Data Staging and Coordination Strategies for Coupled Simulation Workflows

Session Leader: Ana Gainaru

Speakers

- Visualization: Hank Childs, Dave Pugmire
- Application: Stephane Ethier
- I/O management: Bogdan Nicolae, Greg Eisenhauer, Pradeep Subedi, Scott Klasky

EXASCALE COMPUTING PROJECT

National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

Introduction

- One hour BOF
  - Statement sessions: 5 minute statement from each panelist (1 minute for questions) ~40 min
  - Q&A session: 15-20 minutes discussion
- Ask questions in the comments
  - I will keep track of all the questions
  - Broader questions will be saved for the end session
  - I will call you out during the 15 minute discussion

Panel of experts

- Visualization
  - Hank Childs, University of Oregon
  - Dave Pugmire, Oak Ridge National Laboratory
- Application
  - Stephane Ethier, Princeton Plasma Physics Laboratory
- I/O management
  - Bogdan Nicolae, Argonne National Laboratory
  - Greg Eisenhauer, Georgia Institute of Technology
  - Pradeep Subedi, Rutgers Discovery Informatics Institute
  - Scott Klasky, Oak Ridge National Laboratory

1. What are the current trends in data staging?

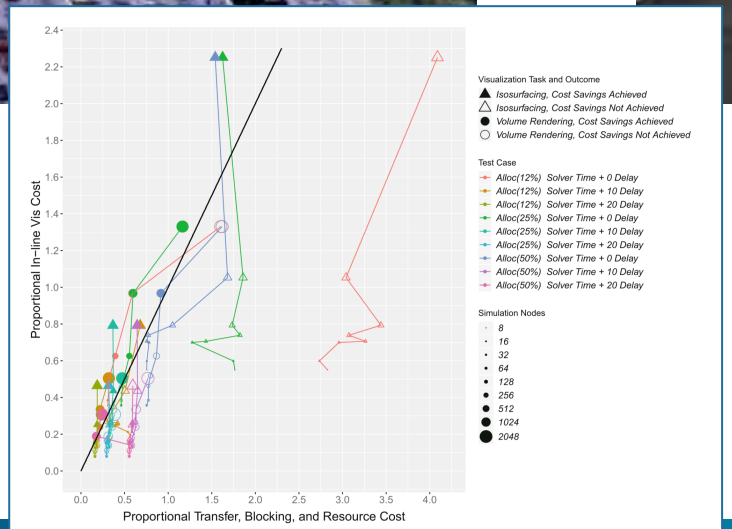
2. What are current solutions?

3. What did you learn using / designing data staging methods?

4. What should we do better in the future? What can make future methods successful?

Data Staging and Coordination Strategies for Coupled Simulation Workflows

Hank Childs, University of Oregon



EXASCALE COMPUTING PROJECT

NNSA National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

Questions from Ana

I am a vis/analysis person, so thinking about ”in transit” in situ instead of “inline” in situ

- What are the current trends in data staging / streaming requirements?
- What are current solutions?
- What did you learn using / designing data staging / streaming methods
- What should be do better in the future? What can make future methods successful?

My answer:
Can we make staging efficient?
Really efficient?

“Inline” (no staging): Run with 100 nodes x 100 seconds = 10,000 node-seconds
“In transit” (staging): Run with 110 nodes x 80 seconds = 8,800 node-seconds

Questions from Ana

- What are the current trends in data staging / streaming requirements?
  - → extra nodes are expected to do tasks, but idle time is tolerated
- What are current solutions?
  - → (for me) Ascent+ADIOS
- What did you learn using / designing data staging / streaming methods
  - → We can *improve* performance / cost
- What should be do better in the future? What can make future methods successful?
  - → More focus on efficiency

In Transit Vs Inline

- J. Kress, et al.
- “Loosely coupled in situ visualization: A perspective on why it’s here to stay”
- ISAV@SC Workshop 2015

		Favored Paradigm	
	Comparison Factor	In-line	In-transit
Data Factors	Data Access	✓	
	Data Movement	—	—
	Data Duplication	✓	
	Data Translation		✓
	Exploratory Visualization	—	—
Implementation Factors	Scalability		✓
	Ease of Use		✓
	Coordination	✓	
	Fault Tolerance		✓
	Resource Requirements	—	—

In Transit Analogy

“Inline”

“In transit”

Credit to: James Kress & Dave Pugmire, ORNL

Why In Transit Should Cost More

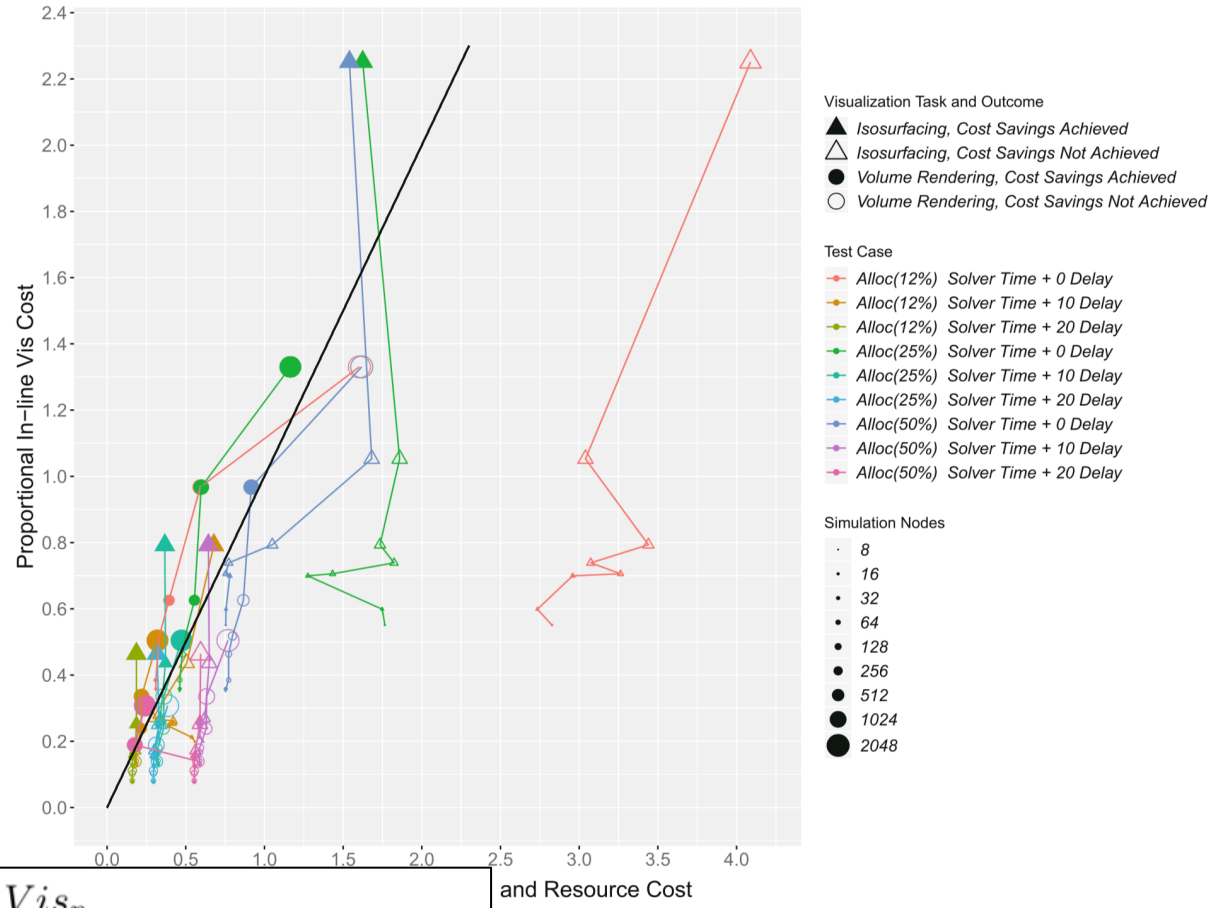
- Dedicated resources may sit idle
- Dedicated resources may not be able to keep up
  - One (very costly) decision is to block the simulation
- Transfer costs

Why In Transit Could Cost Less

- Key observation: can make algorithms more efficient
- Notional example
  - Inline scenario
    - Algorithm A runs for 2s on 1,000 compute nodes
    - Cost: 2,000 node-seconds
  - In transit scenario
    - Algorithm A runs for 200s on 5 compute nodes
    - Cost: 1,000 node-seconds
- How could in transit be faster? (amount saved) > (transfer costs + idle costs + blocking costs)

Can In Transit Be Faster Than Inline?

- Answer: sometimes
- Need a non-scalable algorithm and high scale
- Also need in transit resources to be “rightsized”
- Paper has more details



$$(1 + Res_p) \times (Recv_p + \frac{Vis_p}{VCEF \times Res_p}) < (1 + Vis_P)$$

Study Takeway

- Takeaway #1: running non-scalable algorithms at scale hurts
- Takeaway #2: a way to address non-scalable algorithms is to ... not run them at scale
- Publications:
  - J. Kress, et al. Opportunities for Cost Savings with In Transit Visualization. At ISC High Performance 2020
  - J. Kress, et al. Comparing the Efficiency of In Situ Visualization Paradigms at Scale. At ISC High Performance 2019
  - J .Kress, et al. Comparing Time-to-Solution for In Situ Visualization Paradigms at Scale. At LDAV 2020
  - Full author list: J.Kress, M.Larsen, J.Choi, M.Kim, M.Wolf, N.Podhorszki, S.Klasky, H.Childs, and D.Pugmire.

In Situ Inefficiencies

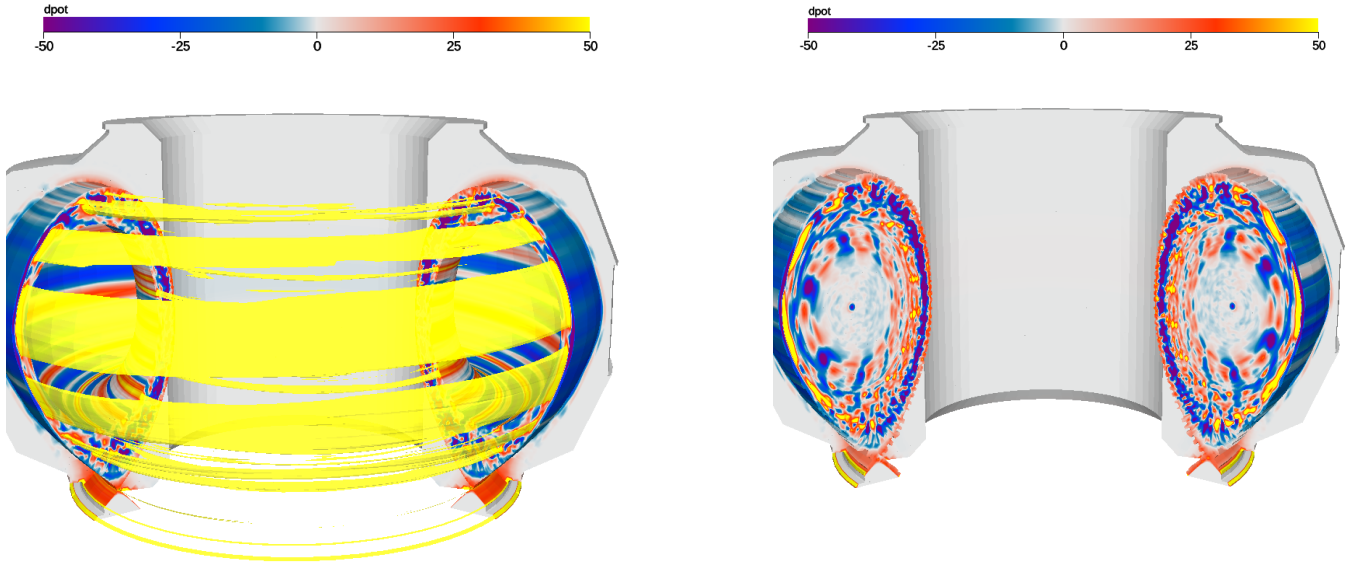
Inline	Hybrid Techniques	In Transit
Scalability	???	Rightsizing
Variability		Overhead

Acknowledgments

- This research was supported by the Exascale Computing Project (17- SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

Data Staging and Coordination Strategies for Coupled Simulation Workflows

Dave Pugmire, Oak Ridge National Laboratory



EXASCALE COMPUTING PROJECT
National Nuclear Security Administration
U.S. DEPARTMENT OF ENERGY | Office of Science

Questions from Ana

- What are the current trends in data staging / streaming requirements?
- What are current solutions?
- What did you learn using / designing data staging / streaming methods
- What should we do better in the future? What can make future methods successful?

What are the current trends in data staging / streaming requirements?

Cores, accelerators and nodes, **Oh MY!**

- Compute in many different places
- Trend towards supercomputing “ecosystems”

Institutional compute

- Simulations
- Digital Twins
- Analysis, Visualization
- ...

Network

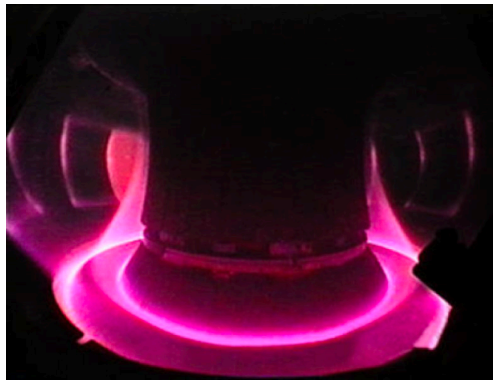
Edge Compute

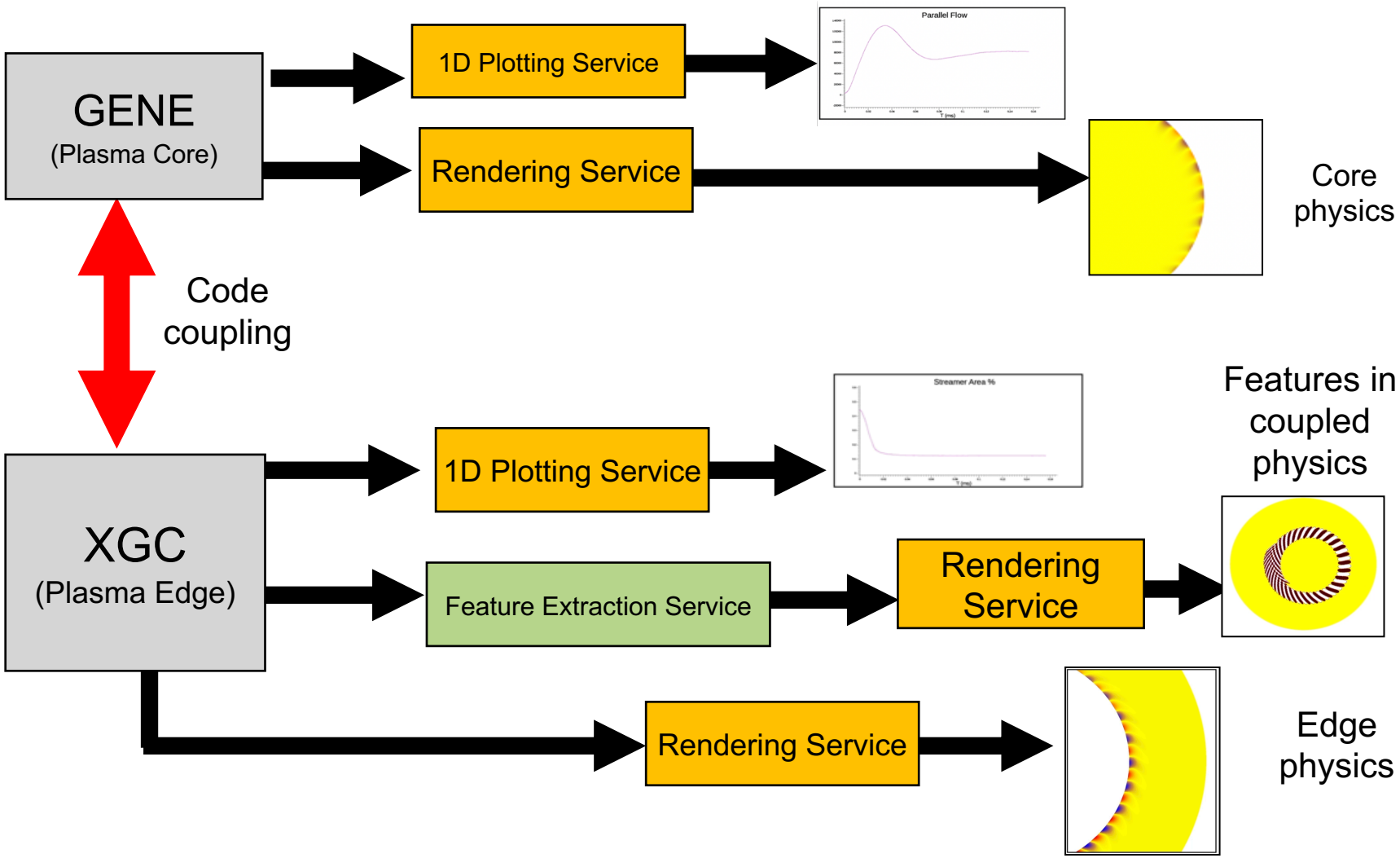
Edge Compute

Edge Compute

EXASCALE COMPUTING PROJECT

Recent motivating example: Coupled Fusion Simulation

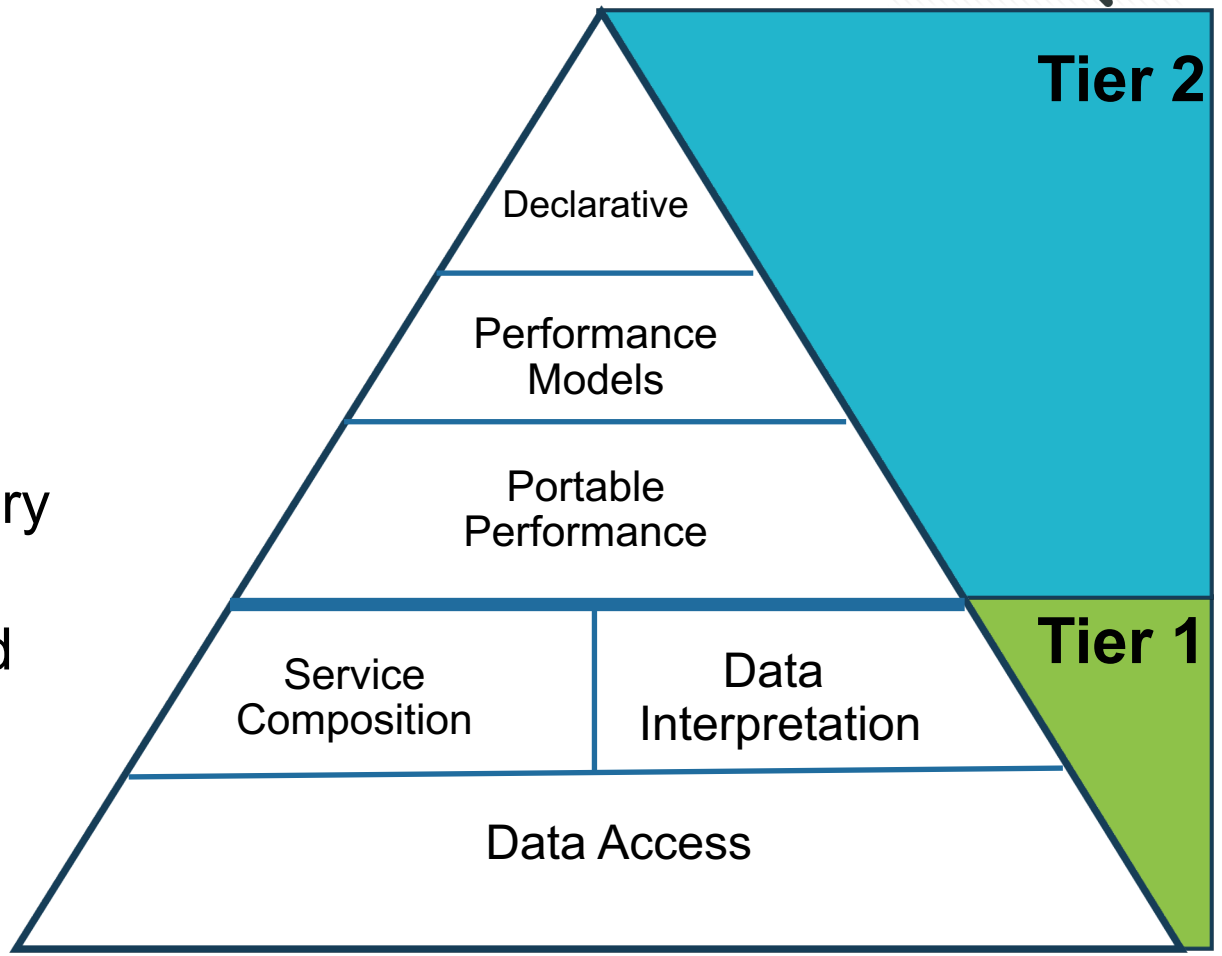




What are current solutions (for me)?

Xenia
Greek term for hospitality

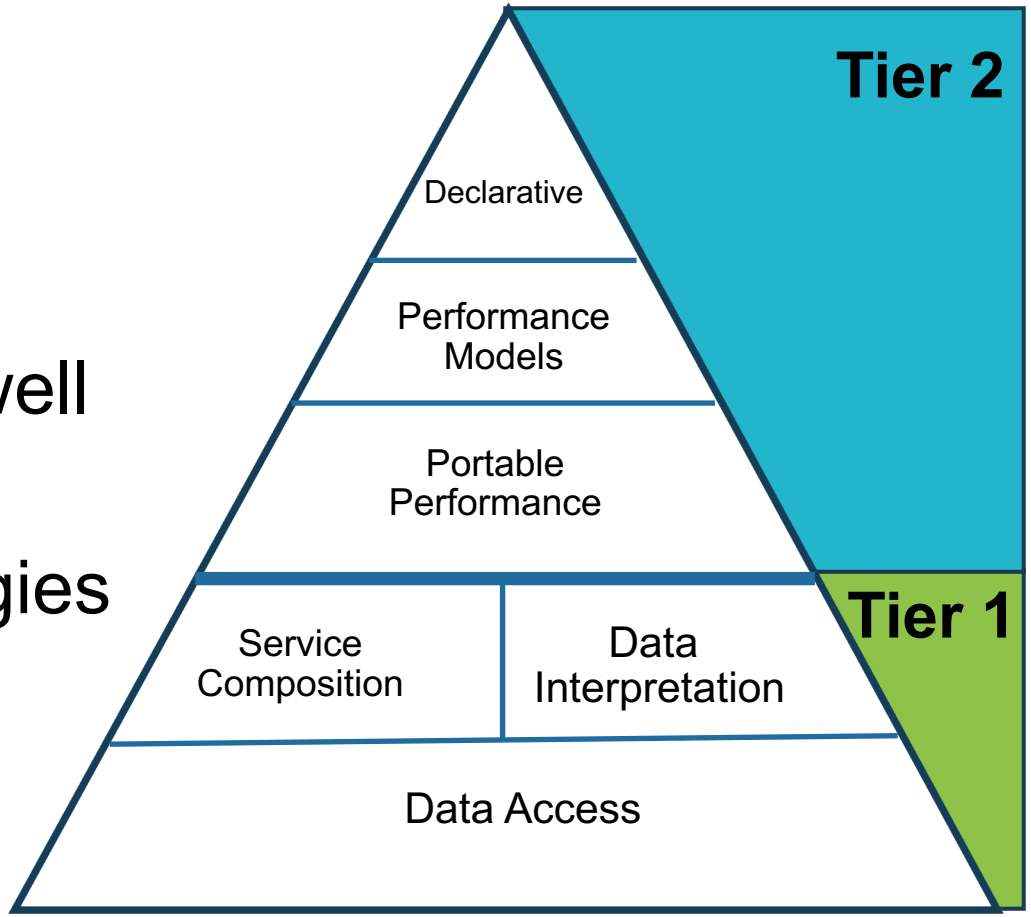
- Xenia: Visualization as a Service
  - Vis and analysis was always hard
    - But the **process** was well understood
  - Now, the process is also hard
    - **Data**: large, reduction, streaming
    - **Hardware**: heterogenous, deep memory hierarchy
    - **Workflows**: tighter integration required
- Builds on top capabilities:
  - VTK-m, ADIOS, FIDES, EFFIS



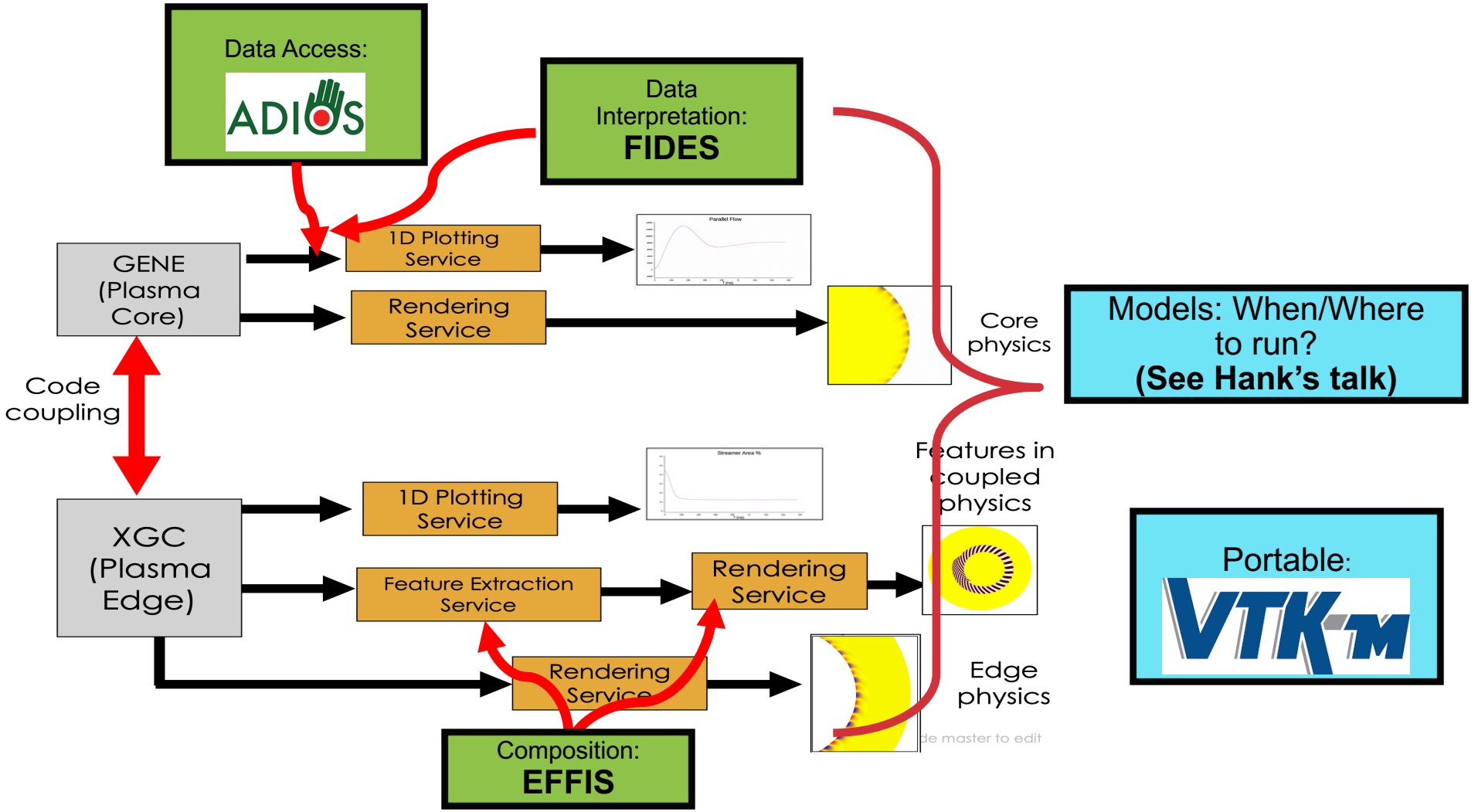
Pugmire et al., “Visualization as a Service for Scientific Data”, Smoky Mountains Computational Sciences and Engineering Conference, August 2020

What did you learn using / designing data staging / streaming methods

- Staging / Streaming is a new paradigm
  - It is different and very hard
- Production analysis / vis tools work very well with files
- **Co-design** across collaborating technologies is key
    - This led us to Xenia



What did you learn using / designing data staging / streaming methods



What should we do better in the future? What can make future methods successful?

- Things are only going to get more complicated
  - Complexity and distribution of **everything** will increase
    - A lot
- We need to work together
  - Our own area is complicated enough
  - Invent new wheels?
    - Or leveraging other expert-made wheels to create scalable solutions

Acknowledgments

- This research was supported by the Exascale Computing Project (17- SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

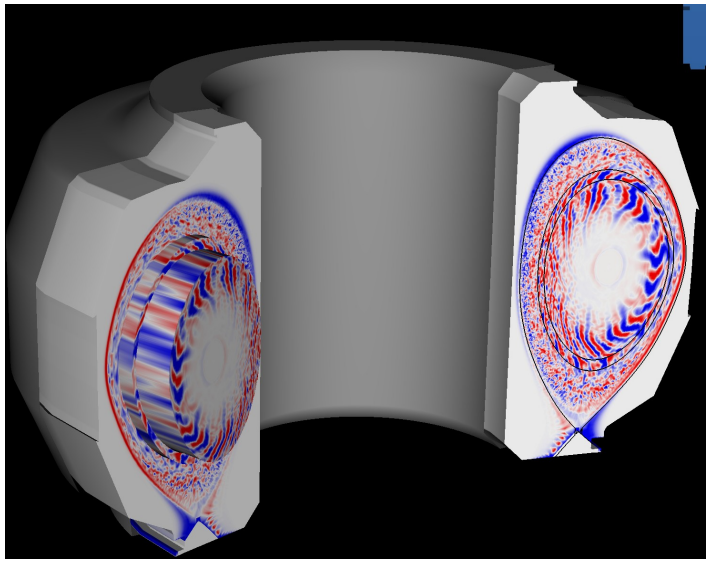
Data movement requirements in the fusion Whole-Device Modeling exascale project (WDMApp)

Approved for public release

Stéphane Éthier, *Princeton Plasma Physics Laboratory*

for the WDMApp team (P.I.: Amitava Bhattacharjee, PPPL)

https://confluence.exascaleproject.org/display/ADSE12



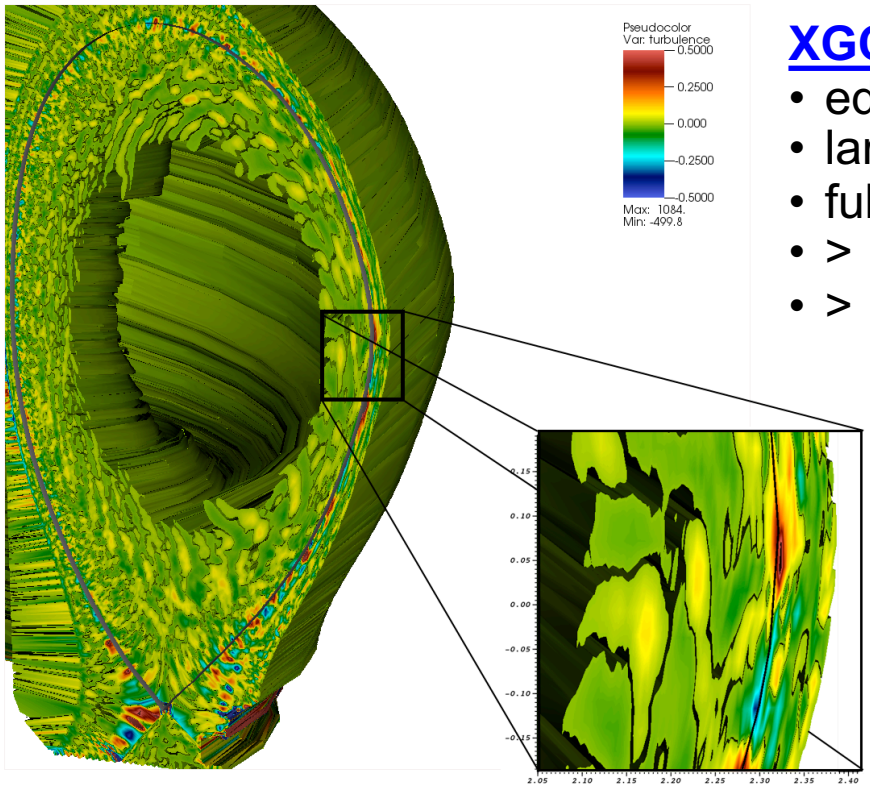
EXASCALE COMPUTING PROJECT

National Nuclear Security Administration

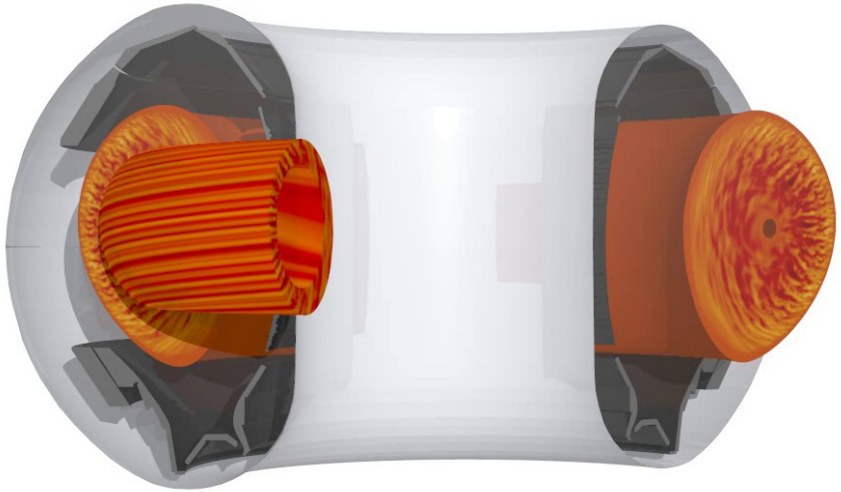
U.S. DEPARTMENT OF ENERGY | Office of Science

WDMApp in a nutshell

- **Goal:** to simulate an entire tokamak fusion device with all the relevant physics at all scales
- WDMApp tries to do that from first principles
- We use 5D “gyrokinetic” formulation as the main physics driver for the framework
- Gyrokinetic codes (XGC, GENE/GEM):
  - cover the broadest length and time scales
  - treat important kinetic effects, such as micro-turbulence, magnetic trapping of particles, plasma transport, instabilities, and more
  - are compute intensive and highly scalable
- Additional physics will be MHD, wave heating, energetic particles, wall model, …



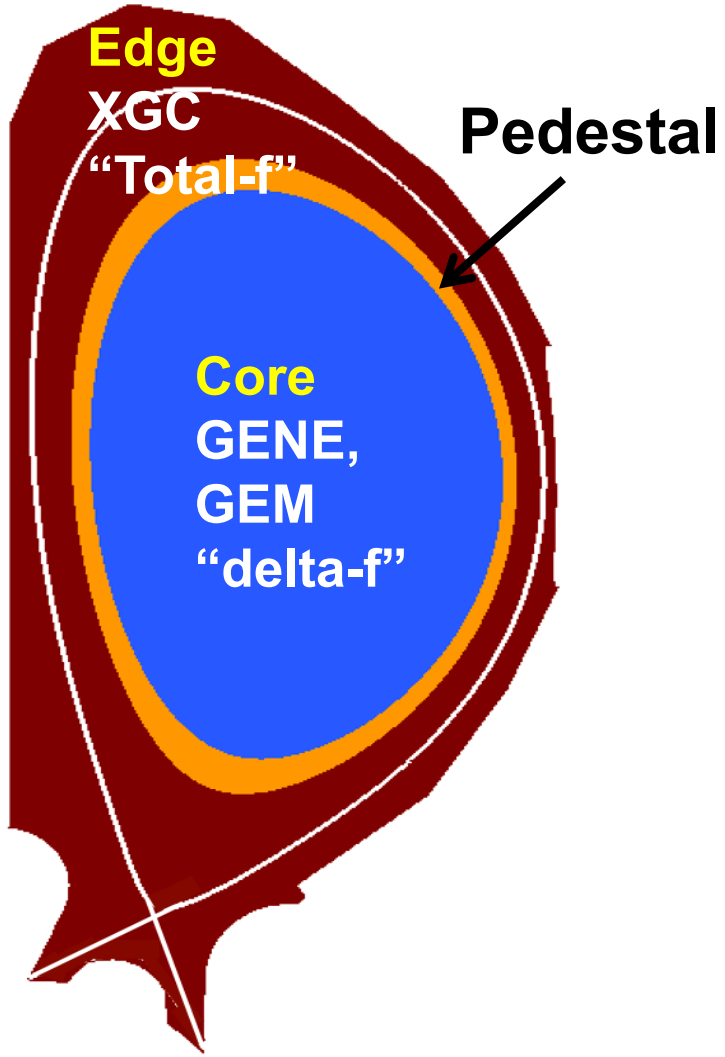
XGC PIC code
- edge physics
- large fluctuations
- full-f distr. func.
- > 1B particles
- > 1M grid pionts



GENE code
- 5D continuum/grid
- “core” physics
- small fluctuations
- delta-f distr. func.
- field-line following grid
- 1M grid points
- 1K velocity pts

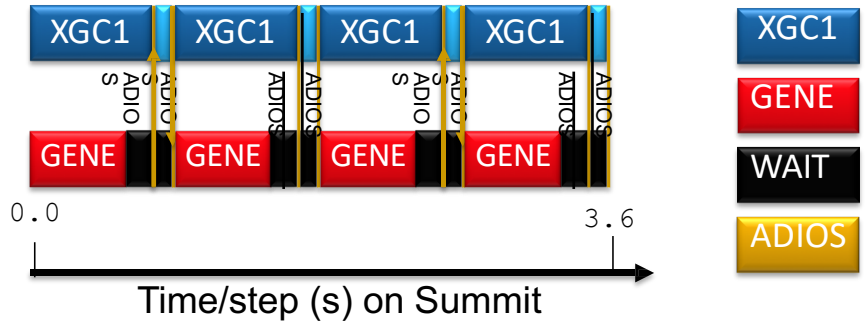
WDMApp coupling framework

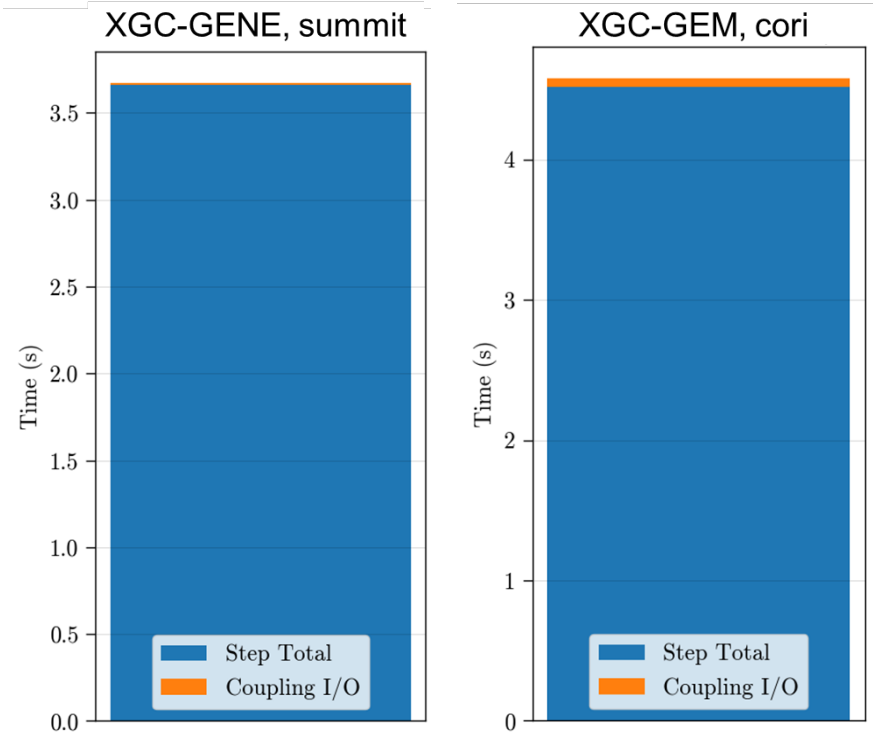
- XGC is optimized for simulating the edge region of the tokamak (only code that can go across the “separatrix”)
  - overkill for the core plasma
- GENE is optimized for core region of the tokamak (cannot go across the separatrix)
  - very fast
- WDMApp couples these codes together, similar to a climate code
  - GENE in the core, XGC in the edge region
- Need to exchange information at the boundary between the 2 regions
- What information? Fields only or both distribution functions and fields???



Data exchange technology: **ADIOS2**

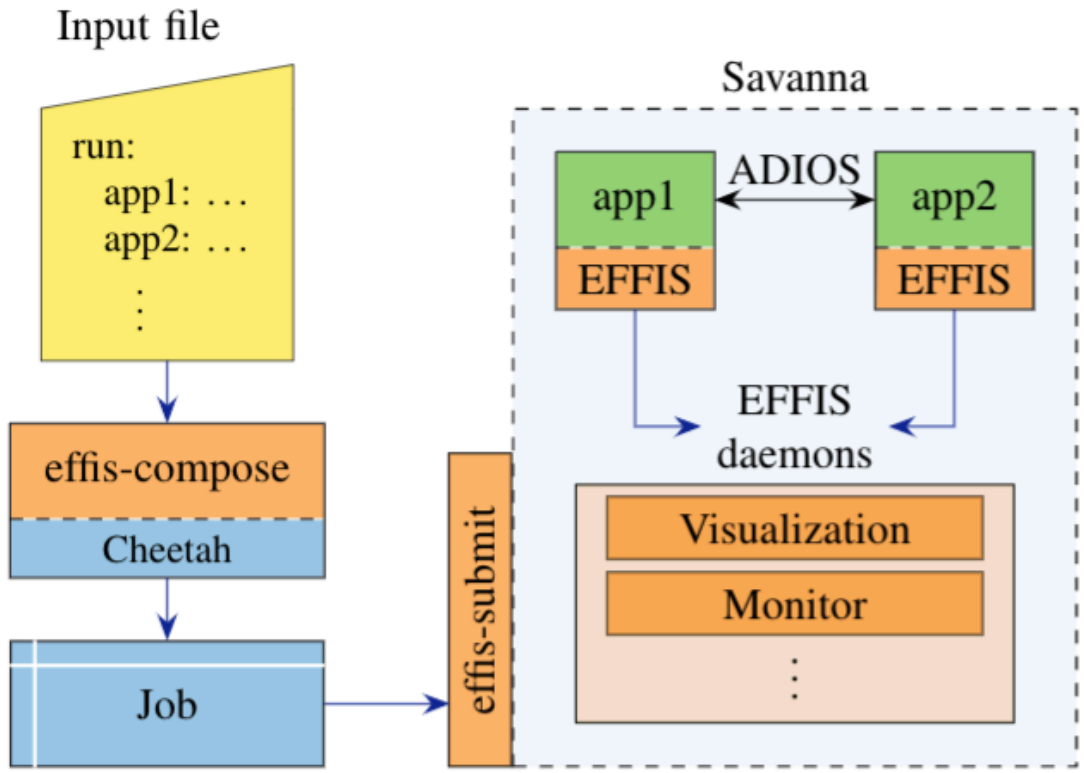
- The two executables, XGC and GENE, are running at the same time on different nodes and within their own MPI comm world
- ADIOS2 in-memory data transfers are used whenever the codes need to exchange data and synchronize
- I/O to files for checkpoint-restarts and diagnostics also done via ADIOS2 calls
  - checkpoints for XGC are the largest
- It turns out that the field data for the entire domain (covered by both XGC and GENE) needs to be consistent and thus evaluated by algorithm in XGC





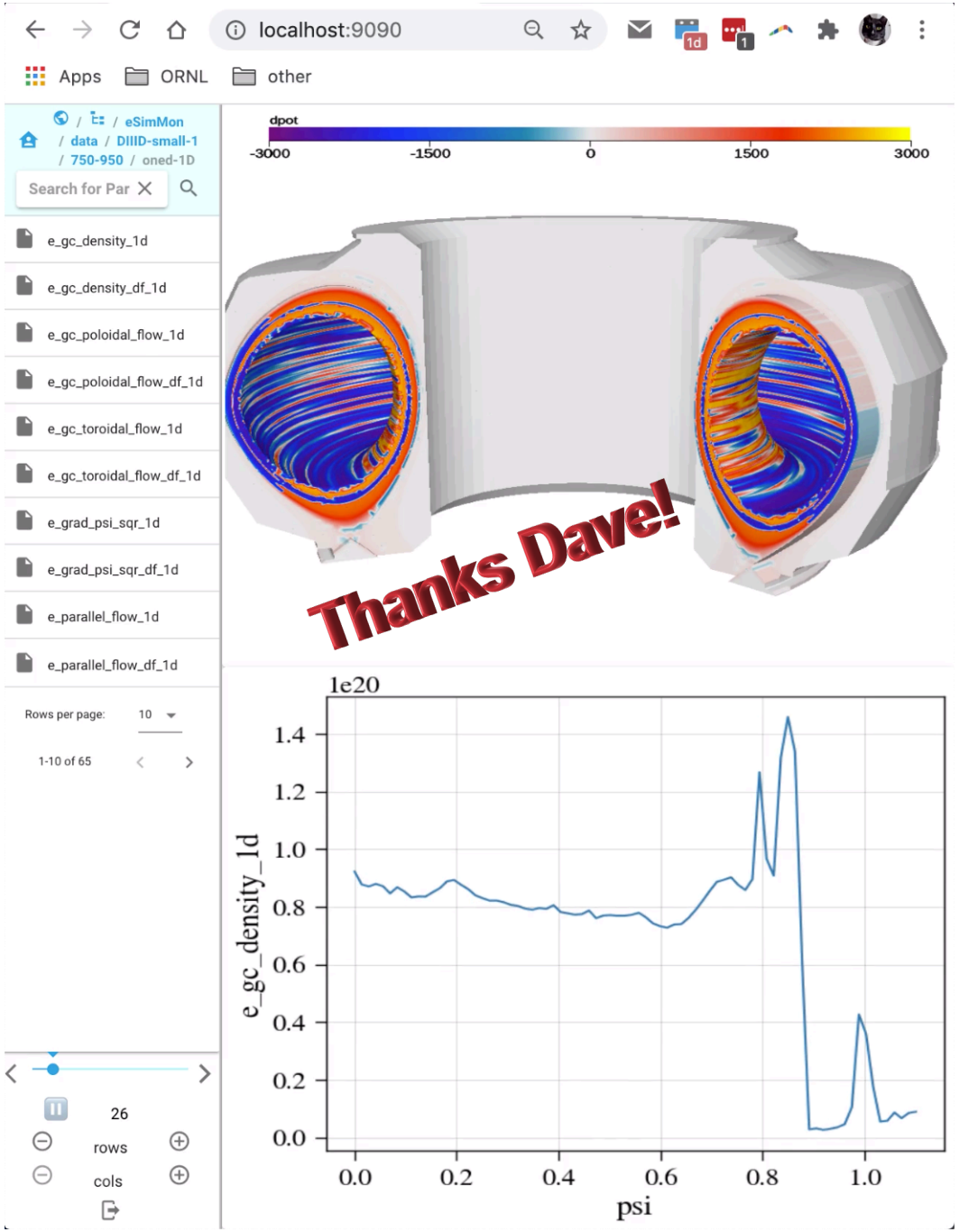
Orchestrator: **EFFIS 2.0**

- EFFIS is a Python-based workflow manager built on top of technologies developed by the ECP CODAR Co-design project
  - Savanna
  - Cheetah
- It simplifies the launching and placement of several applications simultaneously on the various LCF systems (although it can also be used on a laptop…)
- Also takes care of data movement for in-situ analysis and visualization
  - includes its own basic plotting
  - Can also launch user-defined scripts
  - Performance visualization



Dashboard viewer: **eSimMon**

- Developed by Kitware
- EFFIS will run a dashboard packing daemon on the service node (at OLCF nothing is web-accessible from computation nodes)
- Need to use EFFIS plotting pragmas to enable dashboard compatibility:
  - Output image directory mapped to structure
  - *steps* is which EFFIS group plot steps are matched to
  - *name* is a label, which in combination with the executable's label, generates a unique label for the dashboard
- Data moved from compute nodes to web-accessible location to dashboard web server → lots of data to orchestrate



Ana's questions

- Users are often set in their ways. They learn something and stick to it even when something much better comes out!
- Researchers spend a lot of time post-processing/analyzing the results of their simulations
  - This is part of the discovery process
  - In-situ visualization and analysis can speed up that process (and not only for large runs)
- An exascale simulation is challenging enough by itself… coupling several "independent" codes together makes it even harder
  - Please continue to develop those great data movement/staging technologies so that users don't have to worry about (in-memory data exchange, compression, asynchronous data transfers, etc.)
- What the future holds?
  - Increasingly complex coupled codes requiring optimized data exchanges and I/O

Data Staging: Challenges and Opportunities

Bogdan Nicolae
Argonne National Laboratory
bnicolae@anl.gov

Current trends in data staging / streaming

- **Diverse set of scenarios, each with its own trade-offs**
  - Checkpoint-restart: minimize write overhead
  - Revisit previous intermediate datasets: minimize both write and read overhead
  - Streaming: minimize overhead of accessing iterators and data generators

  => Need versatile runtimes with configurable goals

- **Heterogeneity of storage:**
  - Both at node level (HBM, DDR, SSD) and external repositories (PFS, object stores, burst buffers)
  - Too complex for applications to manage directly

  => Need transparent runtimes that unify storage

- **Asynchronous I/O:**
  - I/O overheads continue to increase (less I/O bandwidth per compute unit)
  - Trade-off between less blocking but more background interference

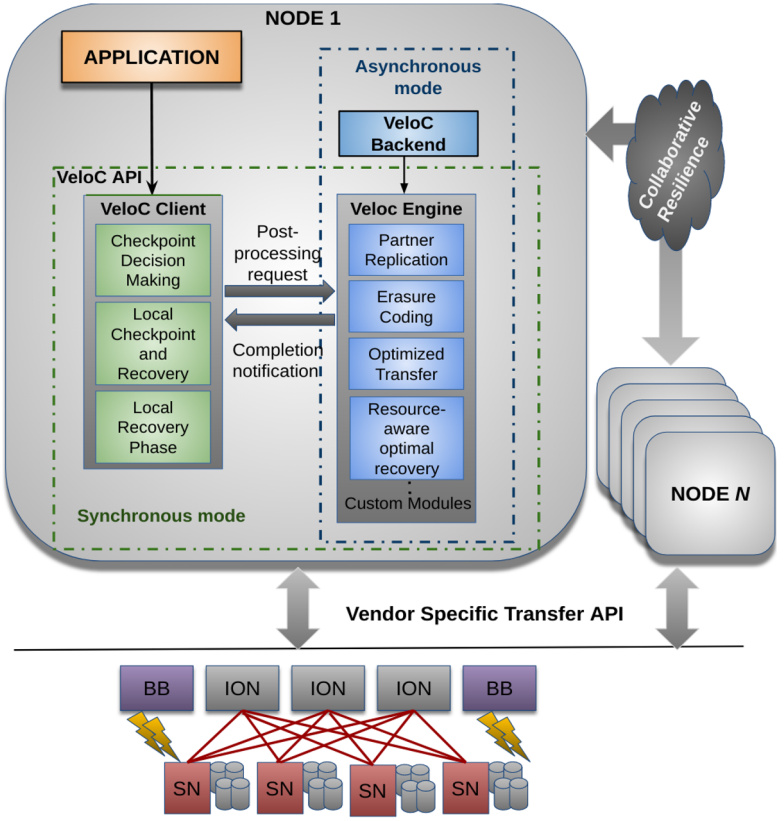
  => Need interference mitigation techniques

- **Advanced features:**
  - Versioning, cloning
  - Smart iterators (adapt to producer-consumer patterns)

Current solutions

ECP VELOC

- Specializes in checkpoint-restart
- Declarative API: specify memory regions, act on them later
- Configurable multi-level I/O and resilience strategies (local, erasure coding, external repositories)
- Configurable post-processing pipeline (checksumming, compression)
- High performance and scalability through asynchronous techniques
- Hides interactions hybrid local storage and external repositories
- Versioning
- Core building block for specialized approaches (e.g. checkpointing of AI models, high-level data management)



*Web: **https://veloc.readthedocs.io***

Lessons learned

- **Overheads often come from unexpected sources:**
  - Serialization of complex data structures (e.g. C++ containers) before write
  - High overhead to access raw data structures in high-level languages/runtimes

  => Need cross-stack optimizations

- **Asynchronous I/O:**
  - Spare compute is often available during runtime, no need to dedicate cores
  - Patterns can often be predicted and leveraged to minimize interference
  - AI/ML applications exhibit apply parallelism at fine granularity (tensors), therefore we need fine-grain async I/O for optimal overlapping

- **Low-level, imperative I/O (read/write or put/get) has limitations**
  - Declarative API simplifies reasoning about data management **and** enables transparent performance optimizations

- **Versatility requires careful design and modularity**
  - HPC platforms have many limitations (launching and detaching processes from scripts, concurrent MPI deployments)
  - Need to minimize and isolate customization code

Opportunities for Improvement

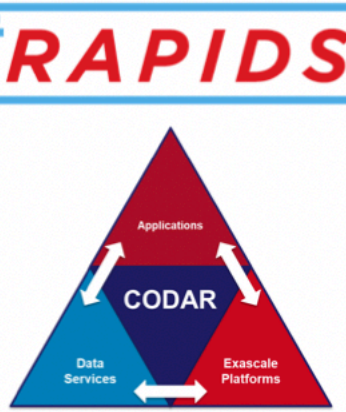
- **I/O performance portability using expressive declarative APIs**
  - Restrictions and hints: scope, persistency, access pattern, expiration date, etc.
  - Multi-objective optimization techniques
  - Complements compute-centric performance portability

- **FAIR-ness (Findability, Accessibility, Interoperability, Reusability)**
  - Metadata annotations and query support
  - Lineage that records evolution and relationship between versions
  - RRR (robustness, reconfigurability, reproducibility)

- **AI/ML-Specific:**
  - Iterators and data generators with replay support (continual learning)
  - Distributed caching with specific patterns (e.g. shuffling)
  - Branching primitives to explore multiple alternatives with low overhead

- **Work in progress:**
  - Several projects: DataStates, Braid, AI-HPC-Big Data Convergence

ADIOS Staging Deeper dive



WDM, HBPS, ISEP, Sirius

Staging BOF– Apr 14, 2021, 11:30 AM

Greg Eisenhauer

SST (Sustainable Staging Transport)
What is it like today?

Designed to be ADIOS's most general-purpose staging engine

- Support every ADIOS feature:
  - MxN
  - Compression
  - Multiple simultaneous streams
  - Multiple readers on a stream
  - Dynamic readers (come and go)
  - Many data transports (RDMA, TCP, RUDP)
  - Timestep consumption modes (every step, most recent only)
  - Queue management (block writer, discard data)
  - Try to be as efficient in special cases as possible without sacrificing generality.

What are current solutions?

In the context of the ADIOS SST engine:

- Writer side data queueing with a background network handler thread
- Different communication mechanisms for metadata/control and for data
  - Metadata/control communication
    - Rank0-to-Rank0 vs. Peer-oriented
    - TCP vs. RUDP
  - Data Communication
    - Libfabric-based RDMA inside a cluster
    - TCP-based for wide-area
- Mechanisms for managing data queueing should reader and writer speeds be mismatched

What did you learn using / designing data staging / streaming methods

- Being general while preserving performance is *hard,* and relatively innocuous things can have big impacts.
- E.G. : In ADIOS, in the most general case, each reader rank gets the metadata for a timestep and can examine it before requesting the data that that particular rank wants
- Implies a request/response data protocol between reader and writer ranks
  - Writers don't know where to send data until particular readers request it
  - Hard to achieve full network utilization or keep analysis fed with data.
    - Different ADIOS staging engines sometimes based on different assumptions that ameliorate this problem
    - In SST, some application scenarios (like fixed communication patterns) can be exploited to lessen impact

What should we do better in the future? What can make future methods successful?

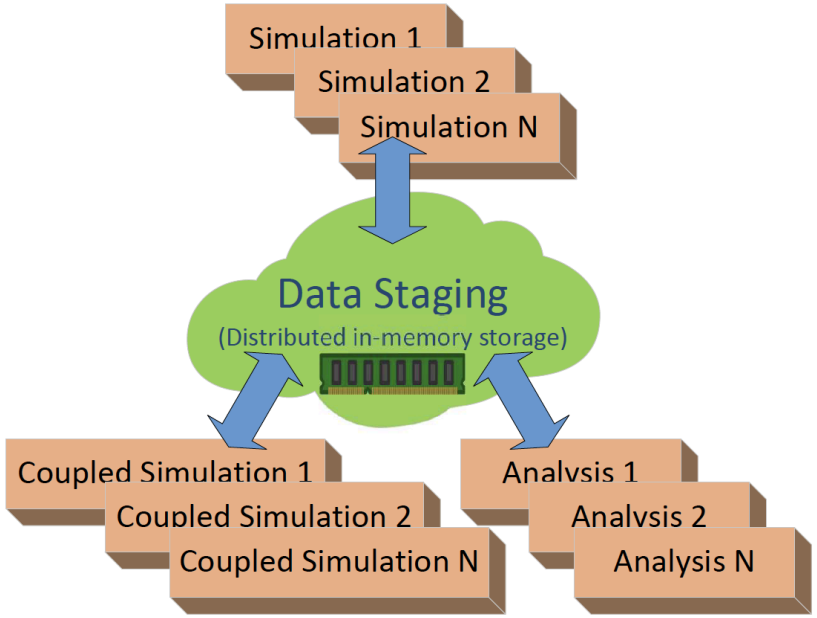
- Performance diagnostics are important, but much harder to do well when multiple codes are linked.
  - Even seemingly simple questions like `what is the current bottleneck' can be hard to answer.
  - We need to do better with instrumentation and tools.
- New technologies like NVM (and DAOS) may play a role in bringing staging and file I/O closer together.

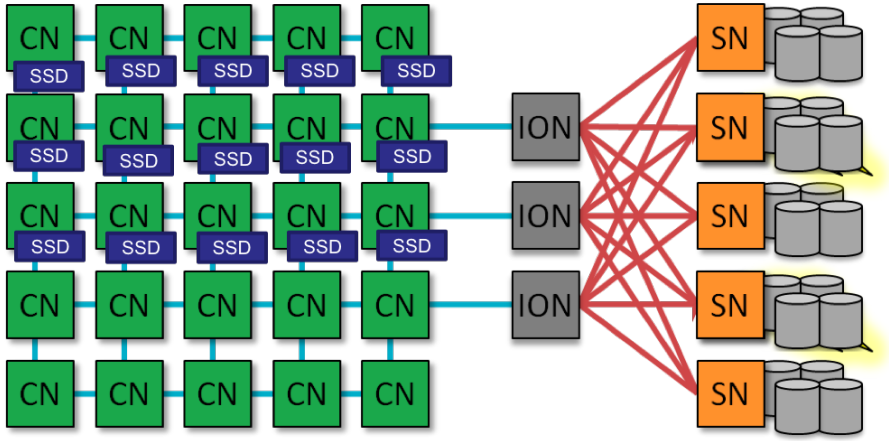
Data Staging: Challenges and Opportunities

Pradeep Subedi
Rutgers University
pradeep.subedi@rutgers.edu

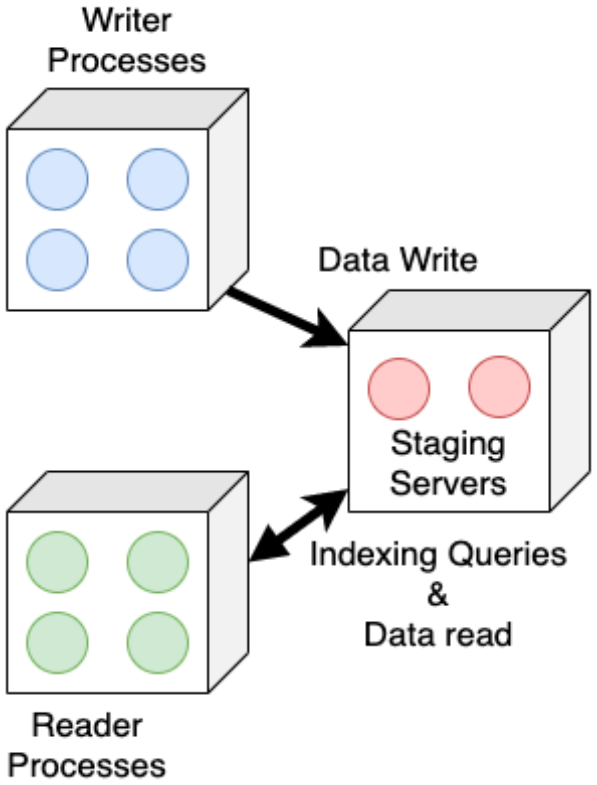
Data Management Challenges in In-Situ Scientific Workflows

- **Scheduling/Mapping**
  - Multi-core /Many-core architectures
  - Computational load balancing among nodes/servers
  - Applications at different temporal and spatial scales
  - Communication pattern known only at runtime
- **Data Placement/Movement**
  - Geo-distributed data centers
  - Data near or far from computation?
  - Variable and asynchronous data access patterns
  - In-situ data transformation
- **Heterogeneous Components**
  - Deep memory hierarchies like burst buffers
  - Efficient data/computation sharing among heterogeneous processors/storage/network

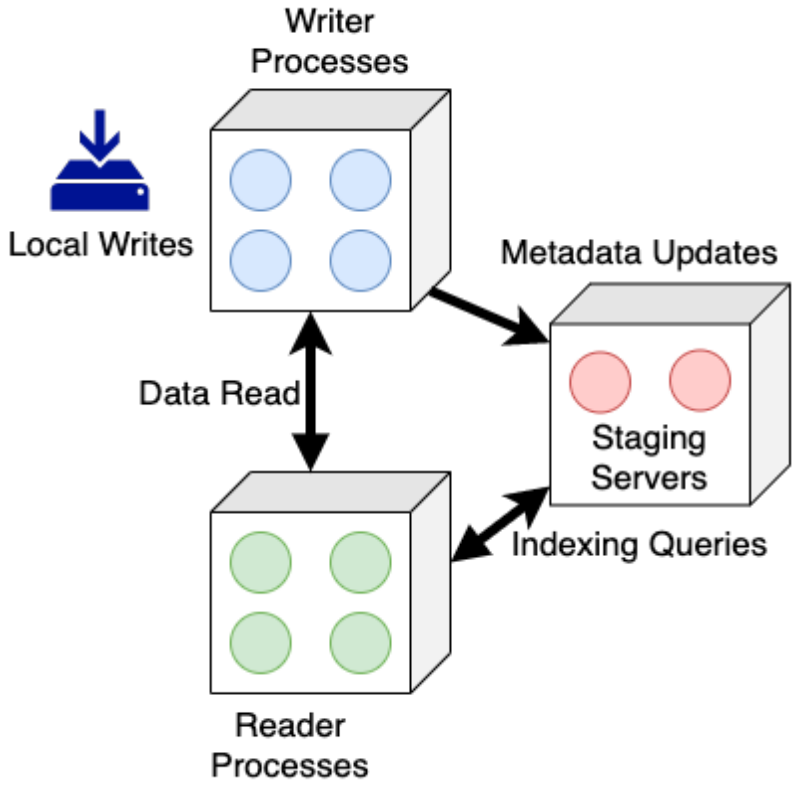




Staging Based Data Management Solutions



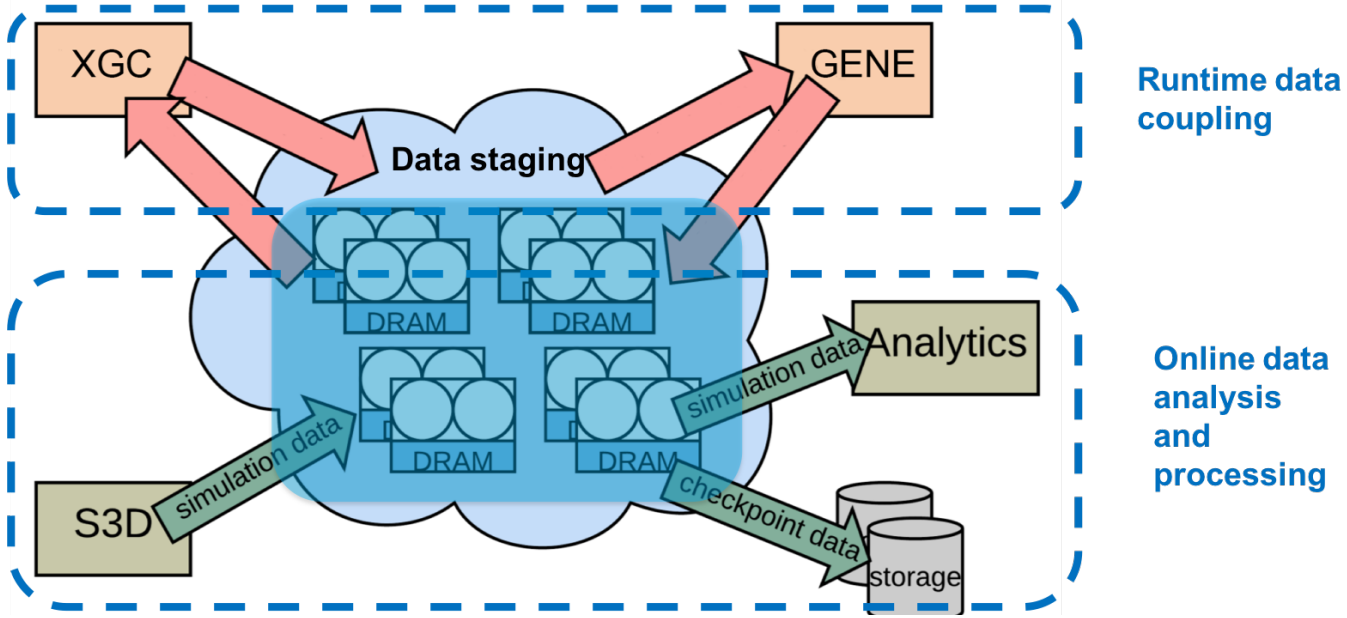
(a) Dedicated Processes/Nodes for Data Storage



(b) Data stored in the writer-processes

The DataSpaces Staging Abstraction

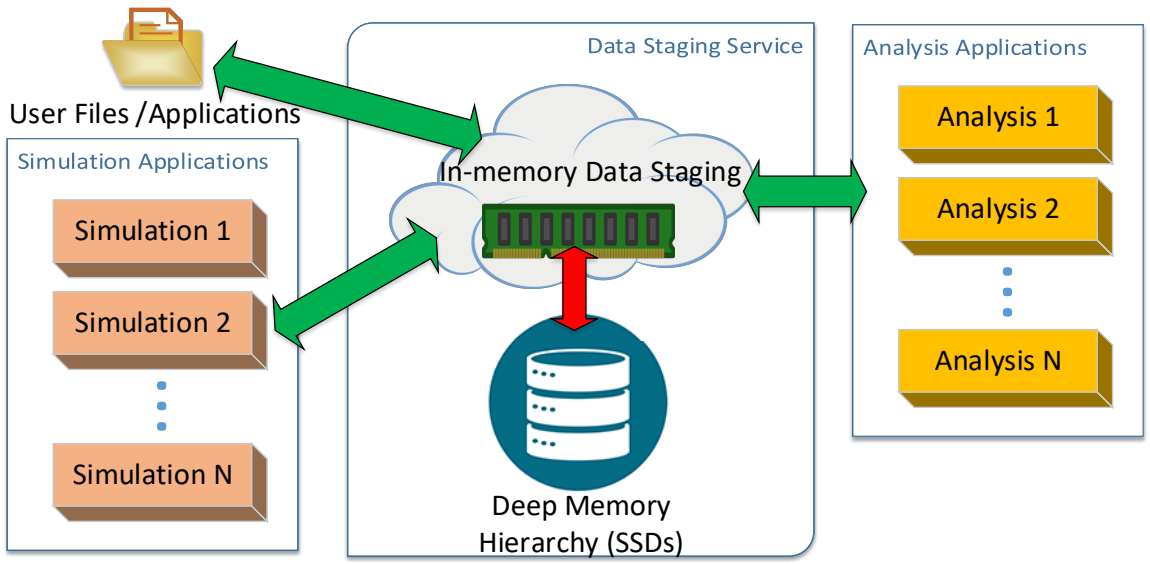
- In-memory storage distributed across set of cores/nodes, using RAM and NVRAM/Burst Buffers.
- In-staging data processing, querying, sharing, and exchange.
- Virtual shared-space programming abstraction.
- Provides an efficient, high-throughput/low-latency asynchronous data transport.

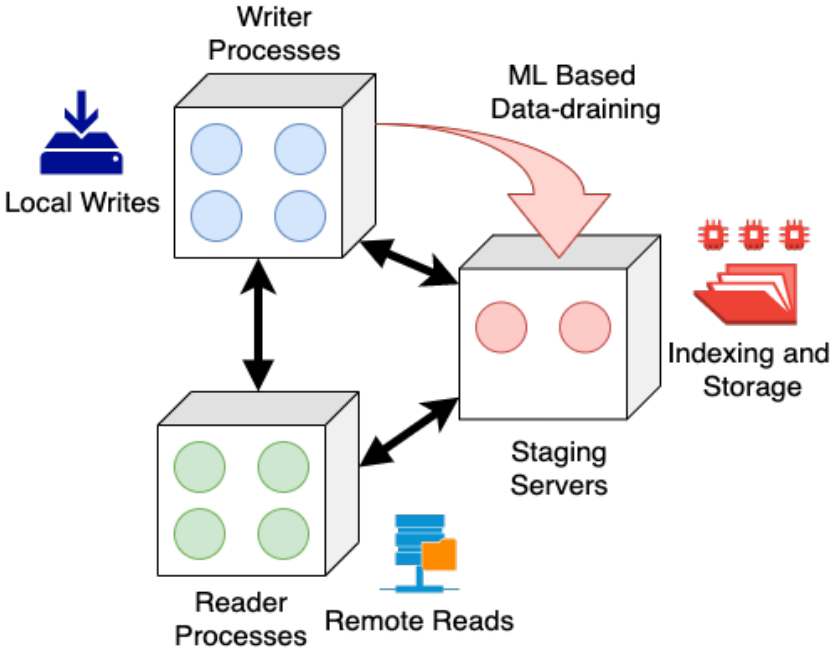


https://github.com/rdi2dspaces/dspaces.git

Research Details & Future Directions

- Automate data management operations by identifying data access patterns.
- Offload analytics (e.g. ML) to background operations on shared space.
- Integrate high-level workflow semantics.





Questions?

Pradeep Subedi
Email: pradeep.subedi@rutgers.edu

www.dataspaces.org
https://github.com/rdi2dspaces/dspaces.git

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

ADIOS: Staging

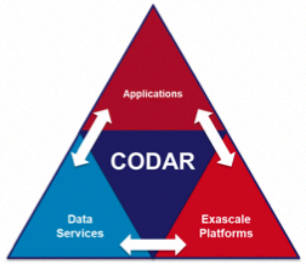
Staging BOF– Apr 14, 2021, 11:30 AM

EXASCALE COMPUTING PROJECT

ADIOS

VTK-m

RAPIDS



WDM, HBPS, ISEP, Sirius

Presenters: S. Klasky[123],

With contributions from: A. Gainaru[1], N. Podhorszki[1,] M. Wolf[1], C. Atkins[8], William Godoy[1], Matthew Wolf[1] , Ruonan Wang[1], Chuck Atkins[8], Greg Eisenhauer[3], Junmin Gu[7], Philip Davis[5], W. Godoy[1], Jong Choi[1], Kai Germaschewski [10], Kevin Huck [11], Axel Huebl[7], Mark Kim[1], James Kress[1], Tahsin Kurc[1], Jeremy Logan[1], Kshitij Mehta[1], Franz Poeschel[8], Eric Suchyta[1], Keichi Takahashi, Lipeng Wan[1], Pradeep Subedi, Mark Ainsworth[19], Berk Geveci [8], Ben Whitney[1]

[1] Oak Ridge National Laboratory, Computer Science and Mathematics Division
[2] University of Tennessee, Knoxville, Department of Electrical Engineering and Computer Science
[3]Georgia Tech, School of Computer Science
[4]New Jersey Institute of Technology
[5]Rutgers University
[6]National Science Foundation, OAC
[7] Lawrence Berkeley National Laboratory
[8] Kitware
[9] Brown University
[10] University of New Hampshire

U.S. DEPARTMENT OF ENERGY | Office of Science | Georgia Tech | OAK RIDGE National Laboratory | Kitware | BERKELEY LAB | THE UNIVERSITY OF TENNESSEE KNOXVILLE | NJIT New Jersey Institute of Technology | THE STATE UNIVERSITY OF NEW JERSEY RUTGERS

What are the current trends in data staging / streaming requirements

- Visualization community has promoted in-line staging
  - Assumption is that visualization algorithms will be fast and can scale as well as apps
- Analytics community has used file-based coupling
  - Assumption is that the I/O overhead is much less than the cost of the analytics
- Distributed computing (WAN) community has said that the latency is large and the focus is on bandwidth
  - Files will always work better, and can work with DMZs better
- ADIOS group focus is that you need to have an easy way to switch from files to
  - In-memory on separate nodes
  - In-memory on separate cores
  - In-memory on the same cores
  - Streaming to different resources
  - Go from multiple consumers to multiple producers, all running on different amount of concurrency
- Depending on the (resources, latency, resiliency, concurrency, bandwidths) users can change the algorithm but keep the same abstractions
  - Creating staging means that is part of a co-design process

What are current solutions?

- Inside ADIOS
  - Inline – if you have consumer which is almost- embarrassingly parallel
  - SSC – if you can have multiple apps running all under MPI, but MPI is NOT resilient
  - DataMan – if you have data movement over the WAN (not fully NxM) [PUSH]
  - DataSpaces – if you want a shared-space abstraction to share data [PULL]
  - SST – if you want resiliency (consumers/producers) [PULL/PUSH]
  - FileStream – if you want full resiliency/good for debugging, or if your application needs the data after the campaign

What did you learn using / designing data staging / streaming methods

- There is never 1 answer to the staging problem
  - We need full flexibility as the consumers/producers/resources change
  - All I/O should be re-designed with these criteria's, i.e., do NOT program for files, program for streams
- We need APIS which are
  - Easy to use
  - Can achieve high performance
  - Implementations can change without having the user change their codes

What should we do better in the future? What can make future methods successful?

- Use ADIOS for the abstraction, and have staging implementations under this 1 abstraction
  - i.e. engines such as HDF5 can be under this, but it can then allow in-line, in-transit, NVRAM, …
  - Have the implementation adapt to the changing problem (and not the API)
- Ensure that this works for MPI and non-MPI codes (i.e. life is more than just MPI)