Oak Ridge National Laboratory

Understanding the Impact of Data Management in Autonomous Scientific Workflows

Ana Gainaru

Scott Klasky, Dmitry Ganyushin, Qian Gong, Kshitij Mehta, Norbert Podhorszki, Eric Suchyta, Lipeng Wan, Ruonan Wang

Dave Pugmire, Jieyang Chen, James Kress

ORNL is managed by UT-Battelle LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Oak Ridge National Laboratory

Table of contents

- Data management systems for coupled applications
- Performance for I/O kernels
- Performance for typical patterns on Summit
  - Embarrassingly parallel
  - Traditional HPC
  - Emerging applications
- Data management in AI applications

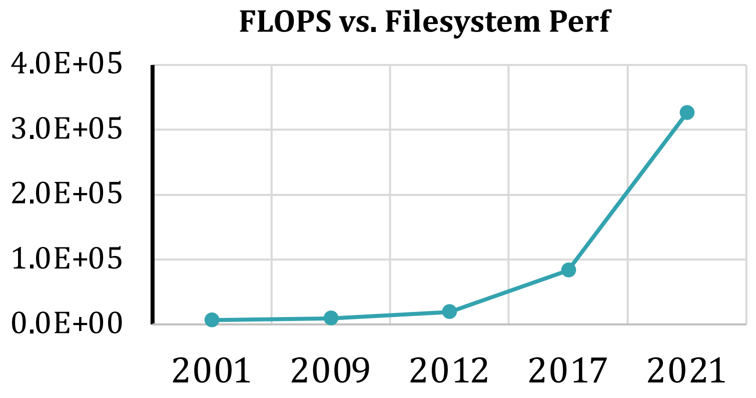
Table of contents

- **Data management systems for coupled applications**
- Performance for I/O kernels
- Performance for typical patterns on Summit
  - Embarrassingly parallel
  - Traditional HPC
  - Emerging applications
- Data management in AI applications

Why do we need data management?

- Data rates has continued to grow at a far greater pace than the development of the network and storage capabilities.

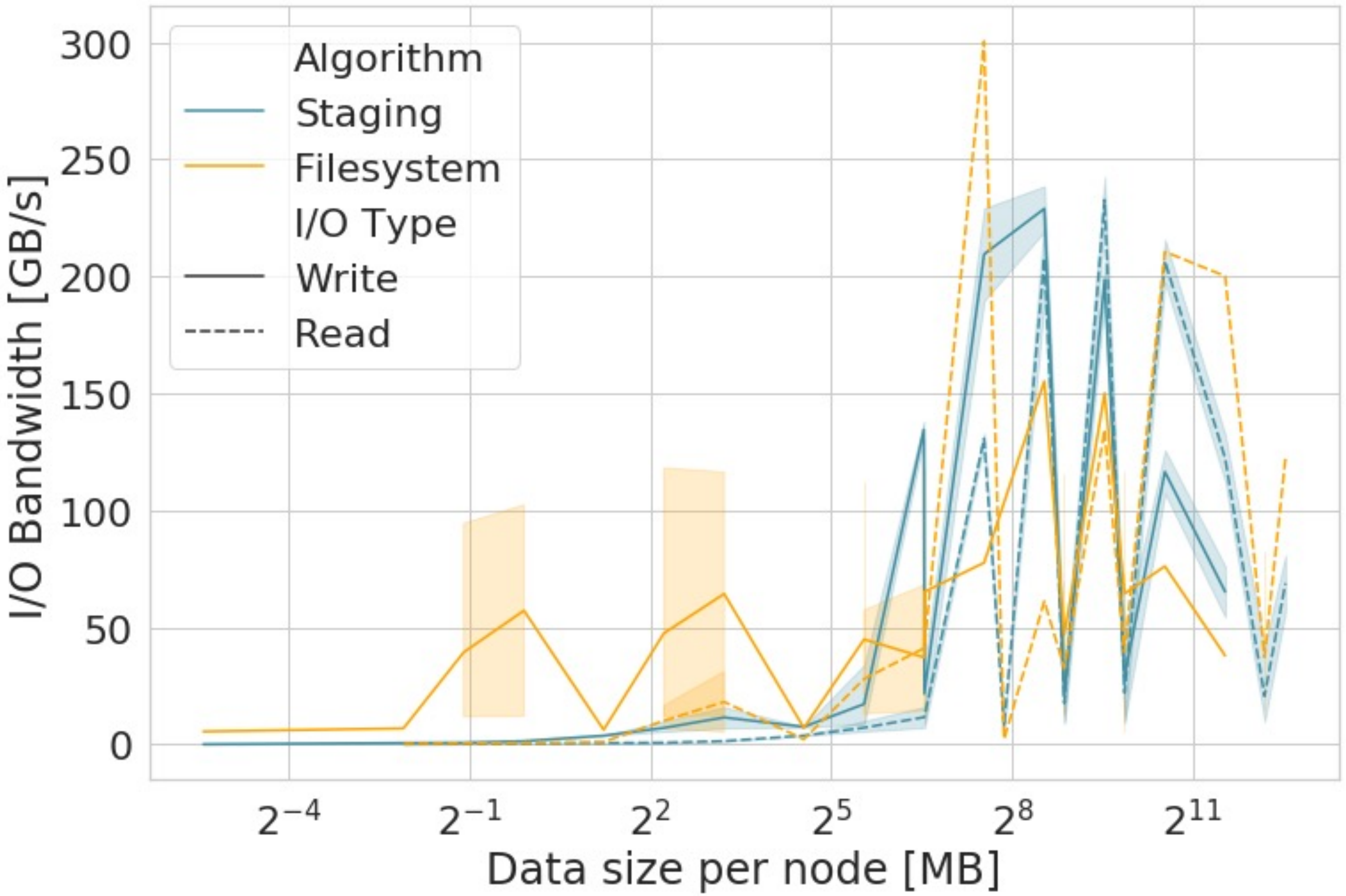
System	Filesystem perf	FLOPS	Ratio
Seaborg	0.003 TB/s	20 TFLOPS	1.50E-04
Jaguar	0.24 TB/s	2300 TFLOPS	1.04E-04
Titan	0.0014 PB/s	27 PFLOPS	5.19E-05
Summit	0.0024 PB/s	200 PFLOPS	1.20E-05
Frontier	0.0046 PB/s	1500 PFLOPS	3.07E-06



- I/O intensive apps
  - Minimize the time applications spend in I/O

Why do we need data management?

- Performance variability
  - Caused by application characteristics
  - **Goal** Achieve high performant I/O on a variety of configurations
- Enable self-describing output for all types of I/O



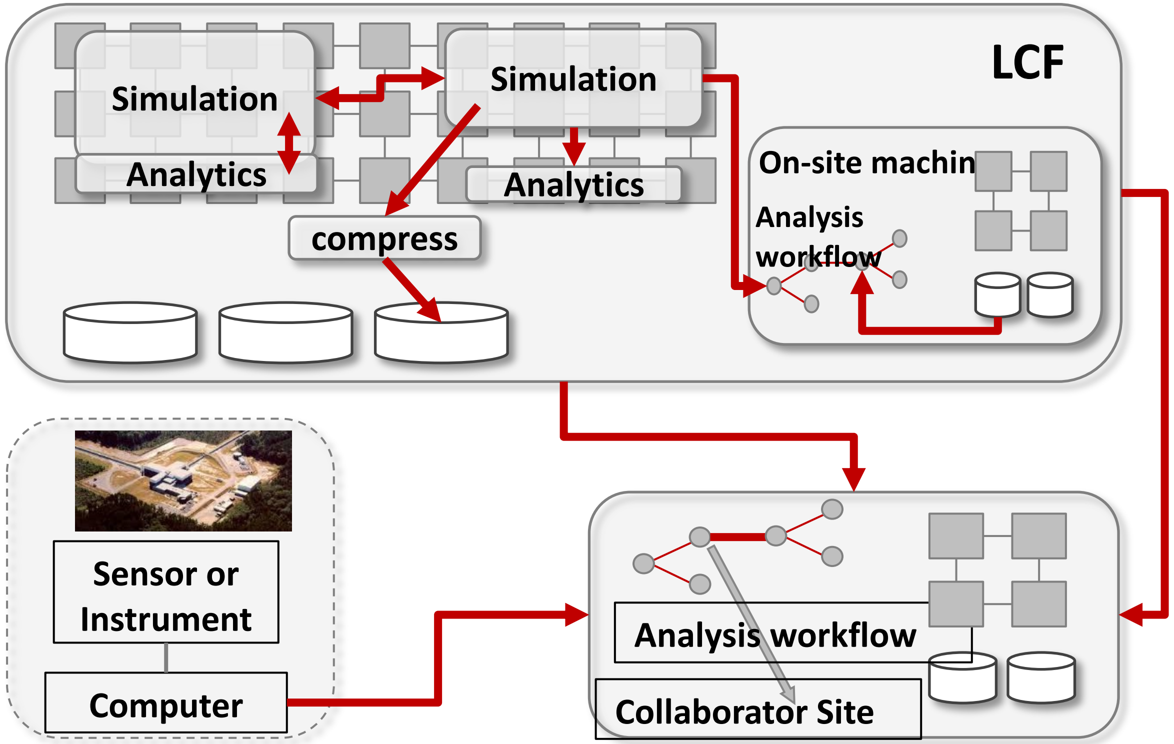
High-Performance Pub/Sub I/O framework

Vision

- Create a high performance I/O abstraction to allow memory/file data subscription service
- Create a sustainable solution to work with multi-tier storage and memory systems

Research Details

- Declarative, publish/subscribe API is separated from the I/O strategy and use of multi-tier storage
- Multiple implementations (engines) provide functionality and performance in different use cases
- Data reduction techniques are incorporated to decrease storage cost



Summit write performance with ADIOS

Application	Nodes/GPUs	Data Size per step	I/O speed
SPECFEM3D	3200/19200	250 TB	~2 TB/sec
GTC	512/3072	2.6 TB	~2 TB/sec
XGC	512/3072	64 TB	1.2 TB/sec
LAMMPS	512/3072	457 GB	1 TB/sec

ADIOS

- Self-describing Scientific Data

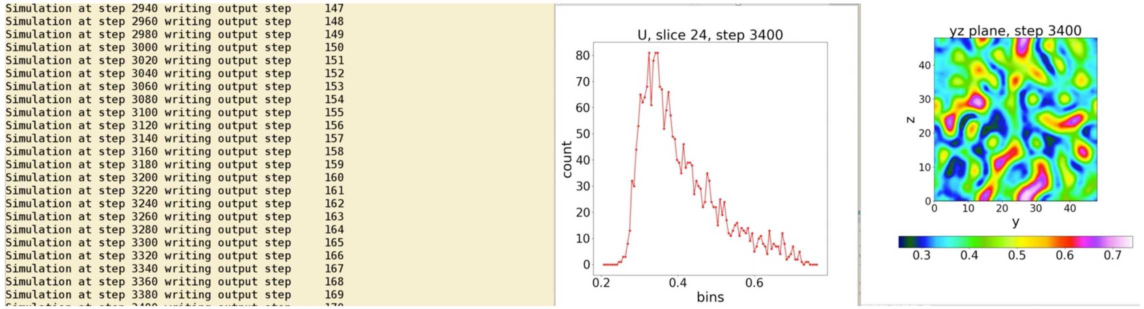
https://github.com/ornladios/ADIOS2

- Variables
  - Multi-dimensional, typed, distributed arrays
  - Single values
    - Global: one process, or Local: one value per process
- Engines
  - Filesystem
  - Staging, inline
  - WAN

GOALS

- Highly scalable (processors, variables, timesteps, consumers, producers)
- Easy to program, easy to achieve high performance
- Extensible
- Well integrated into the mainstream analysis/visualization tools

Data Staging



- Who was it designed for?
  - Direct transfer between I/O producers and consumers
  - High performance data streaming over WAN (federated)
  - Application coupling (simulations, experiments, analysis)
    - Minimizing the ease and time for Near Real Time decisions

- **Research directions:** Optimizations to allow for online processing
  - Allow data to be progressively consumed
  - Adaptive data retrieval (queries, in-transit filtering)
  - Using AI to autotune the prioritization and streaming of data
  - Learning and updating models on the fly for auto-tuning transfers/analysis at runtime

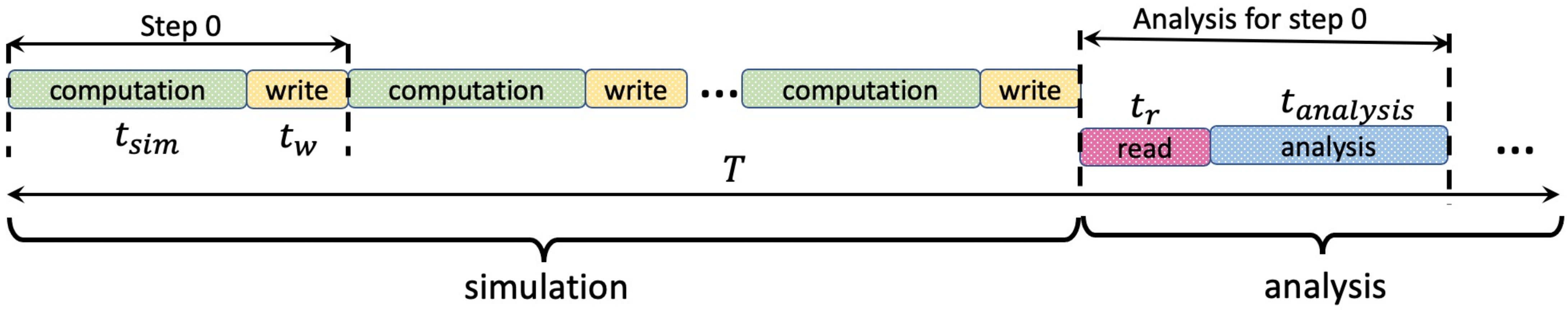
https://users.nccs.gov/~pnorbert/GrayScottInsitu.mp4

Table of contents

- Data management systems for coupled applications
- **Performance for I/O kernels**
- Performance for typical patterns on Summit
  - Embarrassingly parallel
  - Traditional HPC
  - Emerging applications
- Data management in AI applications

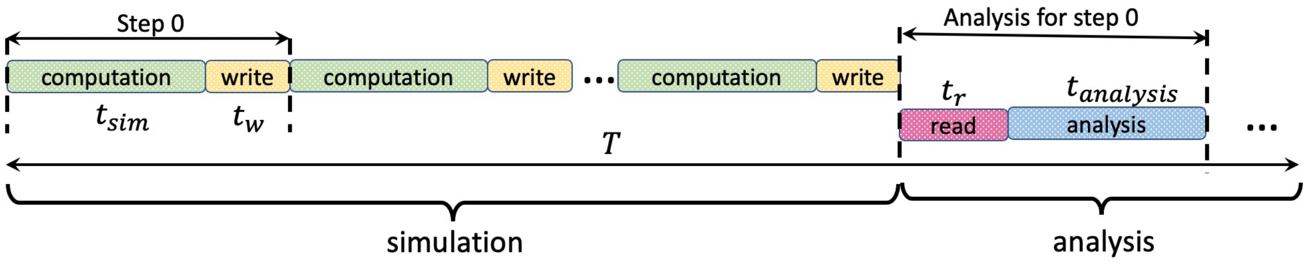
Ways of data transfer between coupled applications

- Data transfer through files

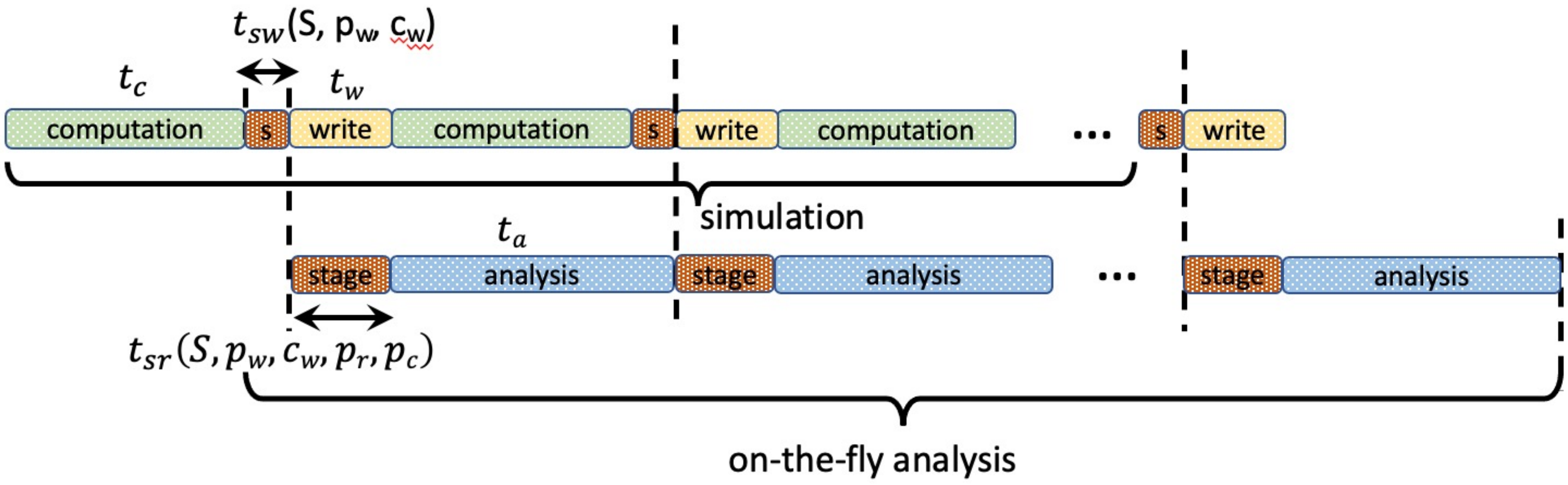


Ways of data transfer between coupled applications

- Data transfer through files

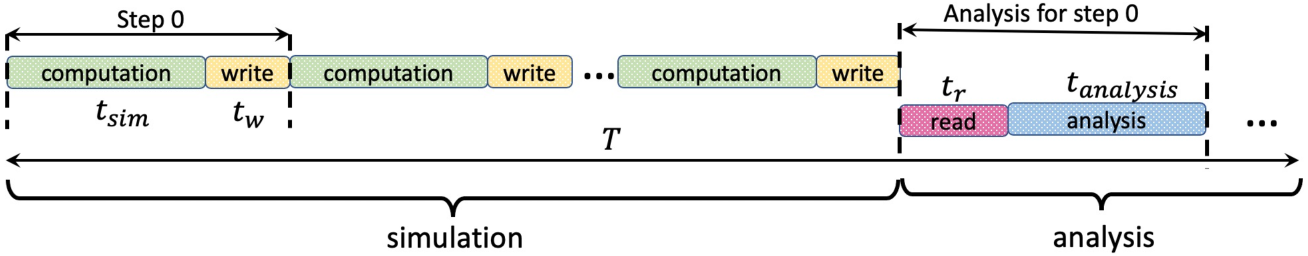


- Data staging

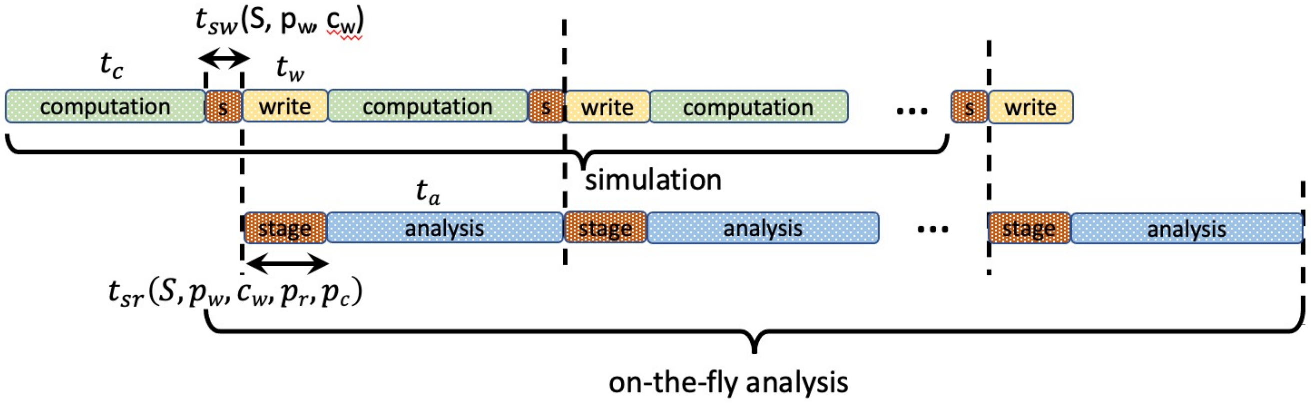


Ways of data transfer between coupled applications

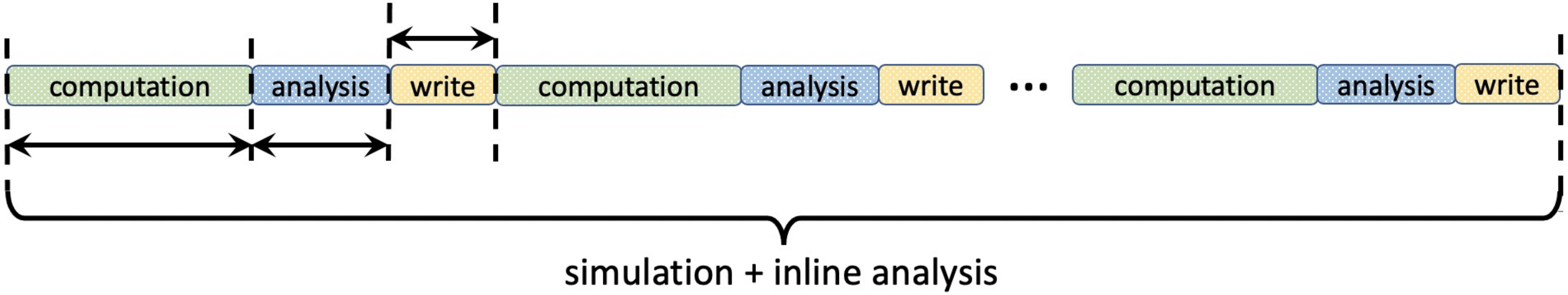
- Data transfer through files



- Data staging



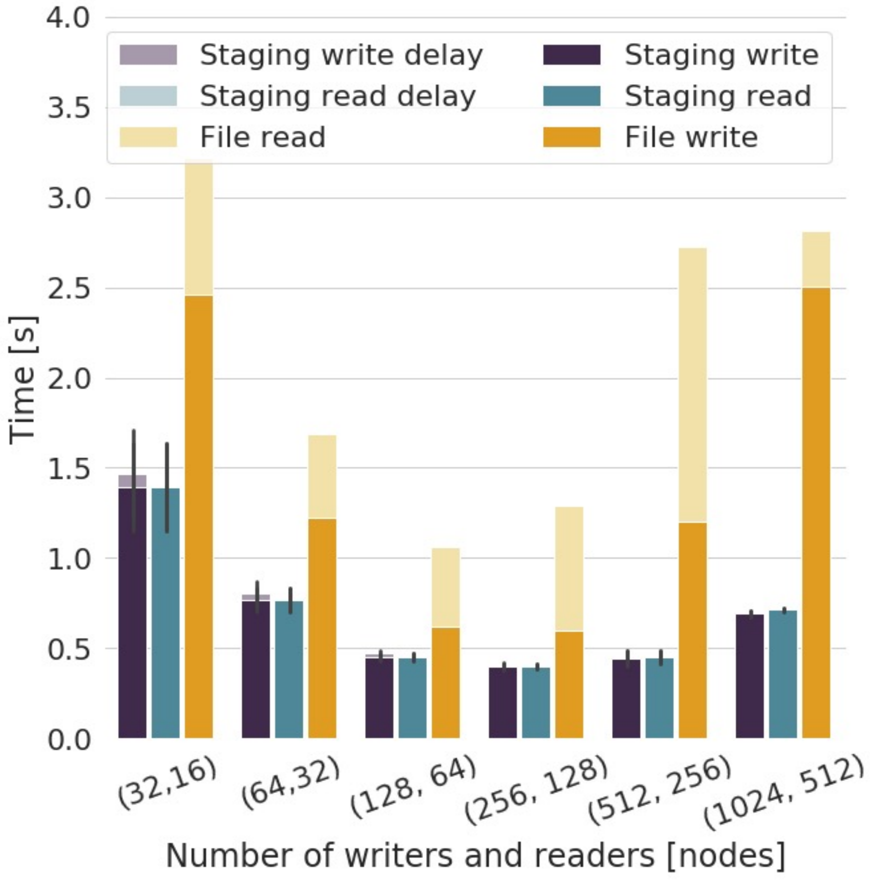
- Inline analysis



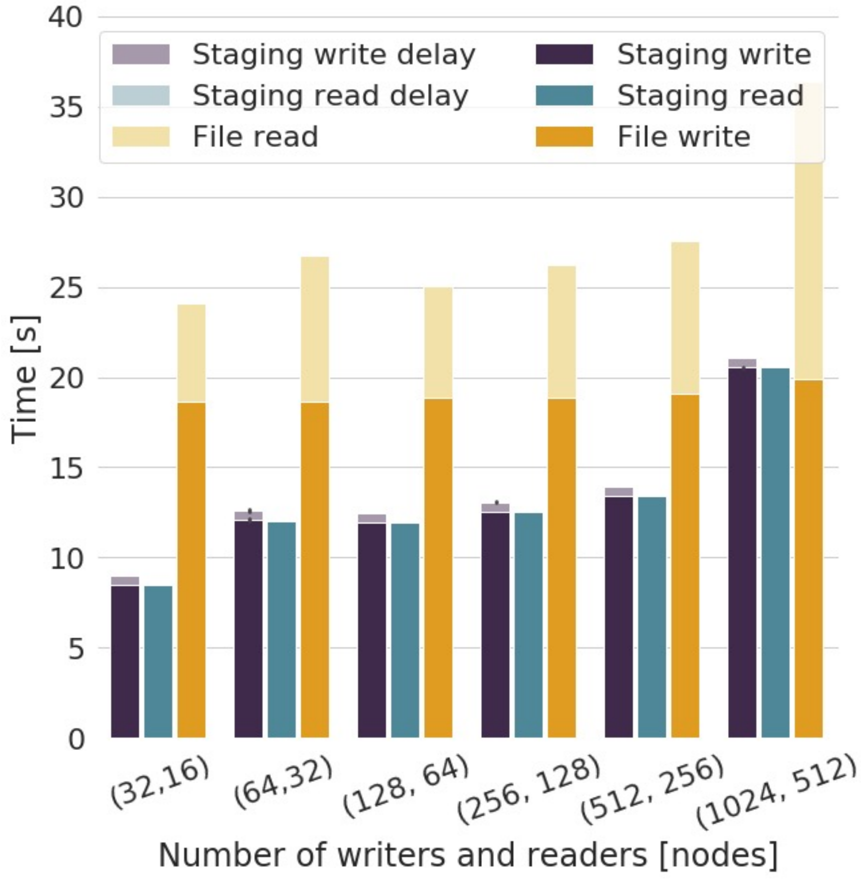
Performance

Data Producer	Data consumer

`simulation(N, p)`
`ADIOS.Put(N)`
	`ADIOS.Get(N)`
	`Prepare_data(N, p)`
	`analysis(N, p)`



Strong scaling



Weak scaling

Strong: total amount of data involved in streaming is kept constant (**100GB total I/O size**)
Weak: amount of data per writer is kept constant (**1 GB of data or 24 GB per node**)

Findings

- Staging algorithms achieve better I/O performance than using the filesystem
  - They sometimes require more node hours
  - Node hours: amount of processing units * allocation time
- Performance is influenced by where to place the writing phase within a staging algorithm
  - In the data producer or data consumer
- Inline analysis works best for in situ visualization/analysis
  - When the data producer and data con-sumer use a 1:1 mapping
  and the data need not be redistributed among the consumers.

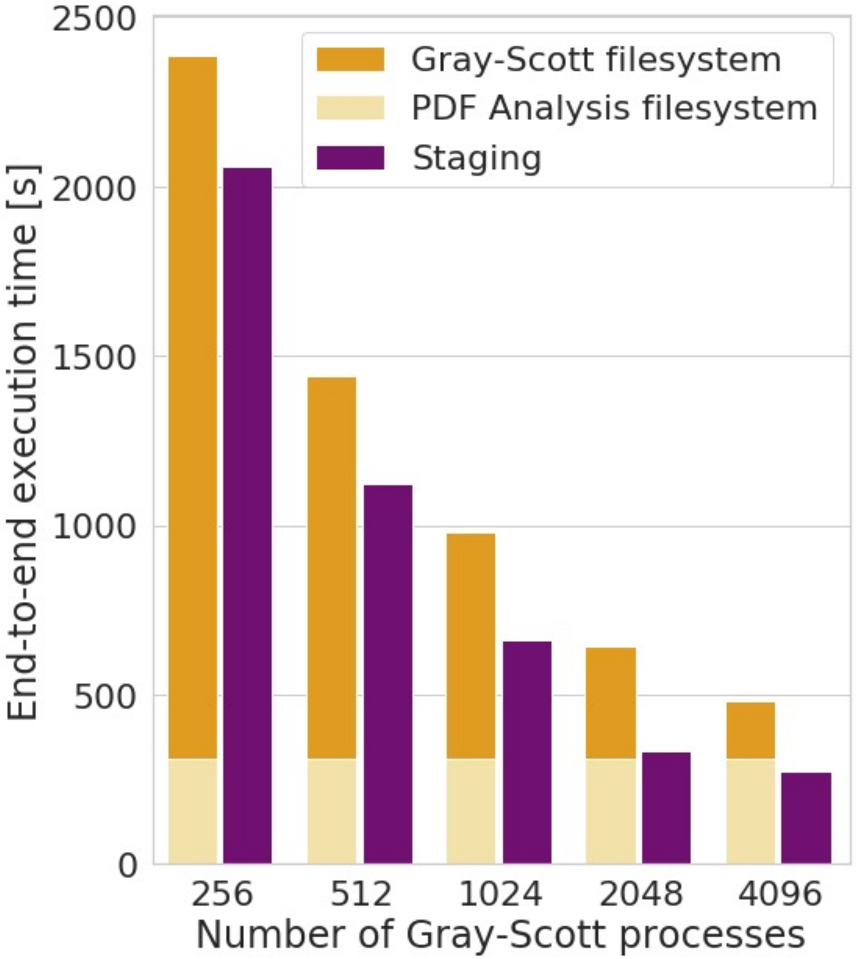
Table of contents

Staging patterns in applications on Summit

- Embarrassingly parallel applications
  - Code scales linearly with the number of processors
  - Monte Carlo simulations
  - **Testcase: the Gray-Scott reaction diffusion model coupled with two analysis codes as a test case**
- Traditional HPC applications
  - Loosely coupled applications that require synchronization between processes. Sometimes complex analysis / visualizations codes
  - **Testcase: XGC, a gyrokinetic particle simulation of edge plasma coupled with a visualization code**
- **New emerging applications**

Embarrassingly Parallel Applications

- Codes scale linearly with the number of processes
  - **For the sequential algorithm**, best performance is given by using as many processes as available
    - As long as the cost to write and read scales the same
  - **For streaming**, using math models can give the optimal ratio between number of producers to consumers



$$p_r = \frac{t^{p=1}_{analysis}}{\frac{t^{p=1}_{sim}}{p_w} + \frac{N_{IO}}{B_{sw}} - \frac{N_{IO}}{B_{sr}}}.$$

$N_{IO} = 50$GB

Summit B_{sw}=2.1 GB/s, B_{sr}=6 GB/s (bandwidth to NVME)

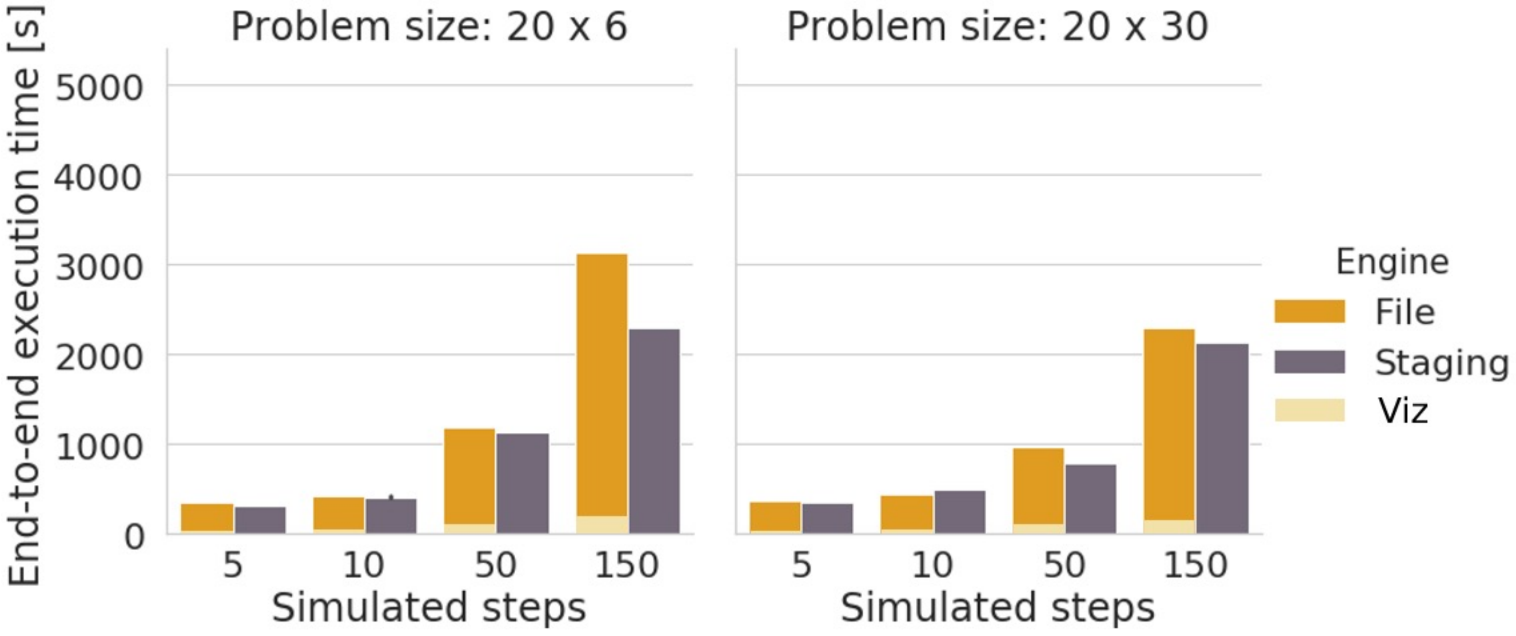
Optimal ratio: 24 PDF processes to 2048 Grey-Scott processes

OAK RIDGE National Laboratory

Traditional HPC applications

- XGC characteristics
  - Produces 149 GB for 20×6 and 890 GB for a 20×30 problem.
  - Processors defined by problem size (240, 1200) + 1 core for visualization

Trade-off between time to solution and cost



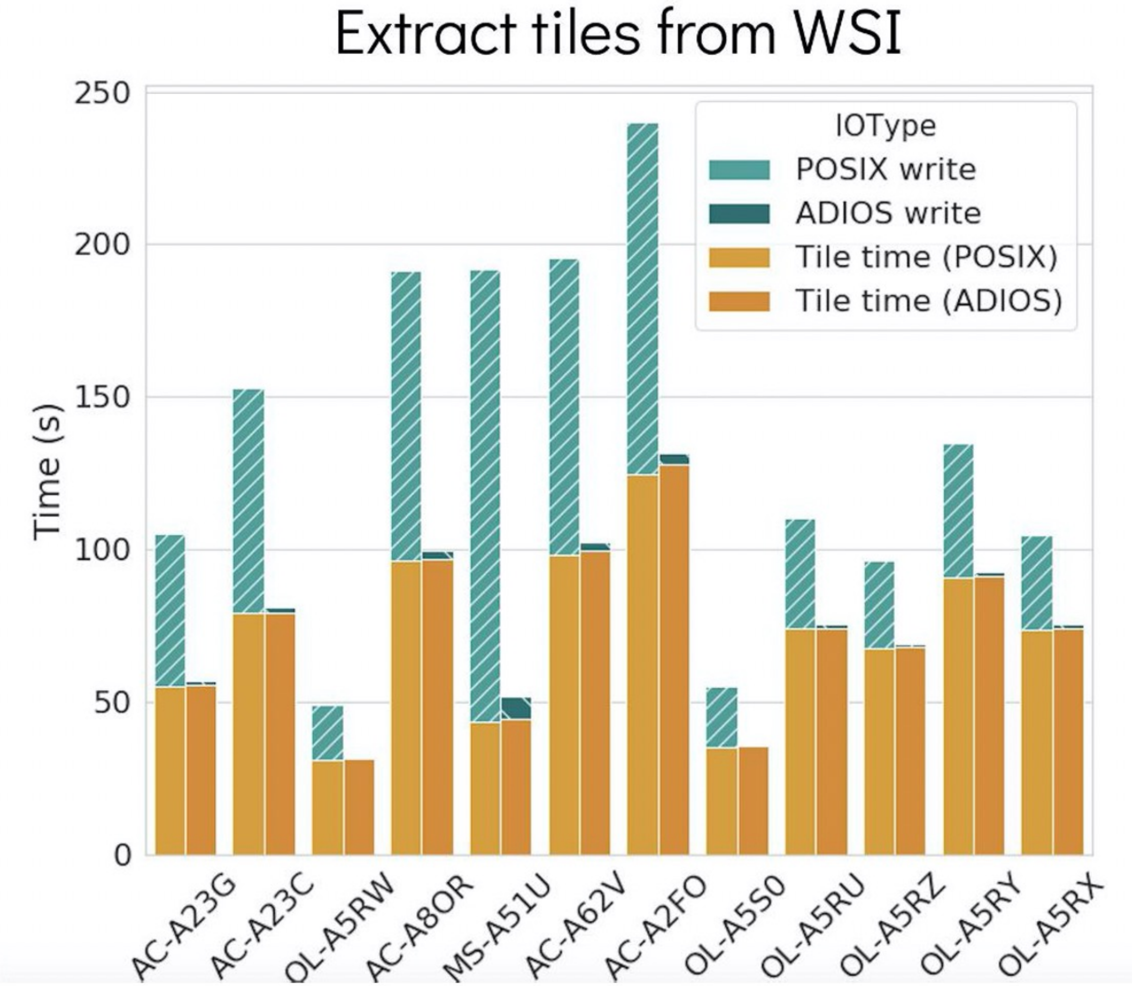
Cost_staging = (240 + k) * time_staging
Cost_file = 240 * time_xgc + k * time_viz

Problem 20 x 6 150 steps
Viz time ~3 min to 45 min of XGC

Viz Cores	Cost staging	Cost file
1	134.22	193.37
24	147.03	194.33
120	200.5	198.33

Emerging applications

- New generation applications
  - Replace computation kernels with AI
  - ML workflows that require training phases
- Focus on Medical imaging processing
- First step: optimize their I/O
  - ADIOS variables instead of files



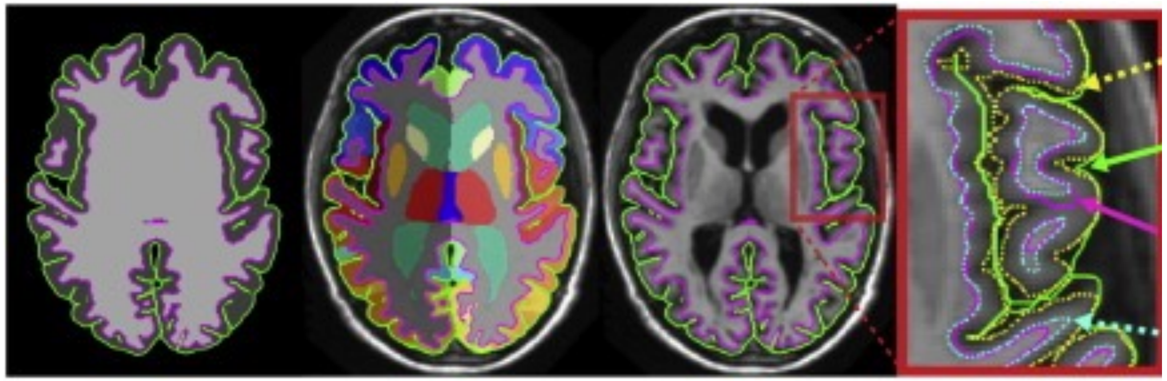
Whole slide image id

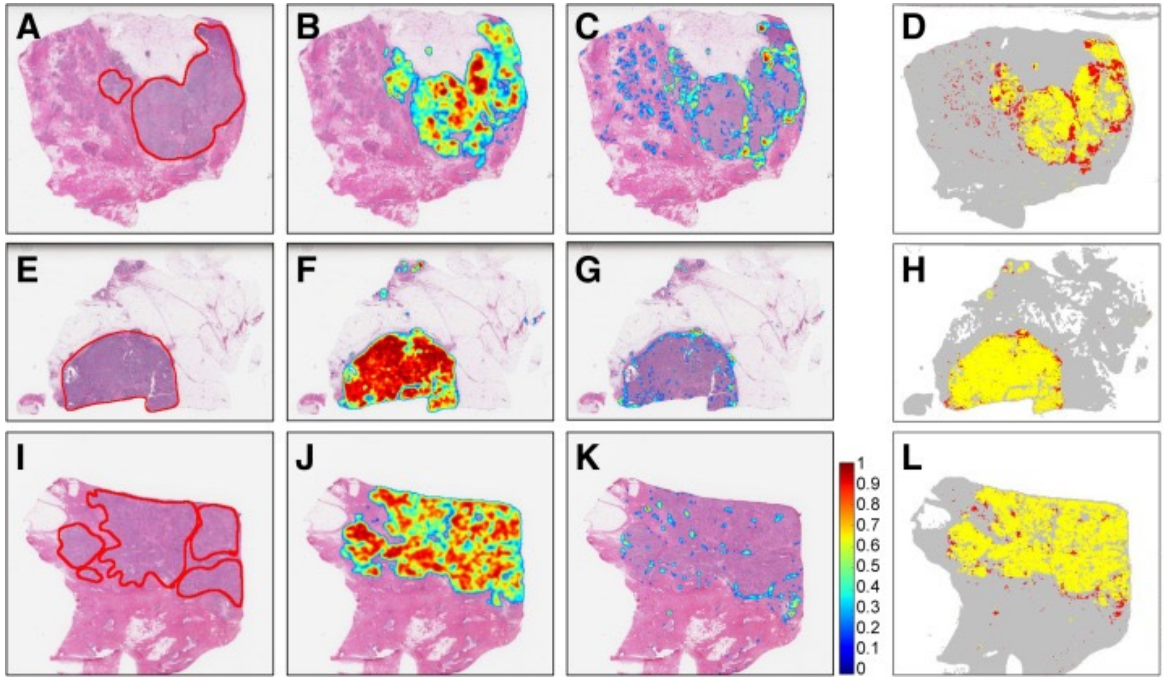
Table of contents

- Data management systems for coupled applications
- Performance for I/O kernels
- Performance for typical patterns on Summit
  - Embarrassingly parallel
  - Traditional HPC
  - Emerging applications
- **Data management in AI applications**

Medical image processing

- Rely on ML approaches for a multitude of analysis tasks
  - Multiple types of samples
  - Multiple types of AI methods
  - Exploratory studies
- Data intensive
  - A single whole slide image corresponding to a single prostate biopsy core can easily occupy 10 GB of space at 40x magnification
  - Vanderbilt MASI lab runs over 10,000 studies per week
- Codes are in continue change





OAK RIDGE National Laboratory

Background

- Multiple types of image processing
  - X-ray radiography, computed tomography (CT), MR imaging (MRI), ultrasound, digital pathology, etc
  - New modalities are being routinely invented (e.g. spectral CT)
  - The pixel or voxel resolution becomes higher
    - CT and MRI has reached the submillimeter level

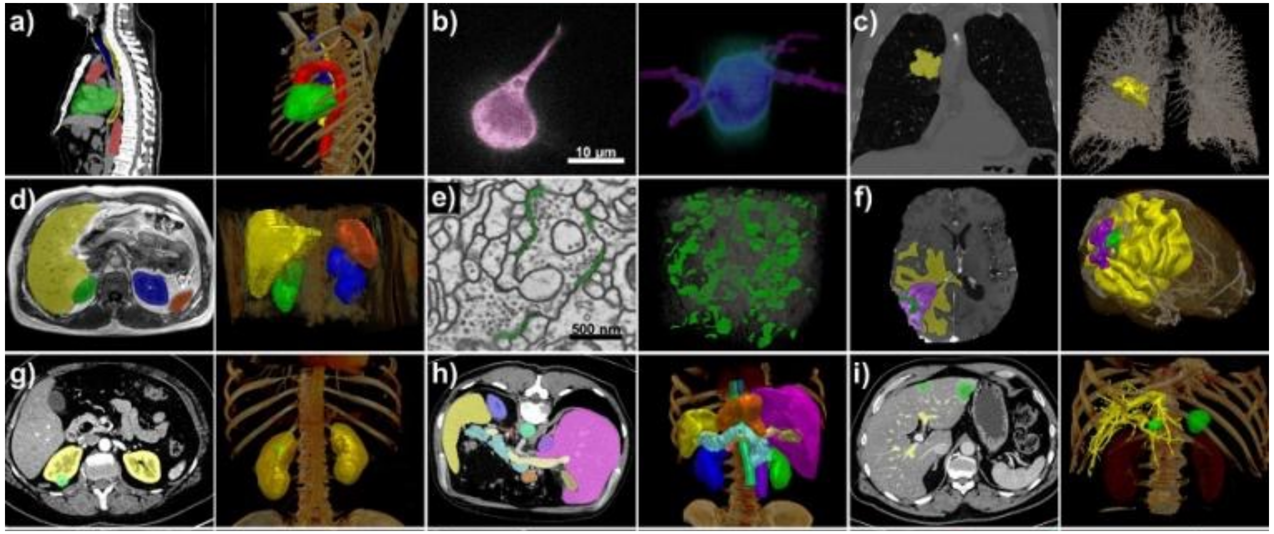
- Labels are sparse and noisy
  - Different tasks require different forms of annotation
  - The disease patterns in medical images are numerous
  - The ratio between positive and negative samples is uneven

Background

- Different types of tasks using ML
  - **Scope**: detection of pathological findings, quantification of disease extent, characterization of pathologies (e.g., into benign versus malignant), decision support software tools
  - Medical image reconstruction / enhancement
  - Segmentation
  - Detection / Diagnosis

Shift in behavior compared to classic scientific HPC applications

Different applications of medical image segmentation



https://tectales.com/ai/deep-learning-based-image-segmentation.html

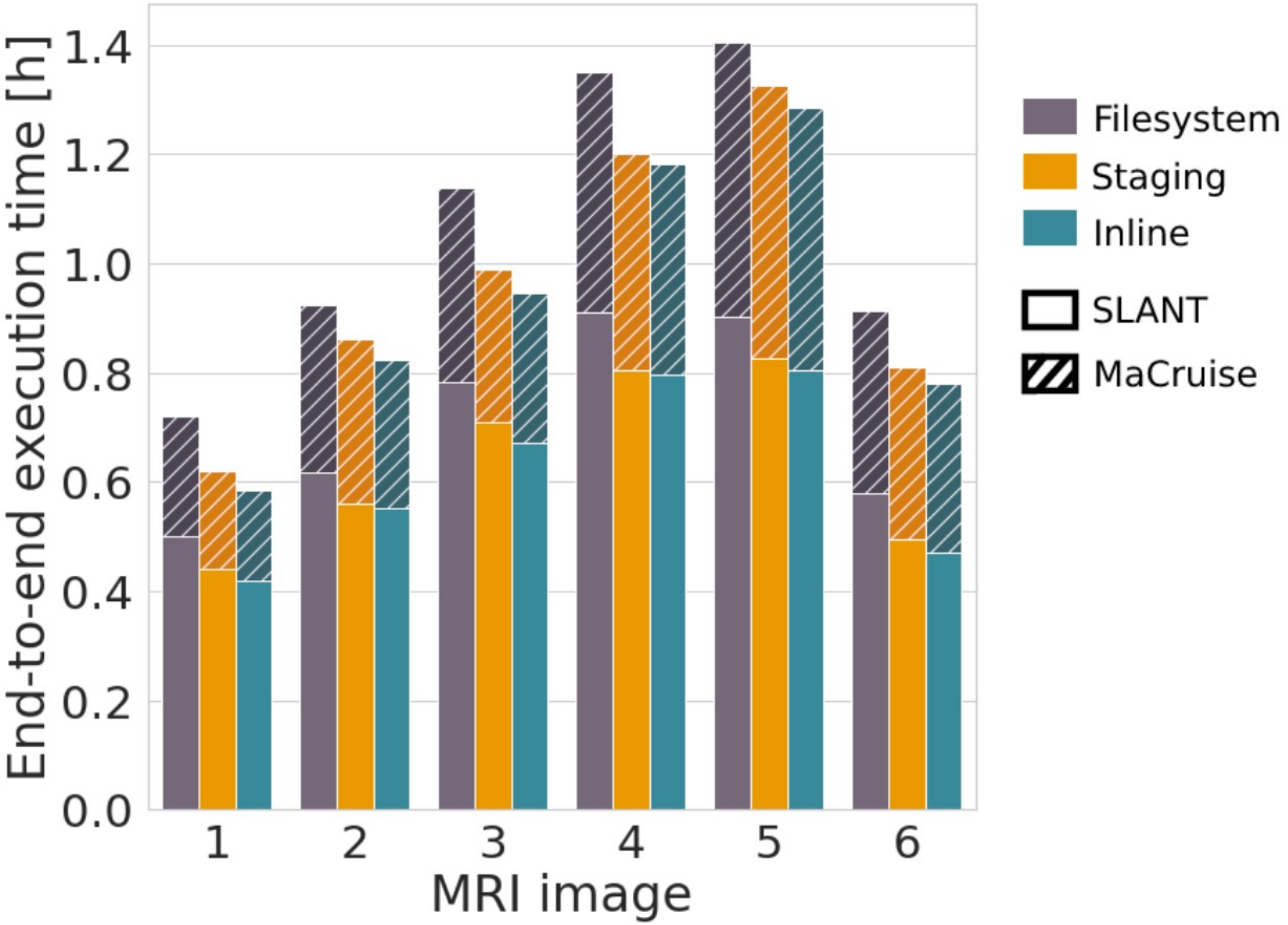
Neuroscience applications

- Multi code coupling
  - Vanderbilt University
  - Medical-image Analysis and Statistical Interpretation (MASI) lab
  - **SLANT**
    - Deep Whole Brain High Resolution Segmentation
    - Input data: MRI image
  - **MaCruise**
    - Deep learning models for cortical reconstruction based on an MRI image and the identified segments

Yuankai Huo et al "3D whole brain segmentation using spatially localized atlas network tiles" NeuroImage 2019
https://github.com/MASILab/SLANTbrainSeg

Data management

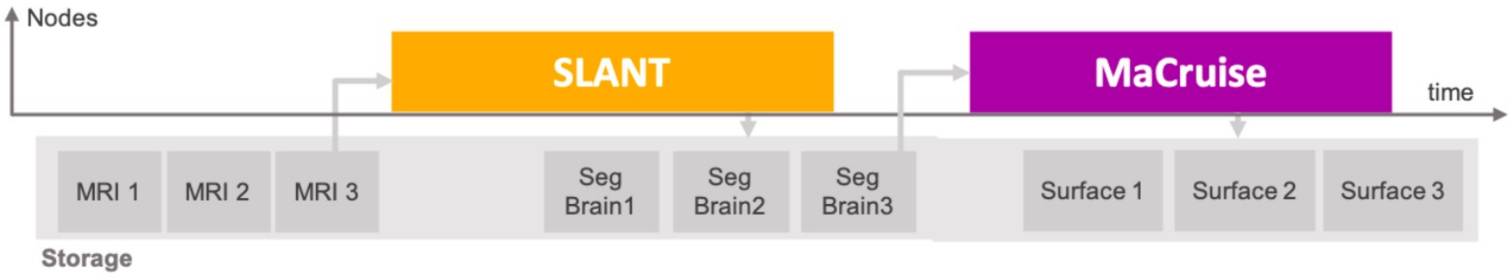
- One node jobs
  - ML using GPUs
- Experiments on 6 MRIs
- ADIOS
  - Used for streaming and inline
- MaCruise needs to wait for SLANT to finish



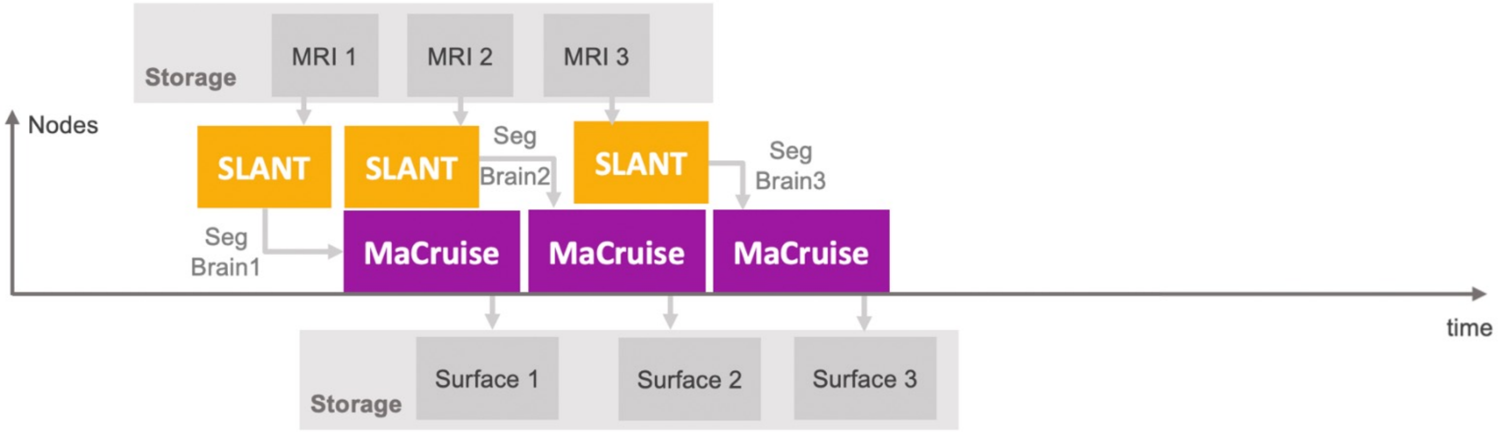
Results when using different ways of transferring the file between SLANT and MaCruise

Automation

- Bulk execution
  - Uses filesystem, one node
- Parallel execution
  - Each MRI in parallel, 2 nodes
- Pipeline execution
  - 2 nodes



((a)) Bulk execution

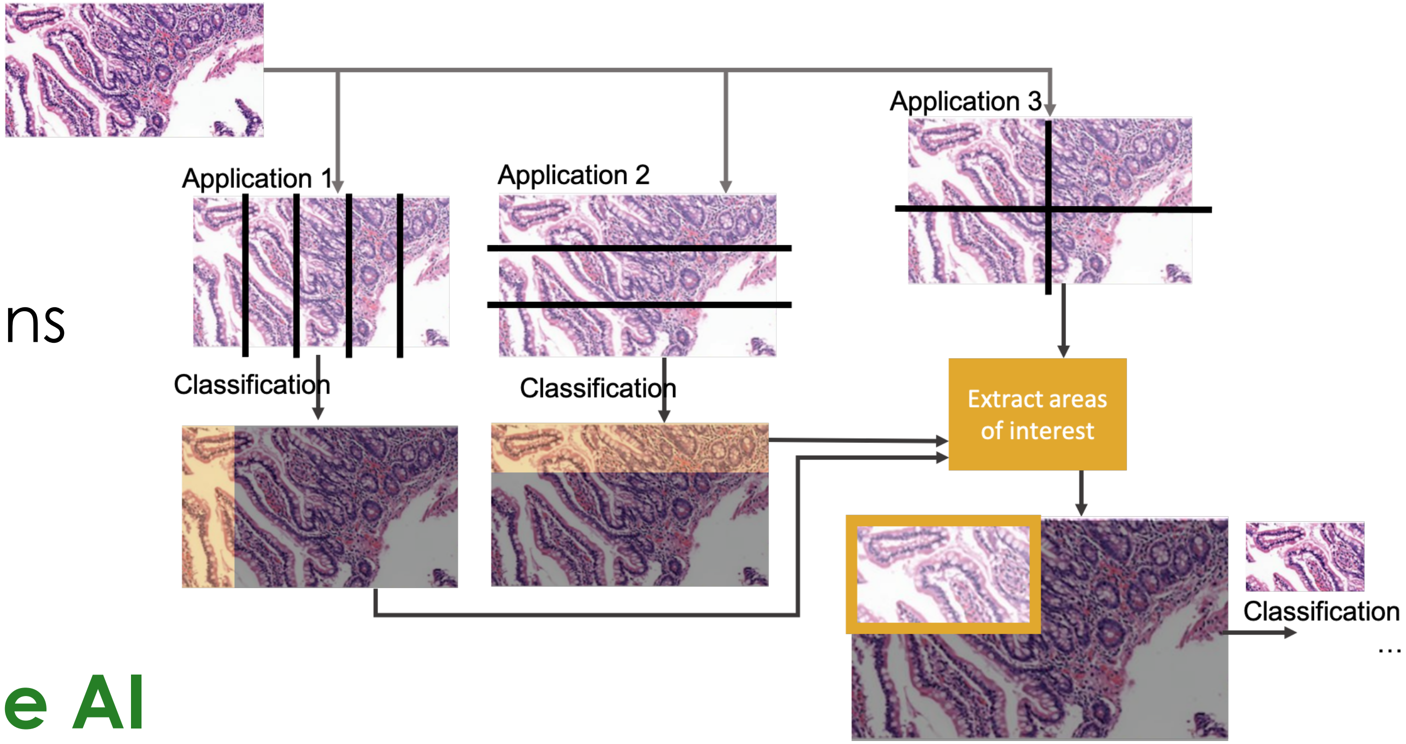


((b)) Pipeline execution

	SLANT (h)	MaCruise (h)	Total (h)	Cost
Bulk FS	4.29	2.16	**6.45**	6.45
Parallel FS	2.83	1.41	**4.24**	8.48
Pipeline	3.83	2.11	**3.83**	7.66

Towards automation

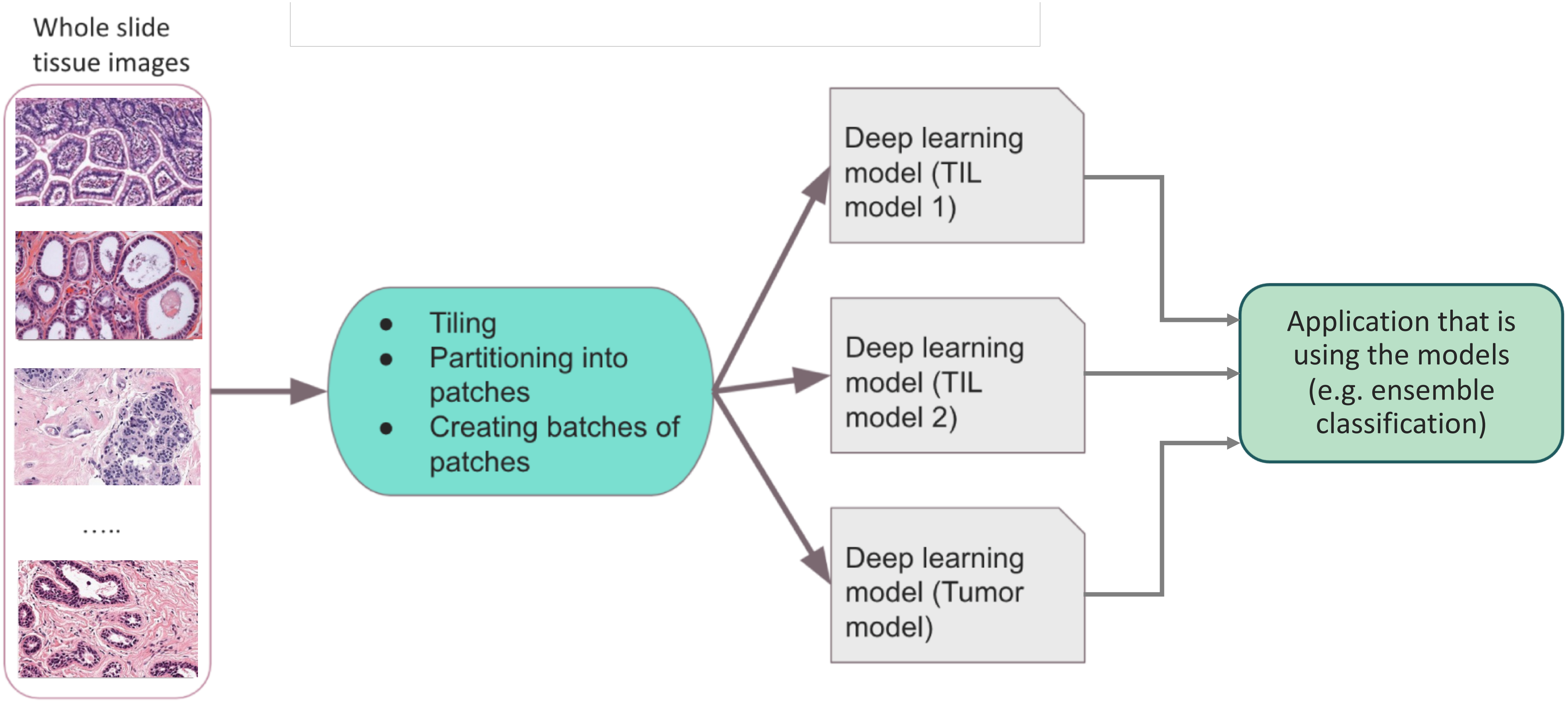
- Data needs to be moved
  - From storage to the AI applications
  - Between different tasks
  - Between different applications



- **Goal: Separate the data management layer from the AI process**

What is next

Data management for Running prediction/inference with multiple models



Conclusions

- **Staging libraries**
  - Provide a solution to move the data on-the-fly from producers to consumers transparently and efficiently
  - Allow for visualization / analysis in near real time
  - If used correctly could reduce the cost

- **Automation is key for emerging applications**
  - Streaming is a necessity
  - First step towards more complex data management solutions
  - Allows flexibility in model management

Q&A

- Thank you

Ana Gainaru
gainarua@ornl.gov