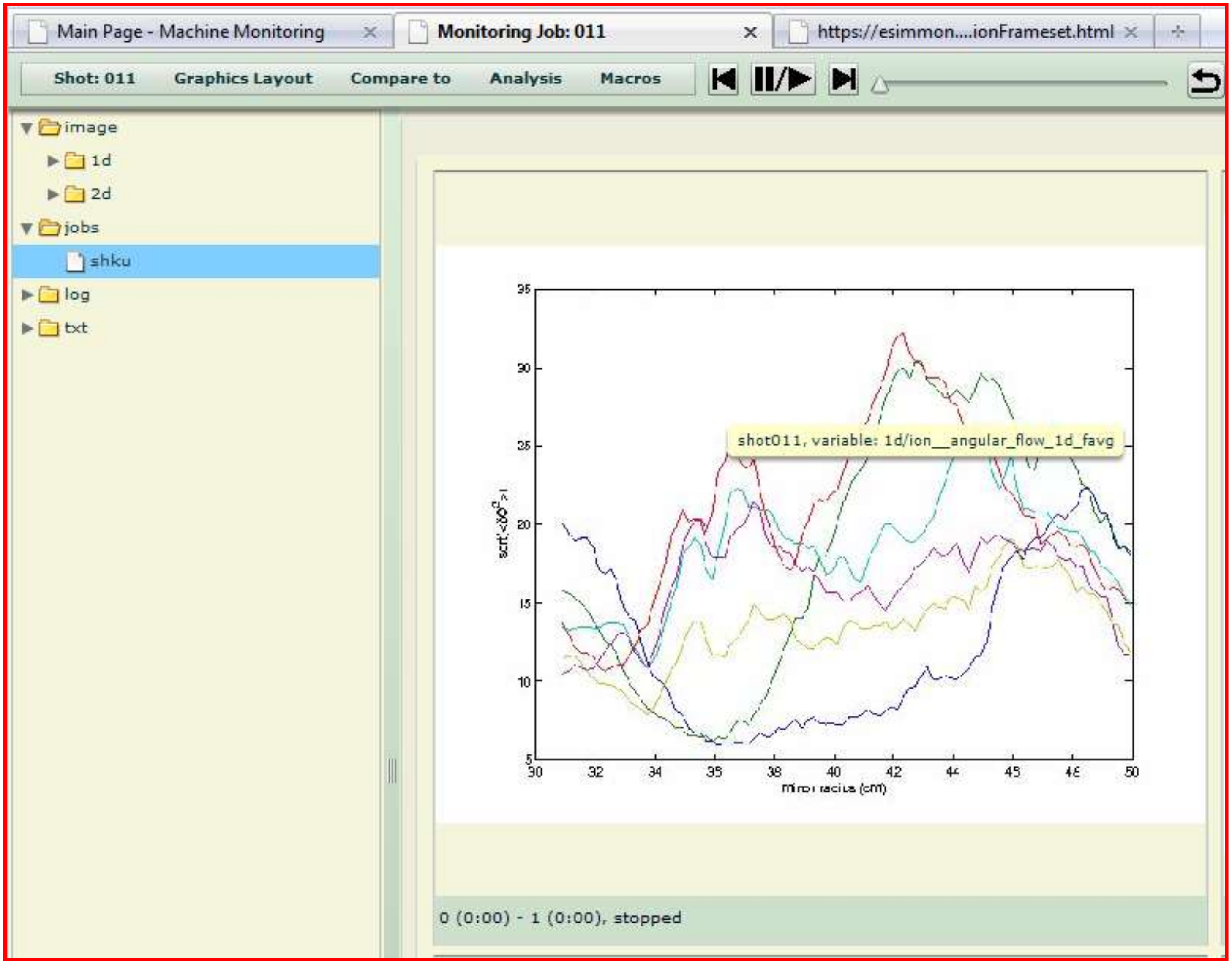


Matlab Jobs



Enter values for each parameter and submit the job.

Matlab Jobs



Get the results

Matlab Jobs

Advantages:

- The script can be ran several times with different parameters
- The description and inputs used are recorded and accessible from the dashboard
- File inputs do not need to be input. Users can use the built-in provenance in the dashboard to run the same analysis on different shots
- We plan to allow users to share scripts and results

Summary and Conclusions

- **Unprecedented opportunities for dramatic insights through computation!**
- 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
- Overarching philosophy
  - Abstraction & Separation through SOA
    - Allows independent development and execution of physics services.
    - Separates computations from composition and coordination; Interface from implementations.
  - Existing and proven concepts - widely accepted/used by the enterprise computing community
- EFFIS Innovations
  - Reducing barriers from a scientists perspective
    - Ease-of-use, simple code integration and maintainability
  - Minimizing performance impact
  - Addressing unique requirements of FSP specifically and scientific computing in general