In-situ analysis with ADIOS

Scott Klasky, Norbert Podhorszki, Ana Gainaru

NNSA National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

Outline

Part I: Introduction to Parallel I/O and HPC file systems (0.5 hours) (Beginner/Intermediate)

- **Lecture:** Parallel I/O
- **Lecture:** HPC Storage Systems – GPFS, Lustre, Burst Buffers

Part II: Self-describing I/O using ADIOS (1 hour) (Beginner/Intermediate)

- **Lecture:** ADIOS framework, I/O abstraction, file format
- **Hands-on**: use a parallel MiniApp to write self-describing data
  - Use ADIOS write API to write data in parallel
  - Write HDF5 files using the ADIOS API
- **Hands-on**: Parallel data reading
  - ADIOS read API in Fortran90, C++, and Python
  - Read HDF5 files using the ADIOS API
- **Lecture**: How to scale ADIOS I/O

BREAK

Part III: Data Compression (0.5 hour) (Intermediate)

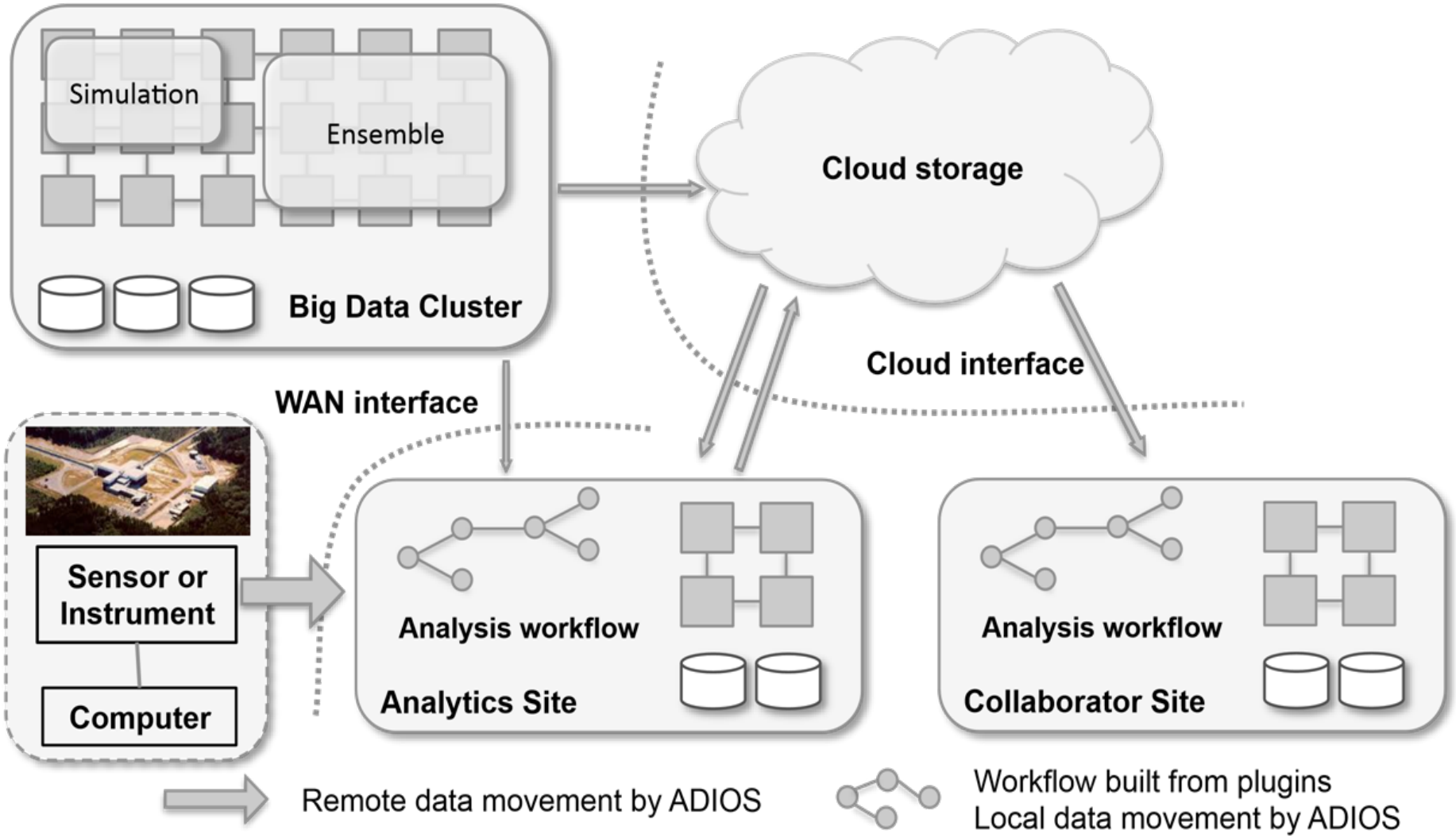
- **Lecture**: Overview of common data reduction techniques for scientific data
  - Introduction to compression
  - Introduction to lossy compression techniques: MGARD, SZ, and ZFP
- **Hands-on**: Adding compression to previous examples

Part IV: In situ data analysis using I/O staging (1 hour) (Intermediate/Advanced)

- **Lecture**: Introduction to “data staging” for in situ analysis and code coupling
- **Hands-on:** Create a simple pipeline using the MiniApp that computes, and visualizes a derived variable using data staging
- **Hands-on**: Add data reduction to the pipeline
- **Demonstration**: In situ visualization with Visit and Paraview
- **Hands-on**: Staging and converting with adios_reorganize tool

Wrap-up

Vision: building scientific collaborative applications

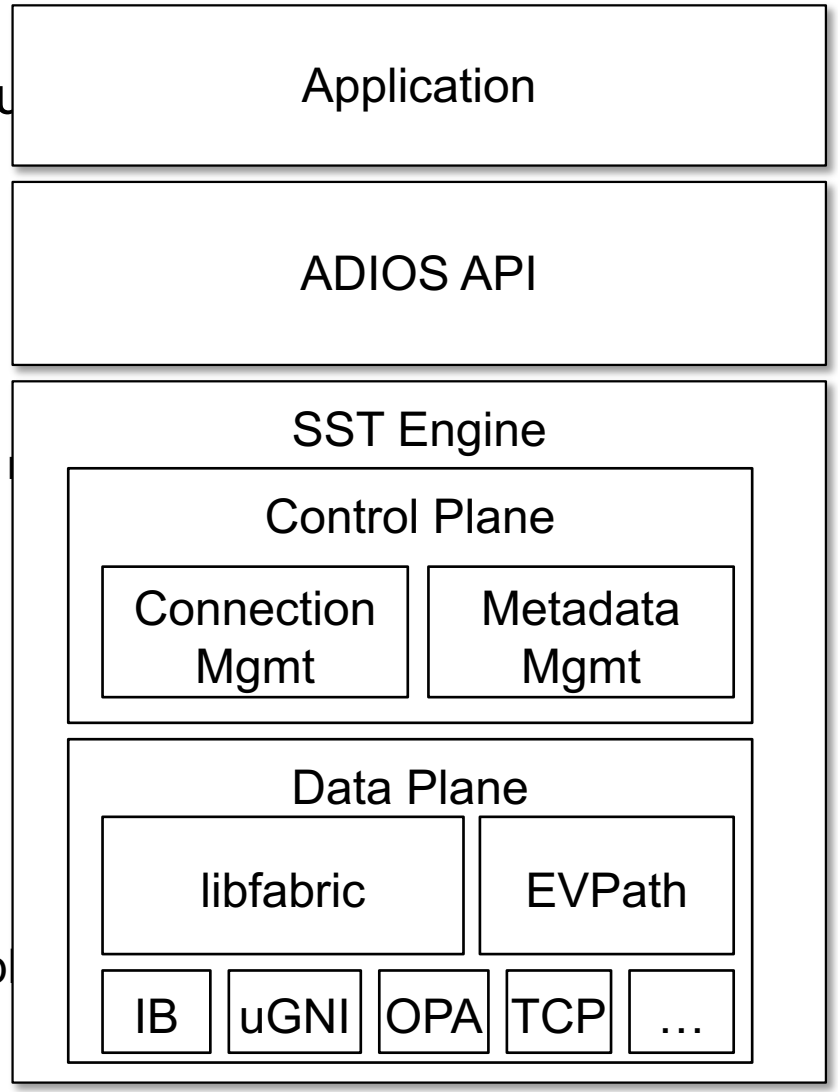


Data Staging in ADIOS

- Sustainable Staging Transport (SST)
  - In-situ infrastructure for staging in a streaming-like fashion using RDMA, SOCKETS
- DataMan
  - WAN transfers using sockets and ZeroMQ
- SSC
  - One-sided MPI for strong coupling of codes
- DataSpaces
  - Staging infrastructure providing a shared memory abstraction

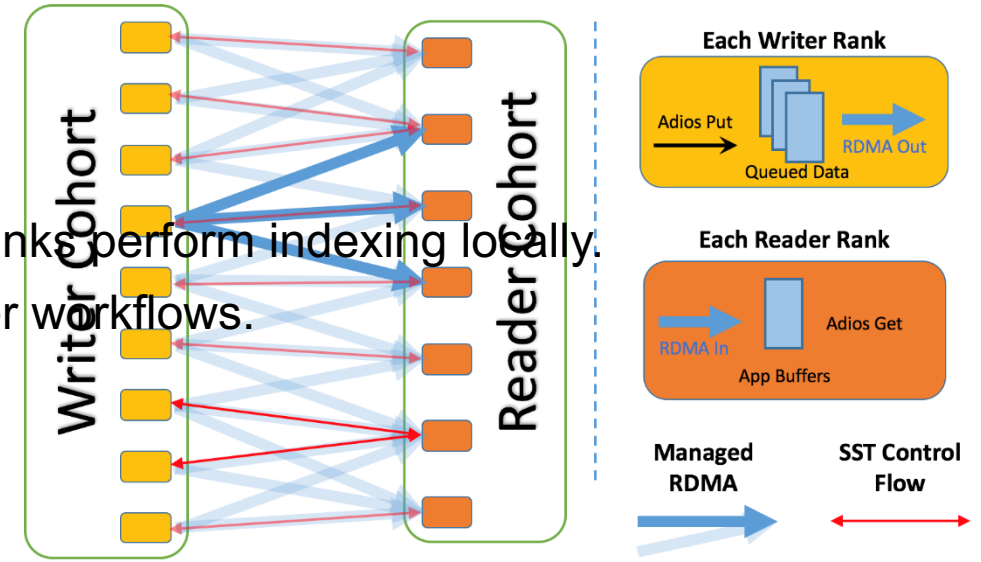
Sustainable Staging Transport (SST)

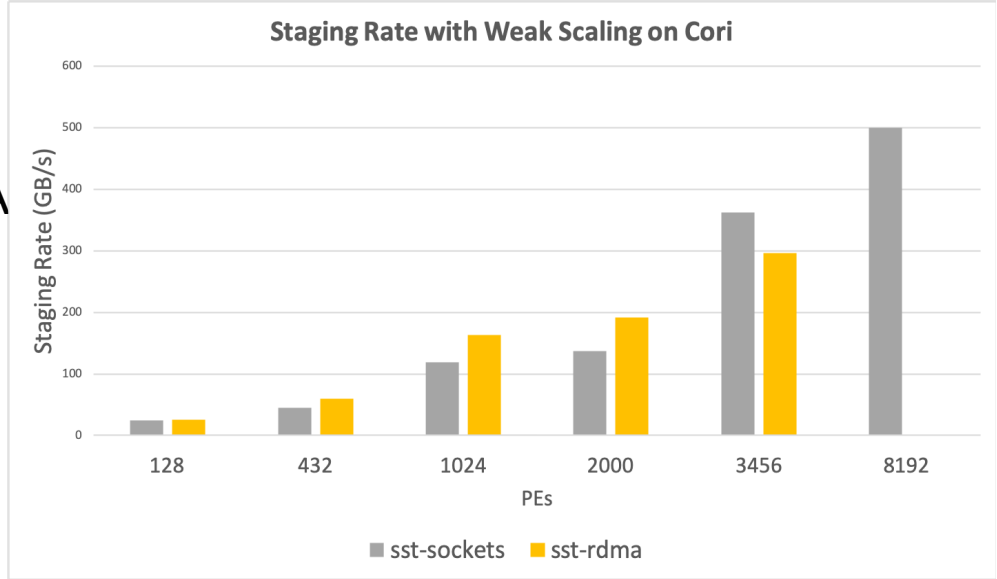
- Direct coupling between data producers and consumers for in-situ
- Designed for portability and reliability.
- Control Plane
  - Manages meta-data and control using a message-oriented protocol
  - Inherits concepts from Flexpath, DIMES, uses EVPath
  - Allows for dynamic connections, multiple readers and complex flow control
- Data Plane
  - Exchange data using RDMA
  - Responsible for resource management for data transfer
  - Uses libfabric for portable RDMA support
  - Threaded to overlap communication with computation and for asynchronous progress monitoring
  - Modular interface with the control plane supports alternative data p



Sustainable Staging Transport (SST)

- Data is staged in writer ranks' memory.
  - Metadata is propagated to subscribed readers, reader ranks perform indexing locally.
  - Metadata structure is optimized for single writer, N reader workflows.
  - Supports late joining/early leaving readers.
- Modular design
  - Well-encapsulated interface between DP and CP.
  - Multiple inter-changeable DP implementations.
- RDMA for asynchronous transfer
  - Currently tested and performant for GNI, IB (verbs), OPA





SSC (Staging for Strong Coupling)

- An ADIOS 2 engine using MPI for portability and performance

- Features
  - Use a combination of one-sided MPI and two-sided MPI methods for flexibility and performance.
  - Use threads and non-blocking MPI methods to hide communication time.
  - Optimized for fixed I/O pattern – push data to consumer

- Target Applications
  - XGC, Gene, and Gem three-way coupling.
  - Other ECP applications requiring strong coupling or always-on in-situ analysis/visualization

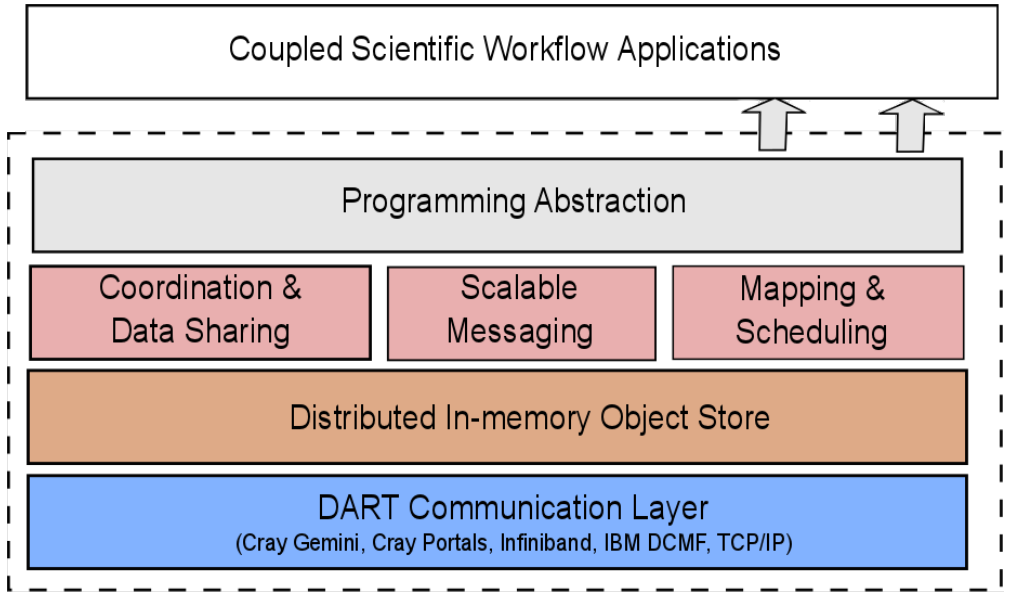
DataMan

- An ADIOS 2 engine subroutine designed for wide area network data transfer, data staging, near-real-time data reduction and remote in-situ processing
- Designed with following principles:
  - Flexibility: allowing transport layer switching (ADIOS BP file, ZeroMQ, google-rpc etc.)
  - Versatility: supporting various workflow modes (p2p, query, pub/sub, etc.)
  - Adaptability: allowing adaptive data compression based on near-real-time decision making
  - Extendibility: taking advantage of all ADIOS 2 transports and operators, and other potential third-party libraries. For example, DataMan can use ZFP, Bzip, SZ that have been built into ADIOS2, as well as any compression techniques that will be built into ADIOS 2 in future.
- Target Applications:
  - ECP applications requiring wide area network data transfer and adaptive data reduction
  - Square Kilometer Array observational data
  - KSTAR fusion experimental data

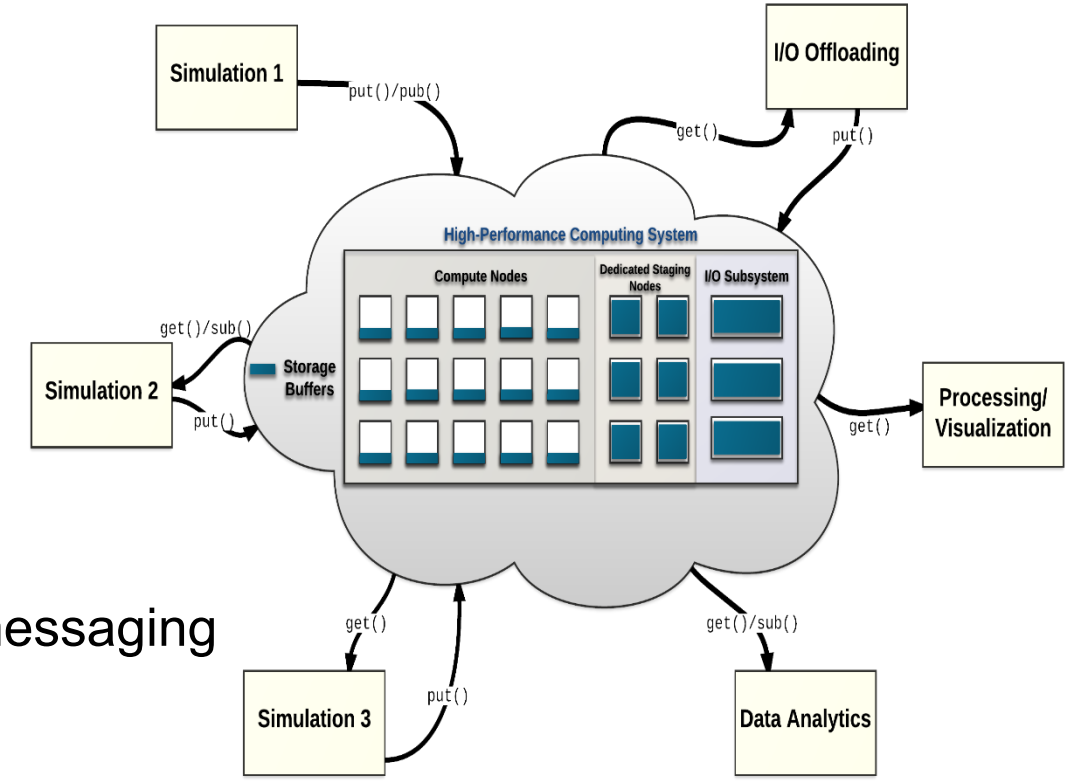
DataMan: Supports three workflow modes

- Push
  - Sender driven workflow
  - Senders are configured to send data to pre-defined receivers.
- Query
  - Receiver driven workflow
  - Receivers are configured to query particular data (particular subsets of particular variables) from senders.
- Subscribe
  - Ad-hoc workflow
  - Free combination of senders (publishers) and receivers (subscribers).
  - Senders and receivers can be launched in any order, and they can be attached and detached freely without affecting other senders or receivers.

DataSpaces: Extreme Scale Data Staging Service



The DataSpaces Abstraction



- Virtual shared-space programming abstraction
  - Simple API for coordination, interaction and messaging
- Distributed, associative, in-memory object store
  - Online data indexing, flexible querying
- Adaptive cross-layer runtime management
  - Hybrid in-situ/in-transit execution
- Efficient, high-throughput/low-latency asynchronous data transport

Inline engine: in-process in situ visualization

- An ADIOS 2 engine executing consumer inside producer's EndStep()

- Features
  - Consumer has zero-copy access to the local data block of the producer's data in memory
  - Synchronous execution of consumer code, during producer's EndStep() call

- Target Applications
  - Traditional in situ visualization routines that scale well with the producer's size

Application: Which staging method to use?

- System considerations
  - Staging between machines/over a WAN? **DataMan**
  - Co-located execution? **SST-SM+X**
  - Availability of RDMA high-speed network? **SST** (and **DataSpaces**) optimized for this case.
- Coupling considerations
  - Highly synchronized writer/reader and a highly optimized MPI? **InsituMPI**
  - Ensemble workflow / multiple independent writers / data lives independently of any given writer component? **DataSpaces**
  - 1 writer, many readers streaming? **SST** optimized for this use case.
- Performance considerations
  - Often application-specific, and not always obvious.
  - Here's where it helps to have a flexible I/O framework…

Staging I/O

Processing data on the fly by

1. Reading from file concurrently, or

2. Moving data without using the file system

Design choices for reading API

- One output step at a time
  - **One step is seen at once after writer completes a whole output step**
  - streaming is not byte streaming here
  - reader has access to all data in one output step
  - as long as the reader does not release the step, it can read it
    - potentially blocking the writer
- Advancing in the stream means
  - get access to another output step of the writer,
  - while losing the access to the current step forever.

ADIOS concepts for the read API (get)

- Step
  - A dataset written within one adios_begin_step/…/adios_end_step
- Stream
  - A file containing of series of steps of the same dataset
- Open for reading as a stream
  - for step-by-step reading (both staged data and files)

```
adios2::Engine reader = io.Open(filename, adios2::Mode::Read, comm);
```

- Close once at the very end of streaming

```
reader.Close();
```

Advancing a stream

- One step is accessible in streams, advancing is only forward

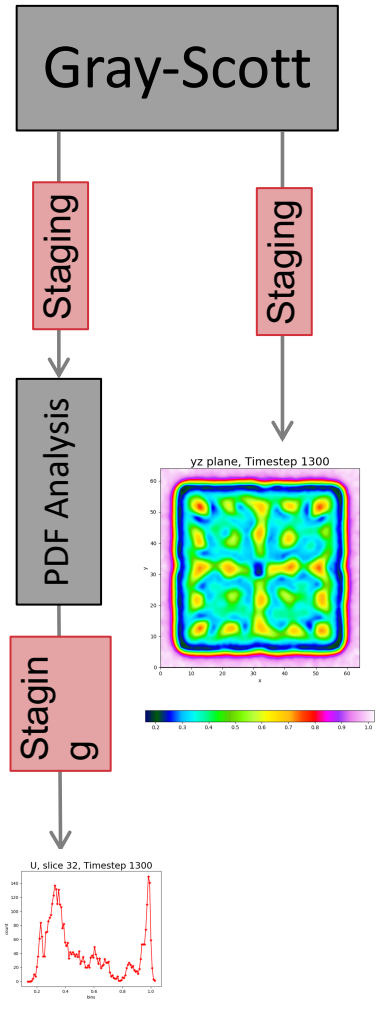
```
adios2::StepStatus read_status =
        reader.BeginStep(adios2::StepMode::Read, timeout);
if (read_status != adios2::StepStatus::OK) {
    break;
}
```

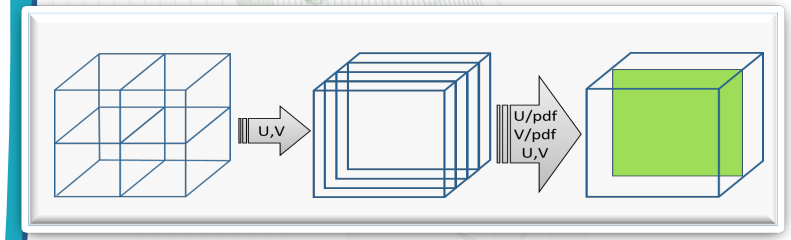
  - float timeout: wait for this long for a new step to arrive
- Release a step if not needed anymore
  - optimization to allow the staging method to deliver new steps if available

```
reader.EndStep();
```

Gray-Scott example

with staging





File-based in situ processing

The runtime config file: adios2.xml

```
<?xml version="1.0"?>
<adios-config>

 <!--==================================
       Configuration for for Gray-Scott and GS Plot
     ==================================-->
  <io name="SimulationOutput">
    <engine type="BP4">
      <parameter key="OpenTimeoutSecs" value="10.0"/>
    </engine>
  </io>
```

Engine types

BP4

HDF5

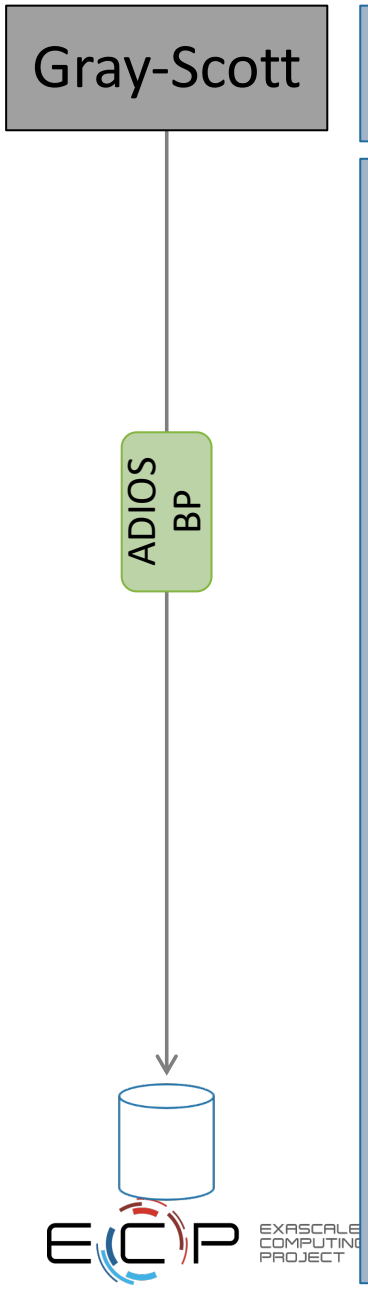
SST

InSituMPI

DataMan

```
  <!--==================================
       Configuration for PDF calc and PDF Plot
     ==================================-->

  <io name="PDFAnalysisOutput">
    <engine type="BP4">
      <parameter key="OpenTimeoutSecs" value="10.0"/>
    </engine>
  </io>

</adios-config>
```

In Situ Visualization with ADIOS

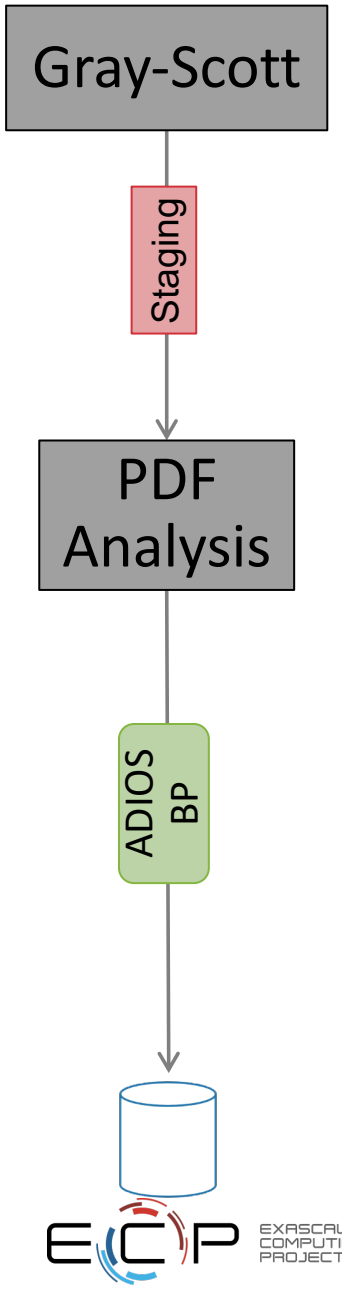


edit **adios2.xml: SimulationOutput, PDFAnalysisOutput** uses **BP4** engine

```
$ ./cleanup.sh data
```

```
$ mpirun -n 4 adios2-gray-scott settings-staging.json
Simulation writes data using engine type:              BP4
=========================================
grid:               64x64x64
steps:              60000
plotgap:            100
F:                  0.02
k:                  0.048
dt:                 1
Du:                 0.2
Dv:                 0.1
noise:              0.01
output:             gs.bp
adios_config:       adios2.xml
decomposition:      2x2x1
grid per process: 32x32x64
=========================================
Simulation at step 0 writing output step     0
Simulation at step 100 writing output step     1
```

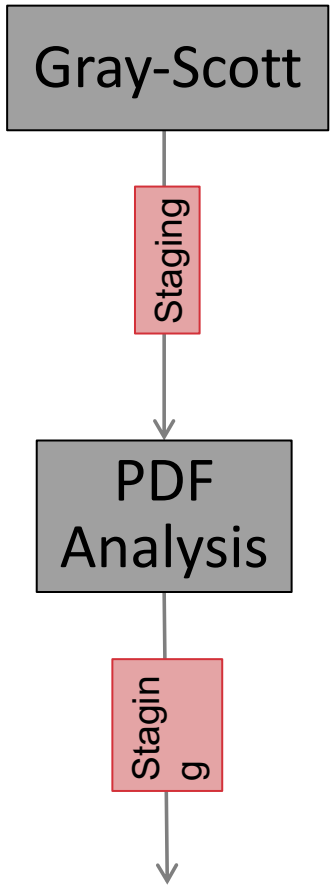
In Situ Visualization with ADIOS




```
$ mpirun -n 4 adios2-gray-scott settings-staging.json
 Simulation writes data using engine type:              BP4
=========================================
grid:               64x64x64
steps:              60000
 …
```

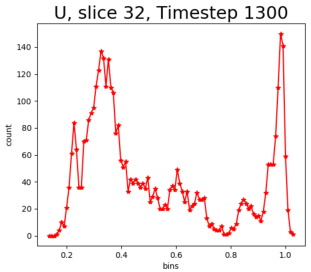
```
$ mpirun -n 2 adios2-pdf-calc gs.bp pdf.bp 100
PDF analysis reads from Simulation using engine type:  BP4
PDF analysis writes using engine type:                 BP4
PDF Analysis step 0 processing sim output step 0 sim compute step 0
PDF Analysis step 1 processing sim output step 1 sim compute step 100
PDF Analysis step 2 processing sim output step 2 sim compute step 200
PDF Analysis step 3 processing sim output step 3 sim compute step 300
PDF Analysis step 4 processing sim output step 4 sim compute step 400
PDF Analysis step 5 processing sim output step 5 sim compute step 500
PDF Analysis step 6 processing sim output step 6 sim compute step 600
...
```

In Situ Visualization with ADIOS



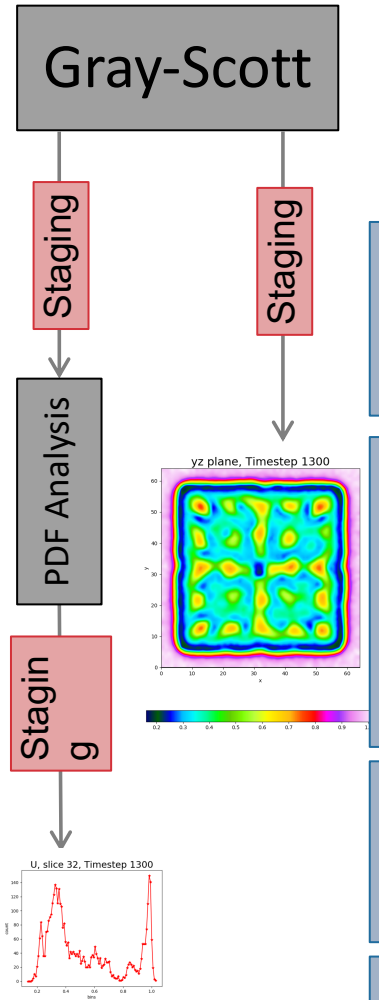

```
$ mpirun -n 4 adios2-gray-scott settings-staging.json
 Simulation writes data using engine type:              BP4
=========================================
grid:               64x64x64
steps:              60000
 …
```

```
$ mpirun -n 2 adios2-pdf-calc gs.bp pdf.bp 100
PDF analysis reads from Simulation using engine type:  BP4
PDF analysis writes using engine type:                 BP4
PDF Analysis step 0 processing sim output step 0 sim compute step 0
...
```




```
$ python3 pdfplot.py -i pdf.bp
PDF Plot step 0 processing analysis step 0 simulation step 0
PDF Plot step 1 processing analysis step 1 simulation step 100
PDF Plot step 2 processing analysis step 2 simulation step 200
PDF Plot step 3 processing analysis step 3 simulation step 300
...
```

In Situ Visualization with ADIOS




```
$ mpirun -n 4 adios2-gray-scott settings-staging.json
 Simulation writes data using engine type:              BP4
=========================================
```

```
$ mpirun -n 2 adios2-pdf-calc gs.bp pdf.bp 100
PDF analysis reads from Simulation using engine type:  BP4
PDF analysis writes using engine type:                 BP4
PDF Analysis step 0 processing sim output step 0 sim compute step 0
...
```

```
$ python3 pdfplot.py -i pdf.bp
PDF Plot step 0 processing analysis step 0 simulation step 0
...
```

```
$ python3 gsplot.py -i gs.bp
GS Plot step 0 processing simulation output step 0 or computation step 0
GS Plot step 1 processing simulation output step 1 or computation step 100
GS Plot step 2 processing simulation output step 2 or computation step 200
...
```

Streaming in situ processing with SST engine

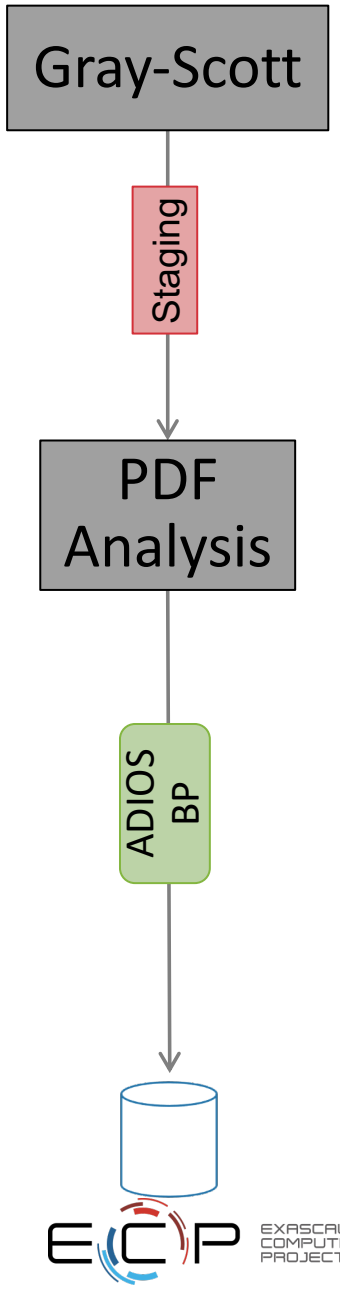
The runtime config file: adios2.xml

```
<?xml version="1.0"?>
<adios-config>

   <!--================================
        Configuration for the Simulation Output
       ================================-->
   <io name="SimulationOutput">
      <engine type="SST">
      </engine>
   </io>
```

Engine types
BP4
HDF5
SST
InSituMPI
DataMan

```
   <!--================================
        Configuration for the Analysis Output
       ================================-->

   <io name="AnalysisOutput">
      <engine type="SST">
      </engine>
   </io>

   <!--================================
        Configuration for the Visualization Input
       ================================-->
   <io name="VizInput">
      <engine type="SST">
      </engine>
   </io>

</adios-config>
```

In Situ Visualization with ADIOS

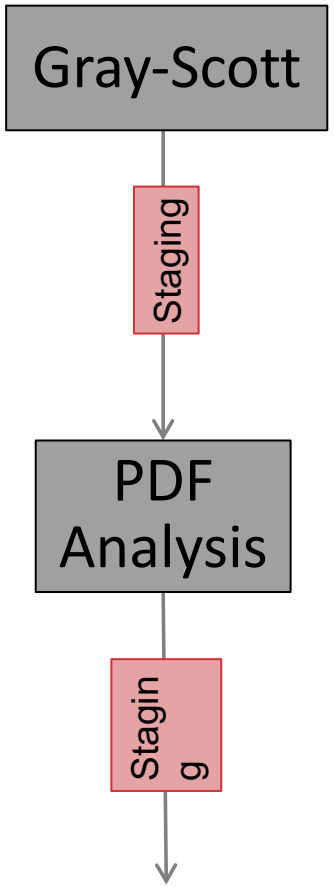


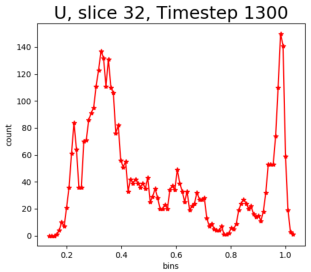

```
edit adios2.xml and change SimulationOutput to SST
                           PDFAnalysisOutput   to BP4
$ ./cleanup.sh data
```

```
$ mpirun -n 4 adios2-gray-scott settings-staging.json
 Simulation writes data using engine type:              SST
=========================================
grid:               64x64x64
steps:              60000
 …
```

```
$ mpirun -n 2 adios2-pdf-calc gs.bp pdf.bp 100
PDF analysis reads from Simulation using engine type:  SST
PDF analysis writes using engine type:                 BP4
PDF Analysis step 0 processing sim output step 0 sim compute step 0
PDF Analysis step 1 processing sim output step 1 sim compute step 100
PDF Analysis step 2 processing sim output step 2 sim compute step 200
PDF Analysis step 3 processing sim output step 3 sim compute step 300
PDF Analysis step 4 processing sim output step 4 sim compute step 400
PDF Analysis step 5 processing sim output step 5 sim compute step 500
PDF Analysis step 6 processing sim output step 6 sim compute step 600
...
```

In Situ Visualization with ADIOS





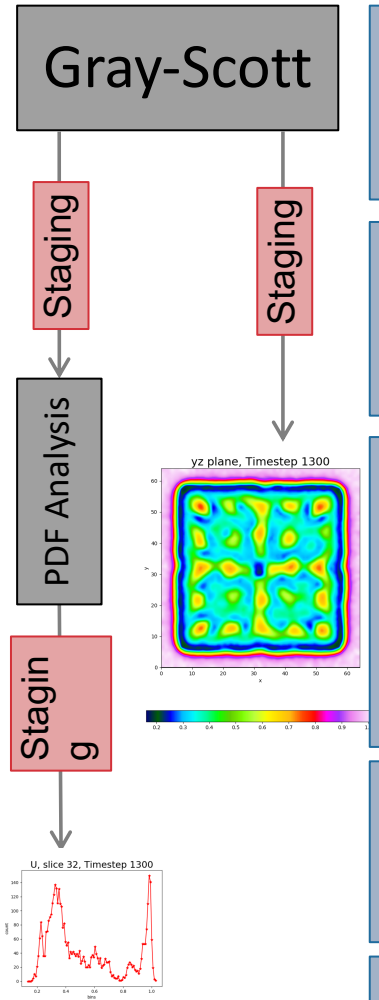

```
edit adios2.xml and change PDFAnalysisOutput to SST as well

$ ./cleanup.sh data
```

```
$ mpirun -n 4 adios2-gray-scott settings-staging.json
 Simulation writes data using engine type:              SST
=========================================
grid:               64x64x64
steps:              60000
 …
```

```
$ mpirun -n 2 adios2-pdf-calc gs.bp pdf.bp 100
PDF analysis reads from Simulation using engine type:  SST
PDF analysis writes using engine type:                 SST
PDF Analysis step 0 processing sim output step 0 sim compute step 0
...
```

```
$ python3 pdfplot.py -i pdf.bp
PDF Plot step 0 processing analysis step 0 simulation step 0
PDF Plot step 1 processing analysis step 1 simulation step 100
PDF Plot step 2 processing analysis step 2 simulation step 200
PDF Plot step 3 processing analysis step 3 simulation step 300
...
```

In Situ Visualization with ADIOS




```
edit adios2.xml and change AnalysisOutput to SST
                           VizInput       to SST
$ ./cleanup.sh data
```

```
$ mpirun -n 4 adios2-gray-scott settings-staging.json
 Simulation writes data using engine type:              SST
=========================================
```

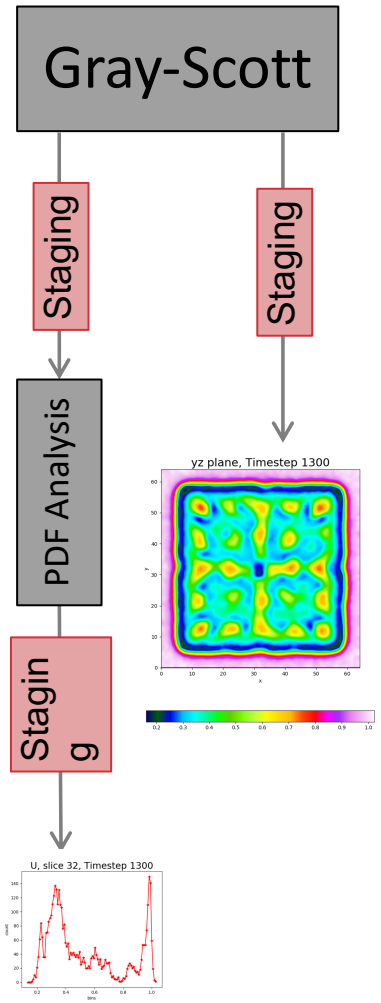
```
$ mpirun -n 2 adios2-pdf-calc gs.bp pdf.bp 100
PDF analysis reads from Simulation using engine type:  SST
PDF analysis writes using engine type:                 SST
PDF Analysis step 0 processing sim output step 0 sim compute step 0
...
```

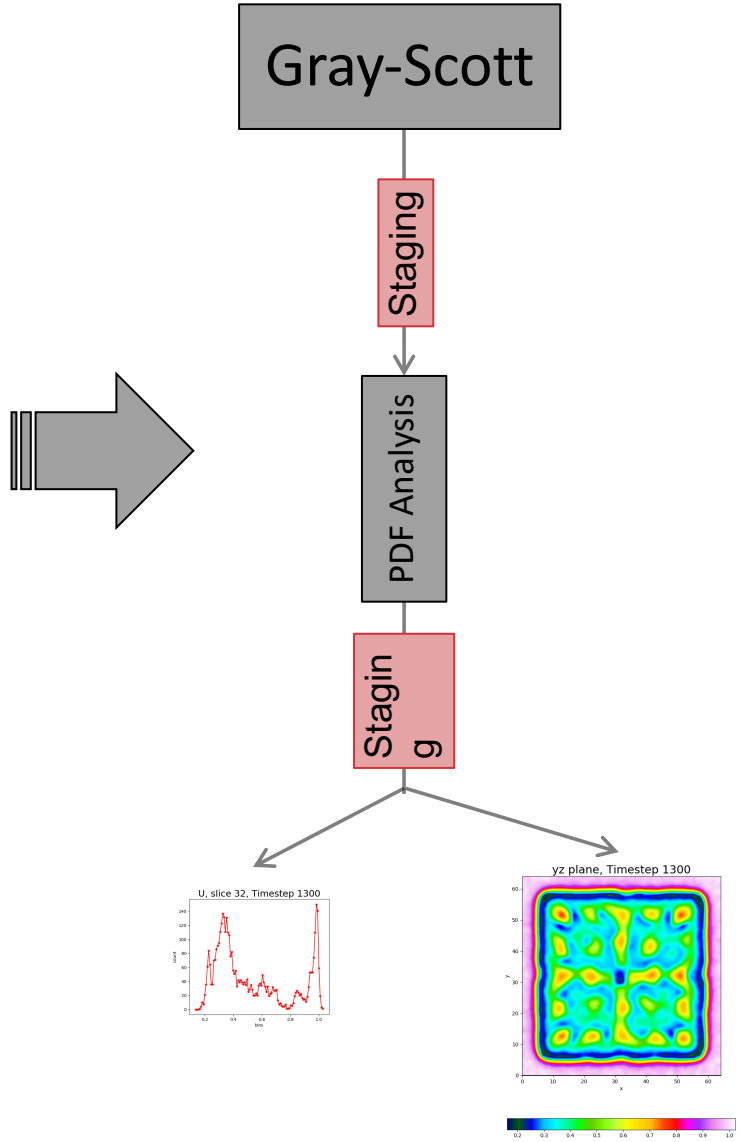
```
$ python3 pdfplot.py -i pdf.bp
PDF Plot step 0 processing analysis step 0 simulation step 0
...
```

```
$ python3 gsplot.py -i gs.bp
GS Plot step 0 processing simulation output step 0 or computation step 0
GS Plot step 1 processing simulation output step 1 or computation step 100
GS Plot step 2 processing simulation output step 2 or computation step 200
...
```

Changing the pipeline

- Run both Plotting from PDFAnalysis output





Hints

- pdf_calc: Add extra command-line argument to : YES
  to write U, V variables to pdf.bp
- gsplot.py: Plot "pdf.bp"
  instead of "gs.bp"
  - Change on command line