ADIOS 2: A Framework for Extreme Scale I/O and In Situ Processing

Norbert Podhorszki
Workflow Systems Group
Computer Science and Mathematics Division
Oak Ridge National Laboratory

Collaborators: Apps, Workflow, Data Management, Reduction, Viz

Scott Klasky
Norbert Podhorszki
Qing Liu
Karsten Schwan
Jay Lofstead
Mark Ainsworth
C.S. Chang
Ana Gainaru
Hasan Abbasi
Rick Archibald
Chuck Atkins
Vicente Bolea
Phillipe Bonnet
Michael Bussmann
Jieyang Chen
Hank Childs

Jong Choi
Michael Churchill
Nathan Cummings
Shaun de Witt
Philip Davis
Ciprian Docan
Greg Eisenhauer
Stephane Ethier
Ian Foster
Dmitry Ganyushin
Kai Germaschewski
Berk Geveci
William Godoy
Qian Gong
Junmin Gu
Jon. Hollocombe

Kevin Huck
Axel Huebl
Toby James
Chen Jin
Mark Kim
Brad King
James Kress
S.H. Ku
Ralph Kube
Tahsin Kurc
Xin Liang
Zhihong Lin
Jeremy Logan
Thomas Maier
Kshitij Mehta
Ken Moreland

Todd Munson
Manish Parashar
Franz Pöschel
Dave Pugmire
Anand Rangarajan
Sanjay Ranka
Stefanie Reuter
Caitlin Ross
Nagiza Samatova
Ari Shoshani
Eric Suchyta
Fred Suter
Keichi Takahashi
William Tang
Roselyne Tchoua
Nick Thompson

Seiji Tsutsumi
Ozan Tugluk
Lipeng Wan
Ruonan Wang
Xinying Wang
Ben Whitney
Andreas Wicenec
Matthew Wolf
John Wu
Bing Xie
Fan Zhang
Fang Zheng

ADIOS Useful Information and Common tools

- ADIOS tutorial: https://tinyurl.com/adios-sc2023
- ADIOS documentation: https://adios2.readthedocs.io/en/latest/index.html
- ADIOS source code: https://github.com/ornladios/ADIOS2
  - Written in C++, wrappers for Fortran, Python, Matlab, C
  - Contains command-line utilities (bpls, adios2_reorganize ..)
  - Examples in C++, Fortran and Python
- Online help:
  - ADIOS2 GitHub Issues: https://github.com/ornladios/ADIOS2/issues

Outline

- Introduction
- ADIOS2 concepts (and python read API)
- Application storage I/O success stories
- GPU-Aware IO
- Data reduction with MGARD
- In situ processing
- Application in situ success stories
- Visualization schema and ParaView
- Building ADIOS2 on Frontier

Motivation

Every application has a maximum frequency by which data (timesteps) would need to be written, and maximum amount of data (variables) at every step, to calculate everything the scientist wants with best possible accuracy.

Limitations:

- Writing cost of this much data
- Storage cost of this much data
- Inability to process all that much data

Copying strategies

- Write less frequently (decimation) – loss of accuracy
- Write less amount of data per timestep - missing data
- Incorporate extra calculations for known Quantities of Interest and write those instead (inline in situ data reduction) – slower execution time, scalability issues
- Lossy compression – losing control of accuracy

Motivation

Our copying strategies

- Write fast, Read fast
- Lossy compression
    - with user control of accuracy
    - on GPU to do it fast
- In situ analysis
    - Add extra calculations for known Quantities of Interest asynchronously on extra nodes and write those instead (in transit in situ data reduction)

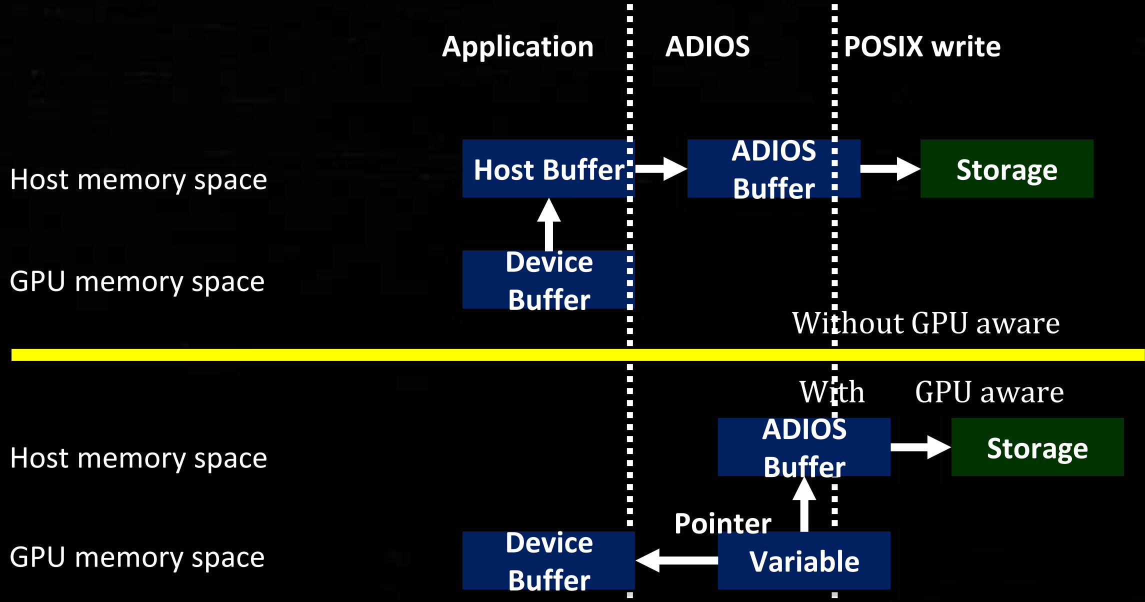
ADIOS: high-performance publisher/subscriber I/O framework:

Vision

- Create an easy-to-use, high performance I/O abstraction to allow for on-line/off-line memory/file data subscription service
- Create a sustainable solution to work with multi-tier storage and memory systems for self-describing data-streams

Details

- Declarative, publish/subscribe API separated from the I/O strategy
- Multiple implementations (engines) provide functionality and performance
- Rigorous testing ensures portability
- GPU-aware to reduce data movement
- https://github.com/ornladios/ADIOS2



Godoy, W. F., Klasky, S. ,et al. (2020). ADIOS-2: The adaptable input output system. a framework for high-performance data management. SoftwareX, 12, 100561.

ADIOS Concepts

Self-describing Scientific Data

```
double   BOUT_VERSION scalar = 5.2
double   Bxy          {68, 20} = 1 / 1
string   Bxy/cell_location attr   = "CELL_CENTRE"
string   Bxy/direction_y   attr   = "Standard"
string   Bxy/direction_z   attr   = "Average"
string   Bxy/source        attr   = "Coordinates"
double   G1           {68, 20} = 0 / 0
double   G2           {68, 20} = 0 / 0
double   G3           {68, 20} = 0 / 0
double   J            {68, 20} = 1 / 1
int32_t  MXG          scalar = 2
int32_t  iteration    143*scalar = -1 / 141
 …
```

```
double   dx           {68, 20} = 0.2 / 0.2
double   dy           {68, 20} = 1 / 1
double   dz           {68, 20} = 0.2 / 0.2
double   g11          {68, 20} = 1 / 1
int32_t  nx           scalar = 68
int32_t  ny           scalar = 16
int32_t  nz           scalar = 64
double   phi          143*{68, 20, 64} =
                      -0.139167 / 0.0899946
string   run_id       scalar =
      "cfc9cd3d-3ec1-4238-8fa0-f75f97a9c949"
double   t            143*scalar = 0 / 142
```

```
double   n            143*{68, 20, 64} =
                      -0.185305 / 0.0961174
```

143 output steps of a 3D array of double type and 68x20x64 dimensions, named n
global min = -3.76192 max = 4.05582

Self-describing Scientific Data

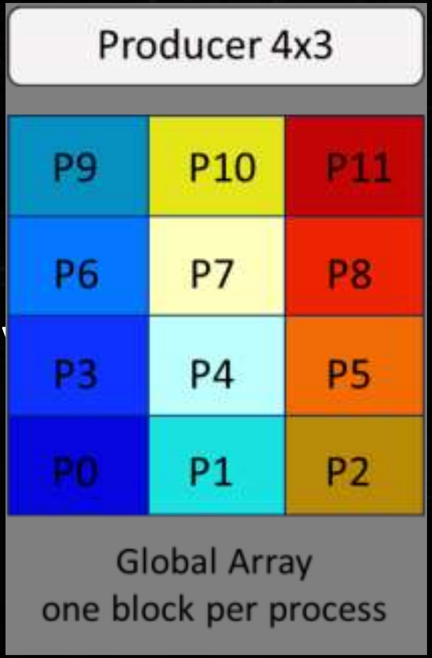
```
 double   n                    143*{68, 20, 64} = -3.76192 / 4.05582
…
     step 142:
          block 0: [ 0:17,  0: 9,  0:63] = -2.06509 / 2.97009
          block 1: [18:33,  0: 9,  0:63] = -0.337289 / 1.85048
          block 2: [34:49,  0: 9,  0:63] = -1.71457 / 0.40956
          block 3: [50:67,  0: 9,  0:63] = -3.25034 / 2.24025
          block 4: [ 0:17, 10:19,  0:63] = -2.06509 / 2.97009
          block 5: [18:33, 10:19,  0:63] = -0.405136 / 1.66294
          block 6: [34:49, 10:19,  0:63] = -1.70201 / 0.395594
          block 7: [50:67, 10:19,  0:63] = -3.25034 / 2.24025
```

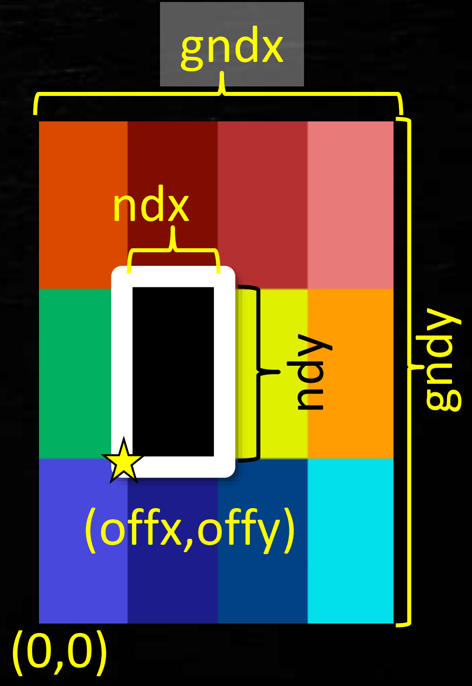
Data is stored in 8 blocks, which usually means
8 MPI tasks, each writing a piece.
Obviously, a toy example ;-)

BOUT++ hasegawa-wakatani-3d example, partial list of variables

Global Array: data produced by multiple processes

- N-dimensional array
  - **Shape**
- Has a type (int32, double, etc.)
  - **Type**
- Blocks of data are written into the array
  - **Start** (offset)
  - **Count** (size of block)





Shape = {gndx, gndy}

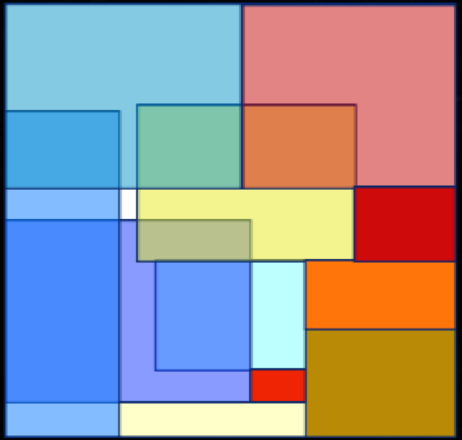
Start = {offx, offy}

Count = {ndx, ndy}

Global Array: data produced by multiple processes

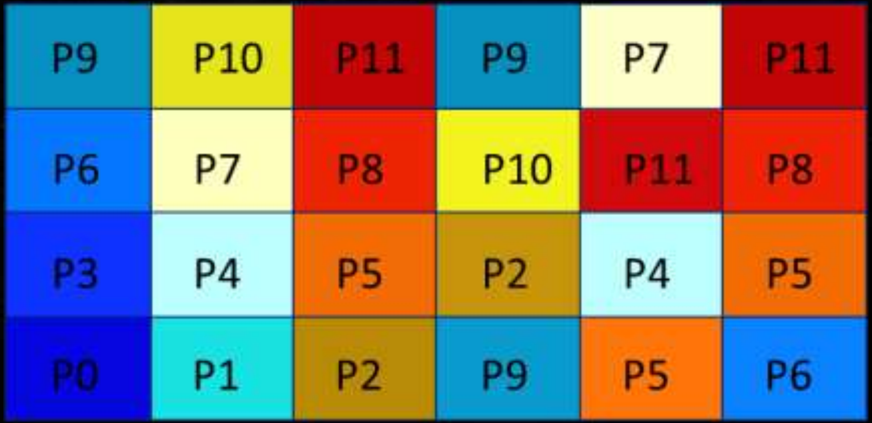
- These are valid global arrays
  - One process can contribute more than one block
  - Some process may not write anything at all
  - Holes can be left in the global array
  - Overlapping of blocks is allowed

Global Array with overlapping blocks

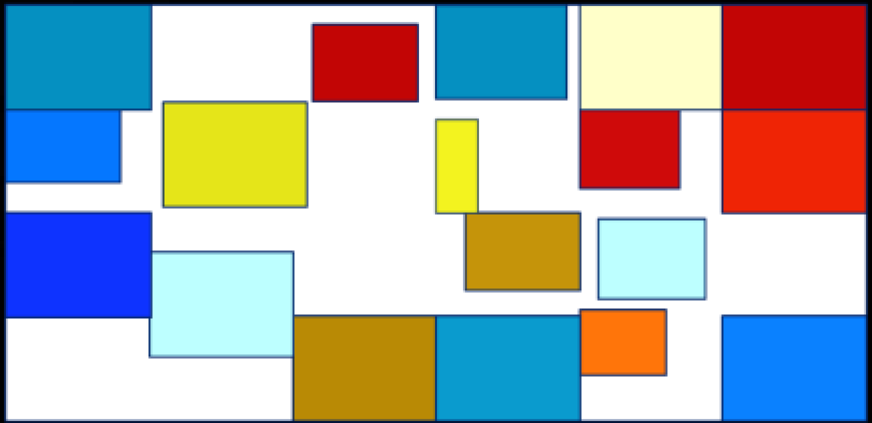


An overlapped cell has "one of the" values

12 Producers
multiple blocks per process



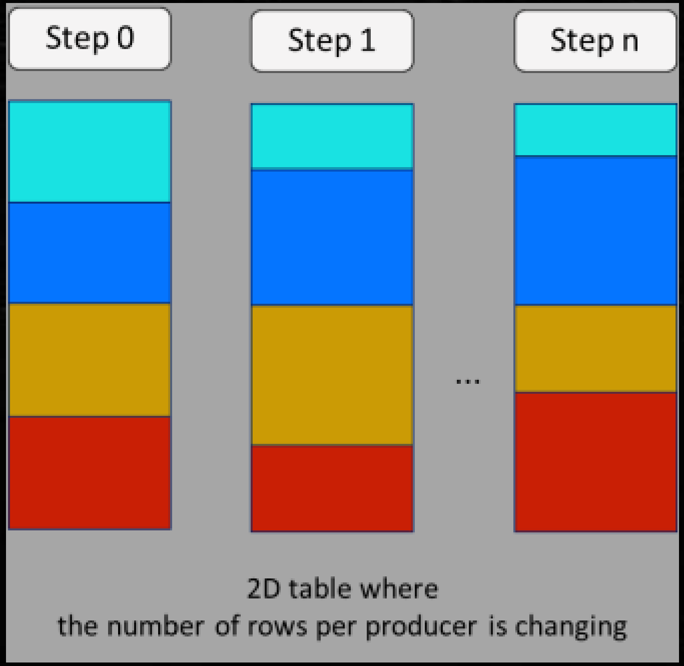
Global Array sparsely filled out

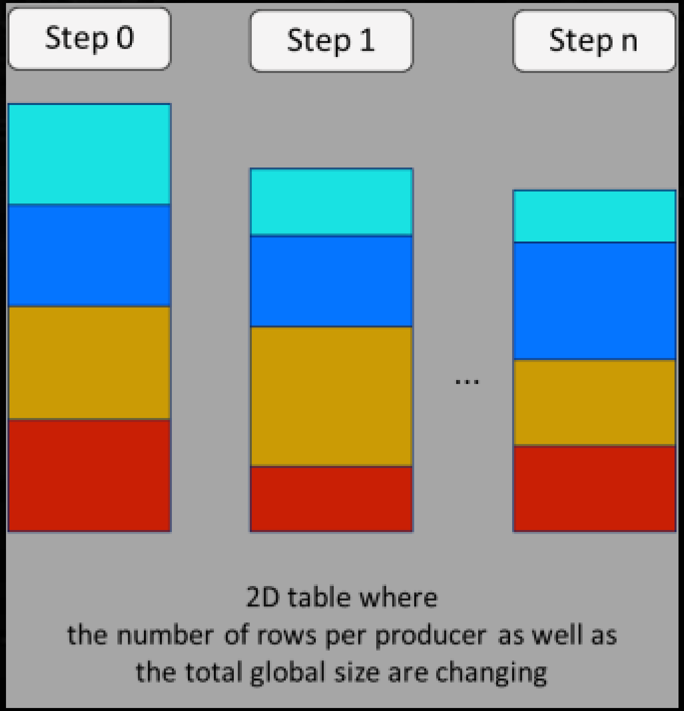


Read returns "nothing" for those cells.

Global Array: Shape and decomposition can change

- Internal decomposition of global array can change in next step
- Global size of array can change in next step
  - Only can read step-by-step though
  - Not multiple steps in a single read request





- JoinedArray is convenient to output tables in parallel without calculating offsets in global space
  - e.g. particles, atom tables
  - See docs, Basics/Internal Components/Shapes

https://adios2.readthedocs.io/en/latest/components/components.html#shapes

ADIOS basic concepts

- Step
  - Producer outputs a set of variables and attributes at once
    - This is an **ADIOS Step**
  - Producer iterates over computation and output steps
- Producer outputs multiple steps of data
  - e.g. into multiple, separate files, or into a single file
  - e.g. steps are transferred over network
- Consumer processes step(s) of data
  - e.g. one by one, as they arrive
  - e.g. all at once, reading everything from a file
    - post-processing only, not able to process in situ this way

Step is a Transaction between producer and its consumers

ADIOS Steps: Rules and constraints

- Step is not necessarily tied to the application timesteps
  - a Step can be constructed over time
- Entire content of a Step is either completely written or not at all
- A new Step can be very different from the previous step
  - may contain a completely different set of variables
  - array sizes can change
  - array decomposition can change
- Consumer is guaranteed to have access to entire content of Step as long as it wants it
- Entire content of a Step must fit into the producer's memory as a copy
  - well, there are ways around this for storage I/O

ADIOS Python API

```
examples/
            hello/helloWorld/hello-world.py
            examples/hello/bpReader/bpReaderHeatMap2D.py

            examples/hello/sstWriter/sstWriter.py
            examples/hello/sstReader/sstReader.py

            simulations/gray-scott-struct/plot/gsplot.py
```

Python common

Sequential python script:

```
import numpy
import adios2

T = numpy.array(...)
```

Parallel python with MPI:

```
from mpi4py import MPI
import numpy
import adios2

T = numpy.array(...)
```

Python Read API: Open/close a file/stream

```
adios2.Stream(path, mode [, comm])
adios2.FileReader(path [, comm])
```

mode: "r", "w", "rra"
mode here is "rra"

Examples:

```
fr = adios2.Stream("data.bp", "r")
fr = adios2.FileReader("data.bp", comm)
```

```
ad = adios2.Adios("adios2.xml", comm)
io = ad.declare_io("myIO")
fr = adios2.Stream(io, "data.bp", "r")
```

Using external XML configuration

```
fr.close()
```

Python Read API: List variables

```
vars_info = fr.available_variables()

for name, info in vars_info.items():
    print("variable_name: " + name)
    for key, value in info.items():
        print("\t" + key + ": " + value)
    print("\n")
```

```
variable_name: T
	Type: double
	AvailableStepsCount: 2
	Max: 200
	SingleValue: false
	Min: 0
	Shape: 10, 16

variable_name: dT
	Type: double
	AvailableStepsCount: 2
	Max: 1.83797
	SingleValue: false
	Min: -1.78584
	Shape: 10, 16
```

Oak Ridge National Laboratory

Python Read API: Read data from **file** -- Random access

```
fr.read(name[, start, count, blockid, step_selection])
```

Examples:

```
data = fr.read("T")
>>> data.shape
(10, 16)

data = fr.read("T", [0,0], [10,16])
>>> data.shape
(10, 16)

data = fr.read("T", [0,0], [10,16], step_selection=[0, 2])
>>> data.shape
(2, 10, 16)
```

variable_name: **T**
- Type: double
- AvailableStepsCount: **2**
- Max: 200
- SingleValue: false
- Min: 0
- Shape: **10, 16**

Only for "rra" mode (FileReader)

Python Read API: Read data from **file/stream**

```
fr.read(path[, start, count])
```

Examples:

```
with adios2.Stream("values.bp", "r") as fr:
    for _ in fr.steps():
      data = fr.read("T")
      print("Shape: ", data.shape)
...
Shape:  (10, 16)
Shape:  (10, 16)
```

variable_name: **T**
- Type: double
- AvailableStepsCount: **2**
- Max: 200
- SingleValue: false
- Min: 0
- Shape: **10, 16**

OAK RIDGE National Laboratory

Seismic Tomography Workflow (PBs of data/run)

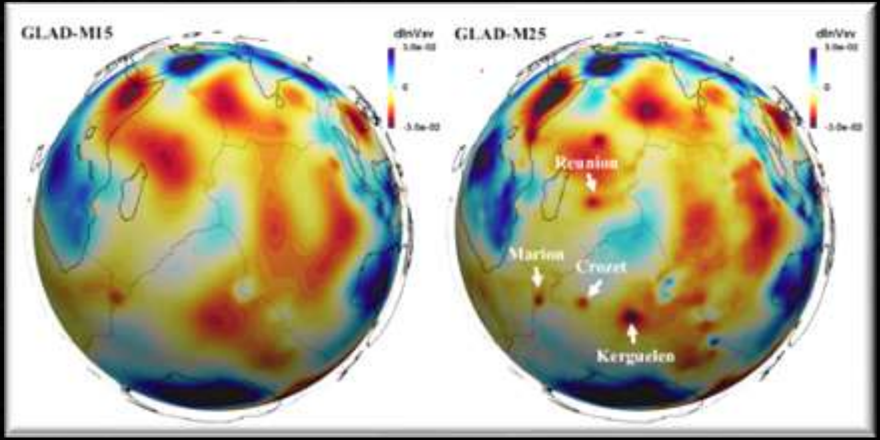
PI: Jeroen Tromp, Princeton

Scientific Achievement

- Most detailed **3-D model of Earth**’s interior showing the entire globe from the surface to the core–mantle boundary, a depth of 1,800 miles

Significance and Impact

- Updated (transversely isotropic) global seismic model GLAD-M25 where no approximations were used to simulate how seismic waves travel through the Earth. The data sizes required for processing are challenging even for leadership computer
- 7.5 PB of data is produced in a single workflow step
    - Which is fully processed later in another step

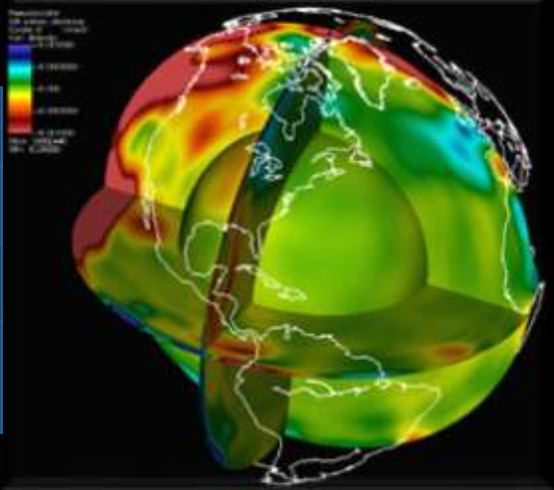


Map views at 250 km depth of vertically polarized shear wave speed perturbations in GLAD-M15 (2017) and GLAD-M25 (2020) in the Indian Ocean. New features have emerged in GLAD-M25, such as the Reunion, Marion, Kerguelen, Maldives, Seychelles, Cocos and Crozet hotspots.

Improvement by appending steps

- 3200 nodes ensemble run, 19200 GPUs
- 50 tasks at once
- 5.2 TB per task in 133 steps
- 260 TB total per 50 tasks
- 7.5 PB per 1500 tasks (total run)

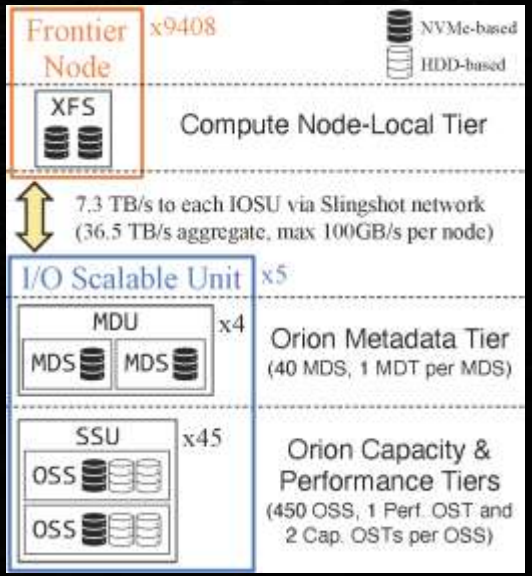
50 tasks, 133 steps, 3200 nodes	Time
No I/O	94s
BP3, one file per step	235s
BP4 one dataset per job 133x reduction in # of files	156s

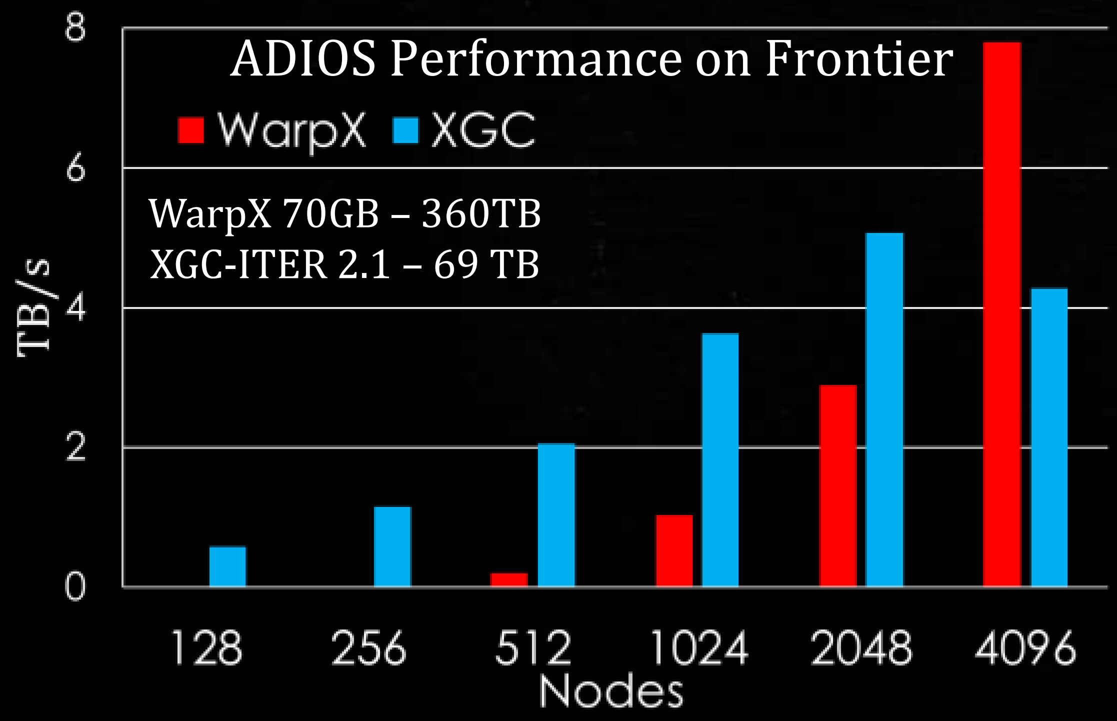


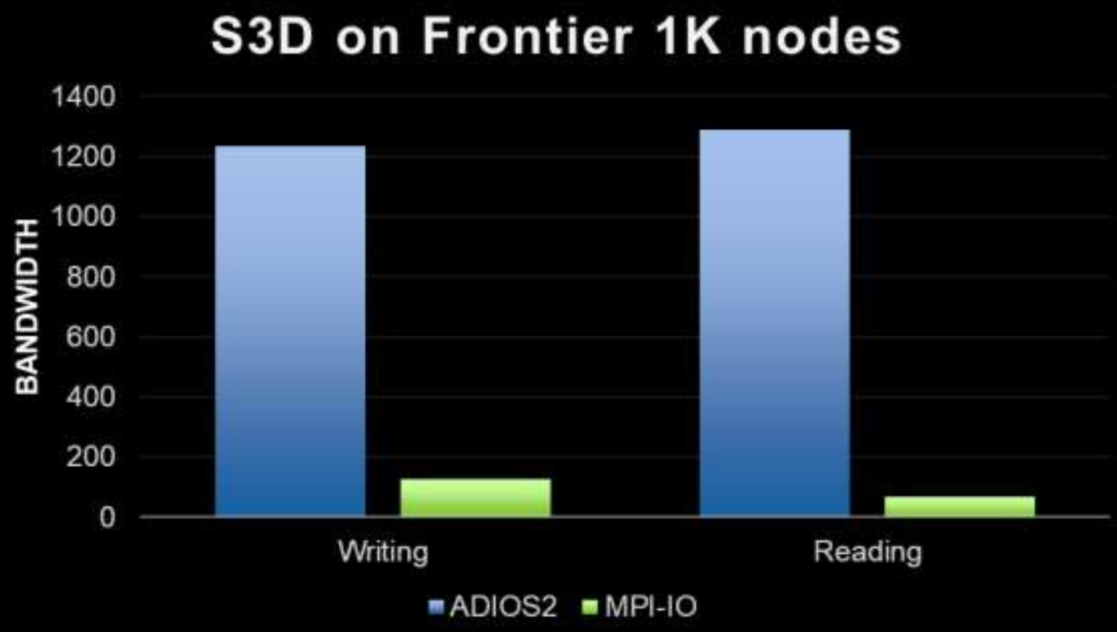
Lei, Wenjie, et al. "Global adjoint tomography—model GLAD-M25." Geophysical Journal International 223.1 (2020): 1-21.

XGC, WarpX, S3D on Frontier

Tier	Capacity (PB)	Read BW (TB/s)	Write BW (TB/s)
Node-Local	33	75	38
Metadata	10	0.8	.5
Performance	11.5	10	10
Capacity	679	5.5	4.6





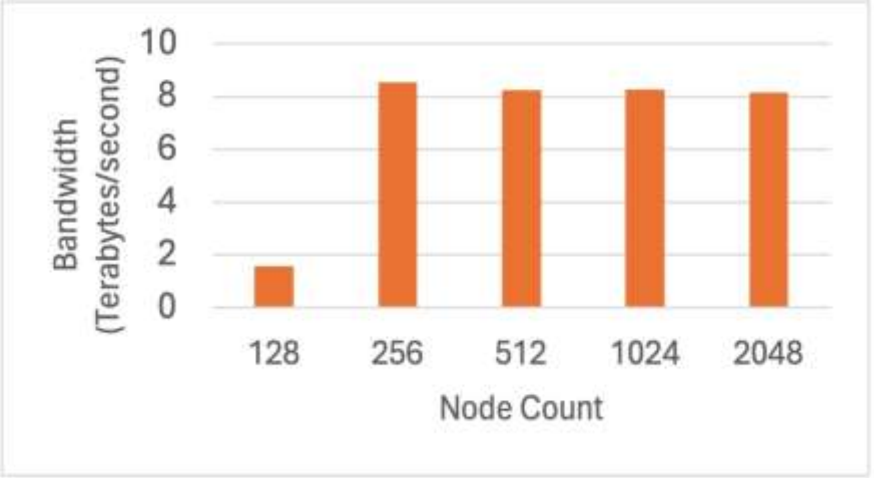


OAK RIDGE National Laboratory

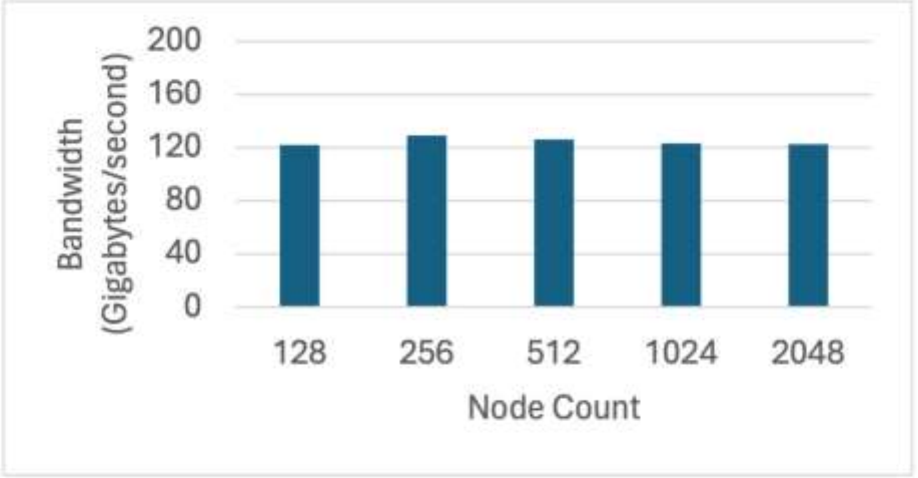
Managing large data and I/O for HydraGNN

- HydraGNN is a graph convolutional neural network developed at ORNL
- Part of AI LDRD for predicting molecular properties
- Uses ADIOS for efficient storage and retrieval of a large volume of training data
- Recent run used over 154 million molecules stored in 5 ADIOS datasets (5+ Terabytes total)
- More than 8 Terabytes/sec obtained during the parallel read step

Parallel reading



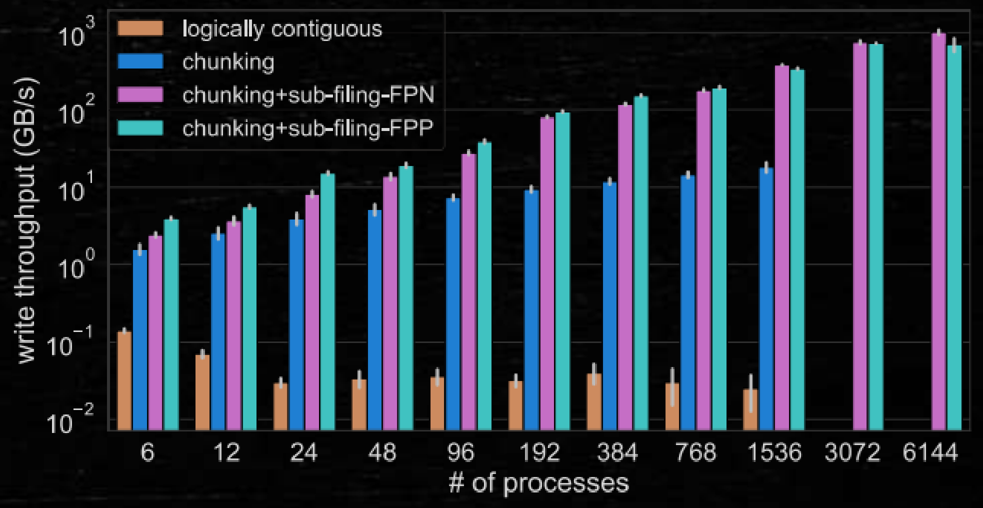
Aggregate bandwidth

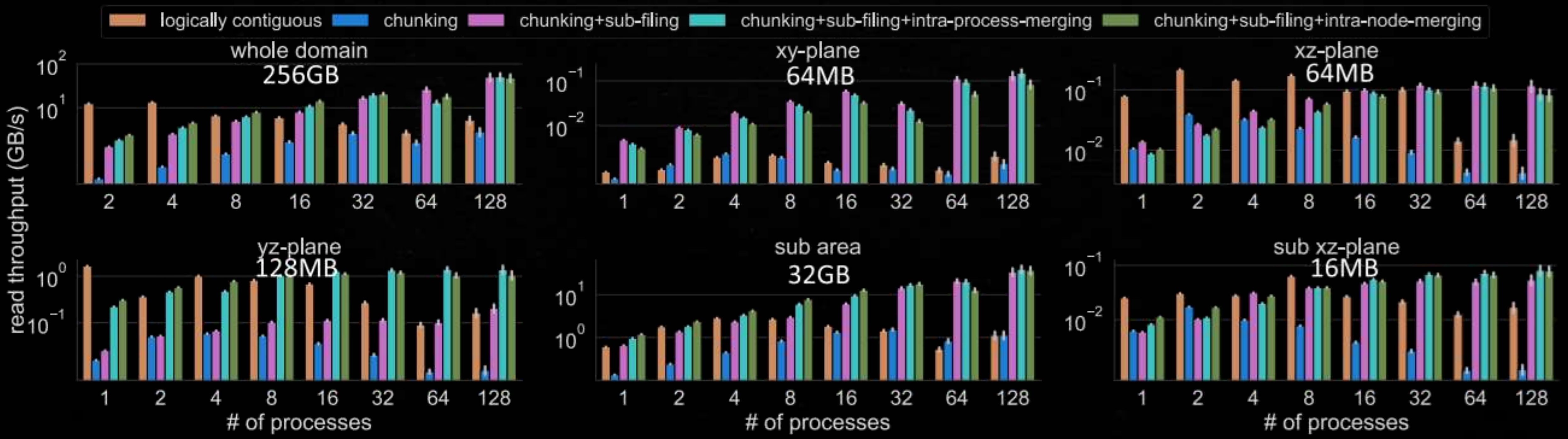


WarpX code

- WarpX is a PIC code with Adaptive Mesh Refinement using AMReX

WarpX write performance on Summit: weak scaling

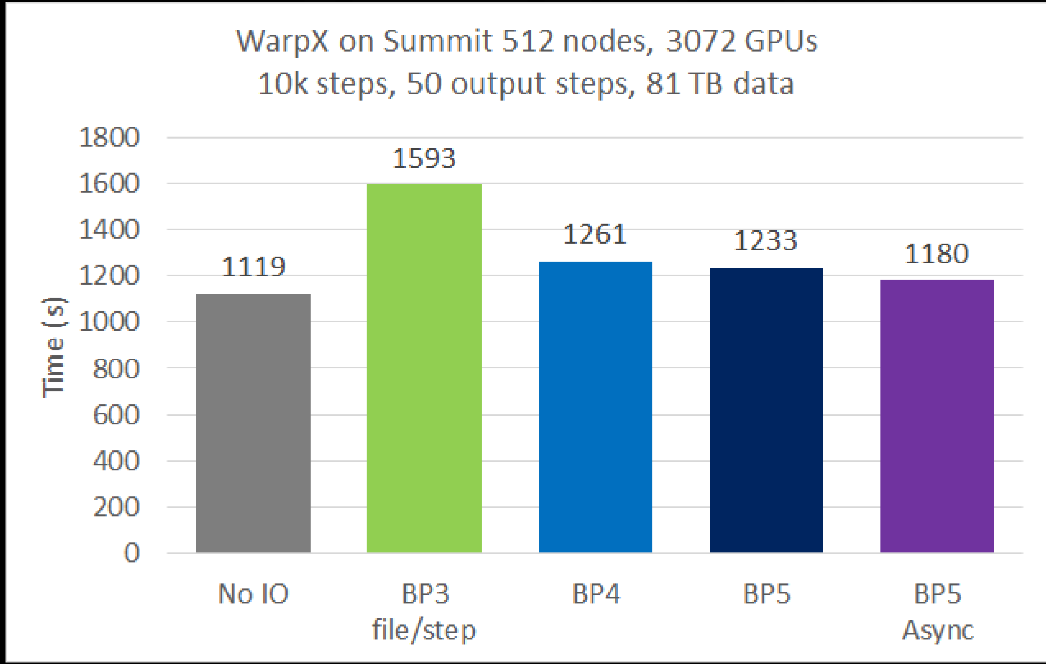


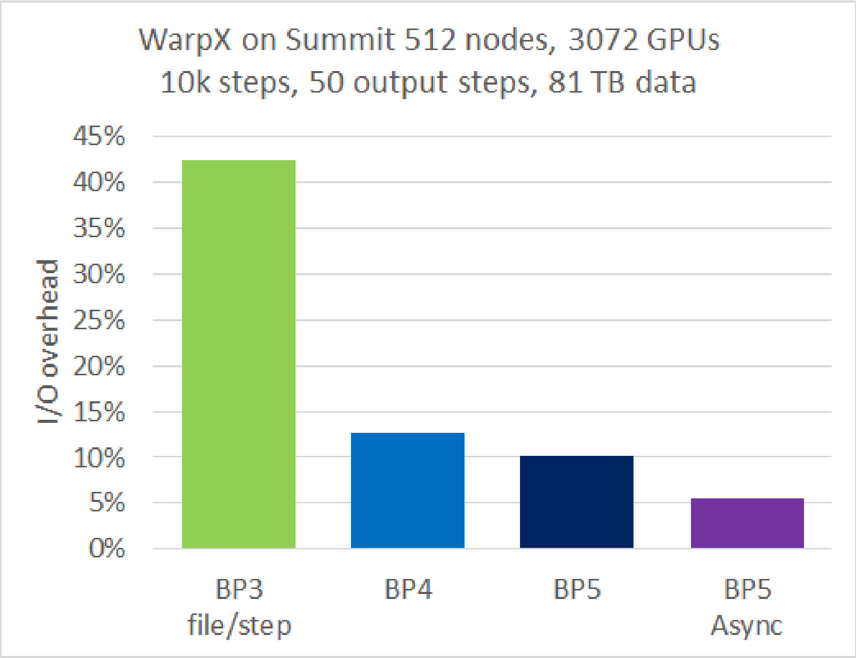


Wan, L., Huebl, A., Gu, J., Poeschel, F., Gainaru, A., Wang, R., ... & Klasky, S. (2021). Improving I/O performance for exascale applications through online data layout reorganization. IEEE Transactions on Parallel and Distributed Systems, 33(4), 878-890.

Asynchronous write to storage on Summit

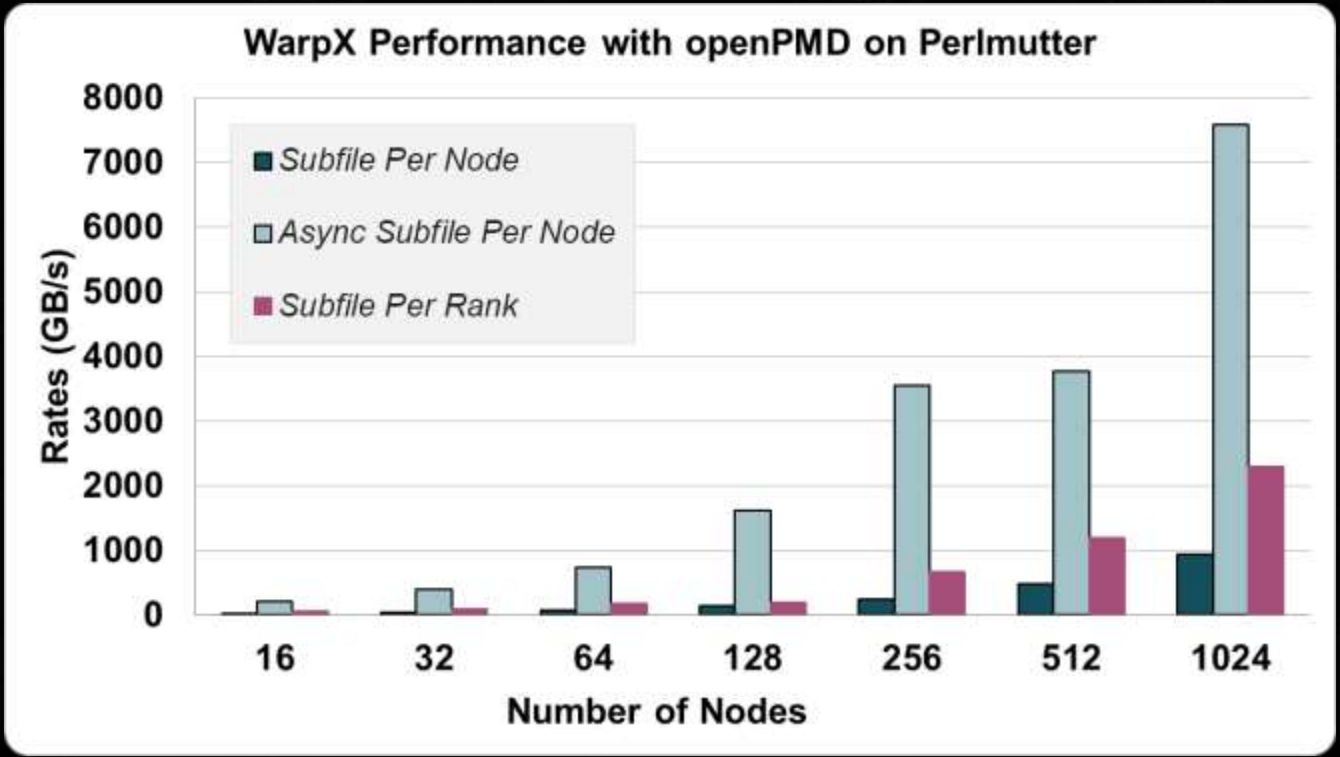
- User friendly On/Off option
  - No need to modify the user code
- Only data writing is async, metadata gathering and writing is still sync
- Don’t have too much experience with this yet





Async IO with WarpX on Perlmutter

- On Frontier we don't see improvement over synchronous I/O

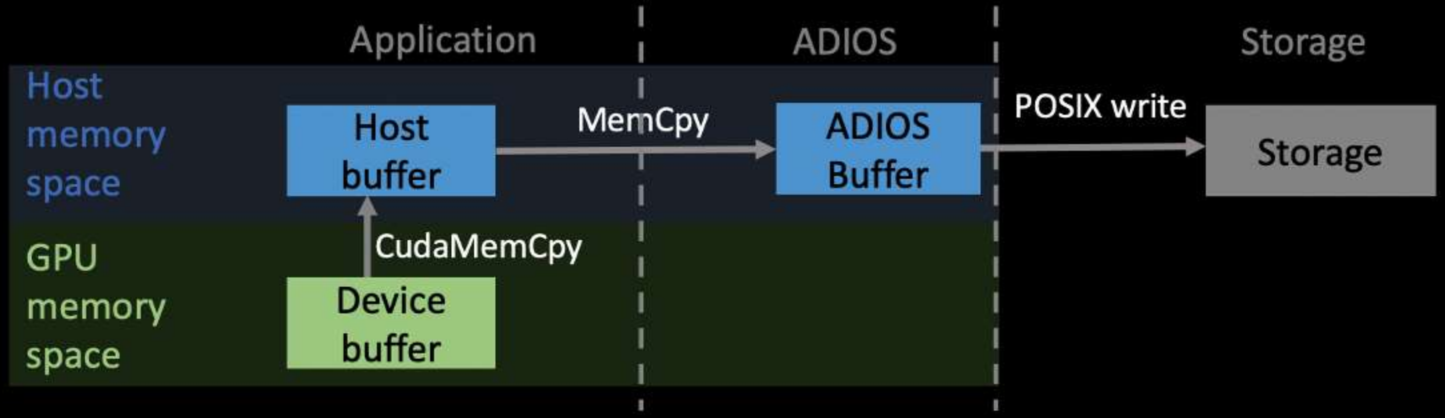


On **Perlmutter** with the default Lustre setting. The rank based aggregations achived **2 TB/sec with 1k nodes**. Turning on **Asynchronous** I/O mode improves this to **7 TB/sec**. Default setup achieved 1 TB/sec.

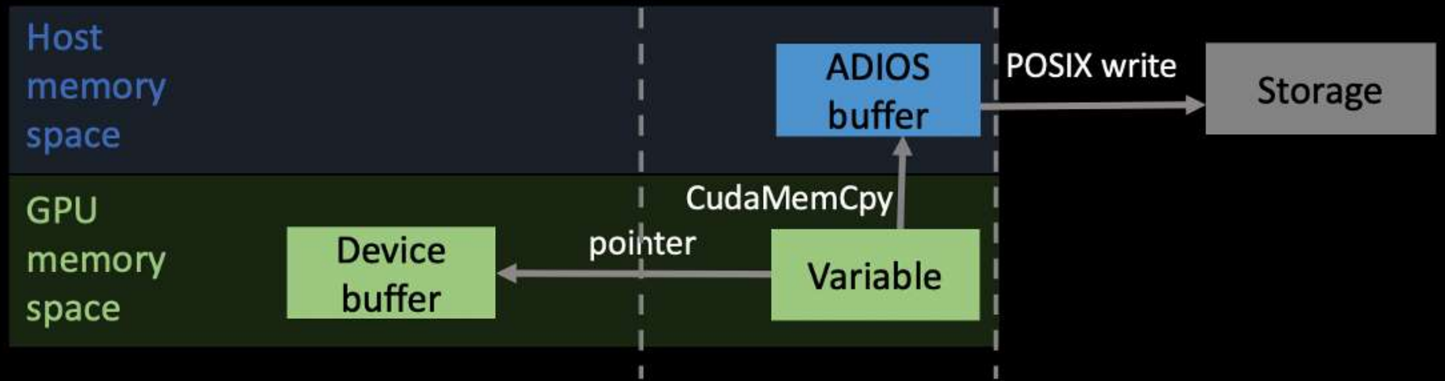
Oak Ridge National Laboratory

GPU-aware I/O

- Allow applications to give ADIOS GPU buffers
  - Decrease number of copies of the data
  - Transparent performance portability to different GPU architectures
  - Allow ADIOS to use GPU direct to storage, compression on GPU, or other optimizations



a) ADIOS using Host buffers (default behavior)



b) ADIOS using GPU buffers

API for GPU-aware I/O

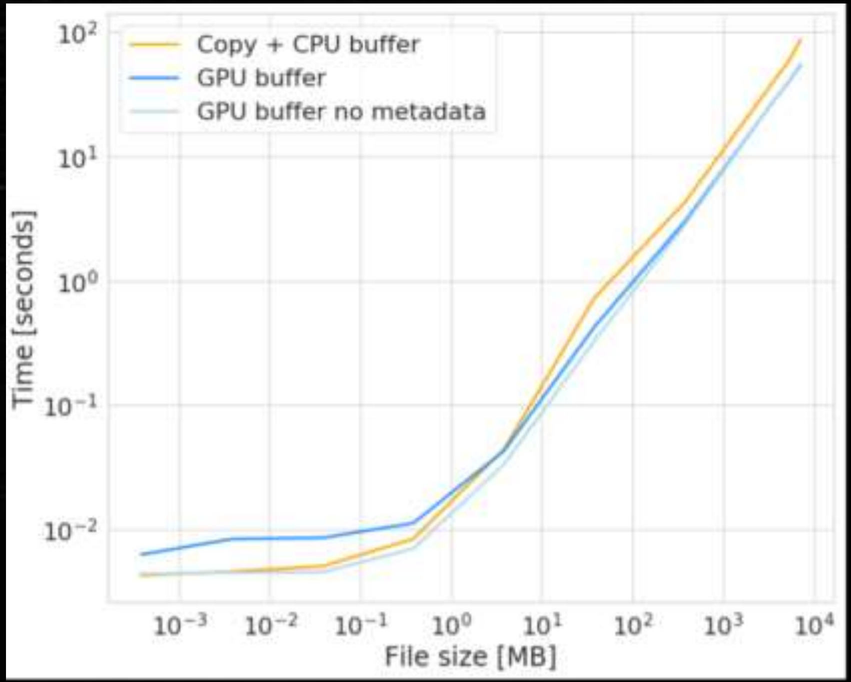
- Build ADIOS2 with CUDA support –D ADIOS2_USE_CUDA=ON
- The user provides a memory space associated with ADIOS2 variables
  - If not set ADIOS2 will detect automatically the memory space

```
adios2::Engine bpWriter;
...
auto data = io.DefineVariable<float>("data", shape, start, count);

bpWriter.Put(data, cpuData);

data.SetMemorySpace(adios2::MemorySpace::GPU);
bpWriter.Put(data, gpuData);
```

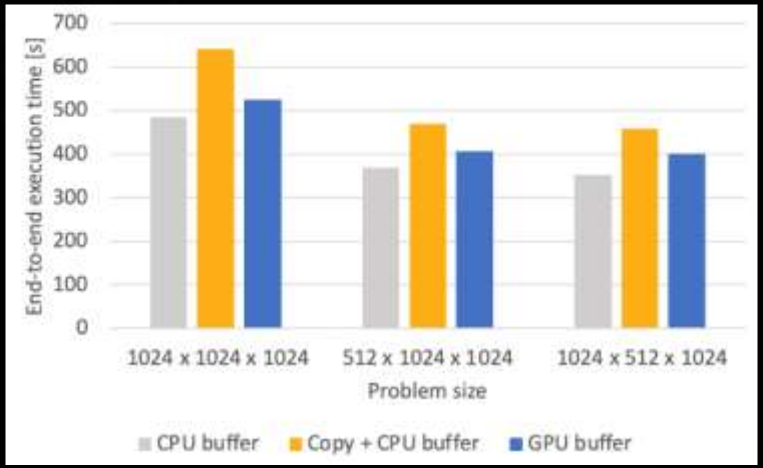
- ADIOS2 saves pointers to data and copies data to internal CPU buffers (in deferred or sync mode)
- Computes metadata for each Get/Put using CUDA kernels

Overhead for detecting where buffers are allocated

CPU STD vector	CUDA CPU buffer	CUDA GPU buffer
5-6 μs	1-2 μs	1-2 μs



Results on I/O kernels and OpenPMD

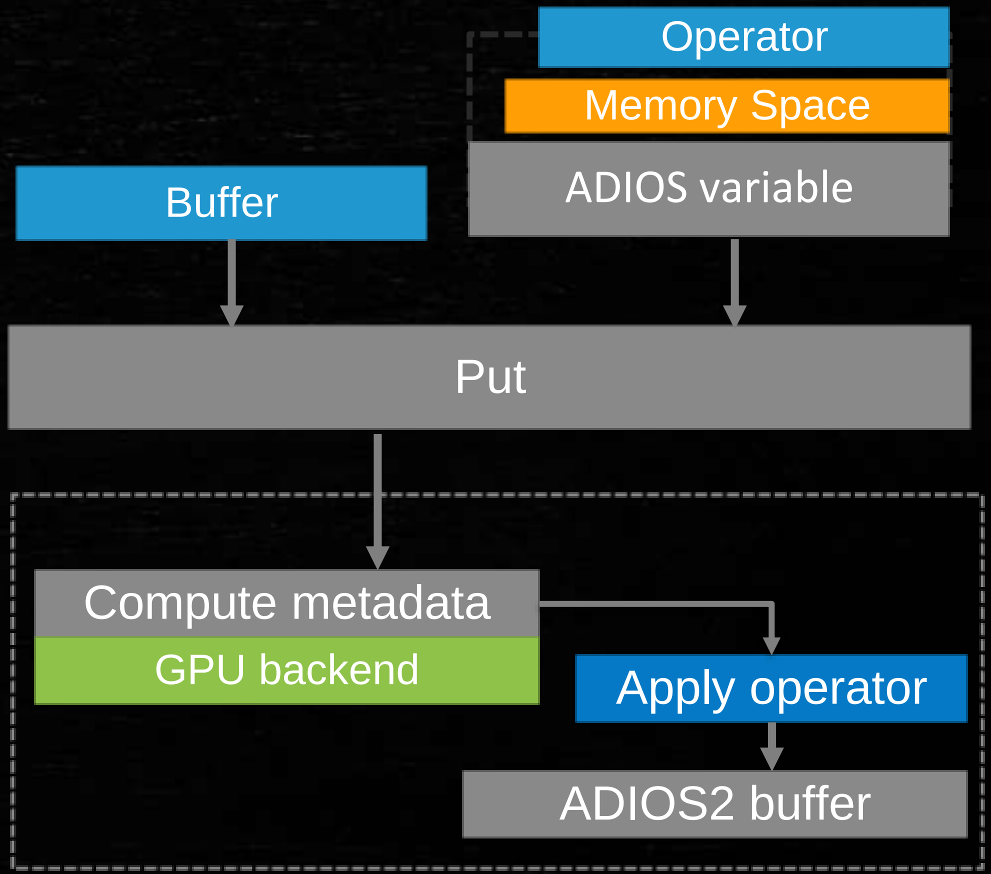


Compression with GPU-aware I/O

- No changes required in the source code
  - Operator attached to a variable
  - Memory space attached to a variable

```
auto var = io.DefineVariable<double>("test", shape, start, count);

// define an operator
adios2::Operator varOp =
  adios.DefineOperator("mgardCompressor", adios2::ops::LossyMGARD);

//attach operator to variable
var.AddOperation(varOp, parameters);

var.SetMemorySpace(adios2::MemorySpace::GPU); // optional
bpWriter.Put(var, gpuSimData);
```

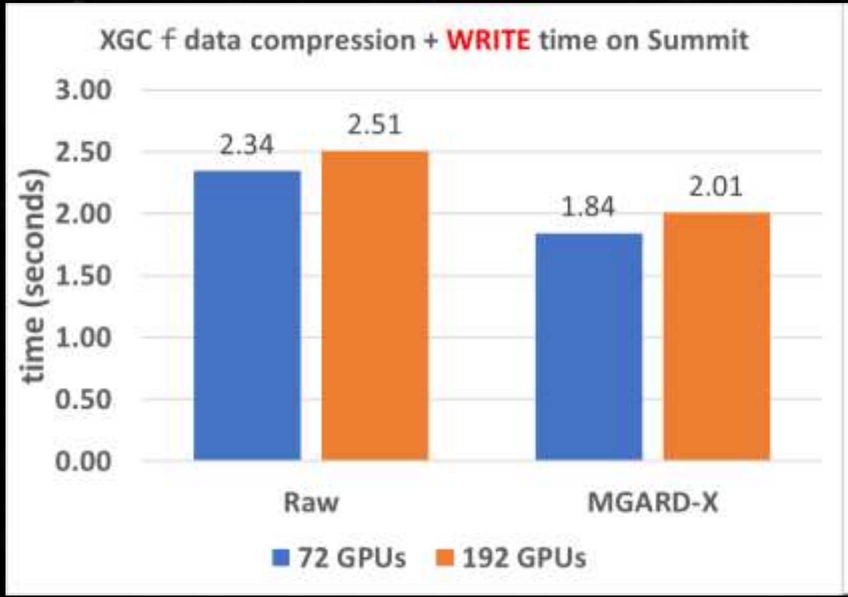
- Internal logic
  - Metadata is computed using the GPU backend
  - The operator is applied on the GPU buffer and the compressed data is copied directly in the ADIOS buffer

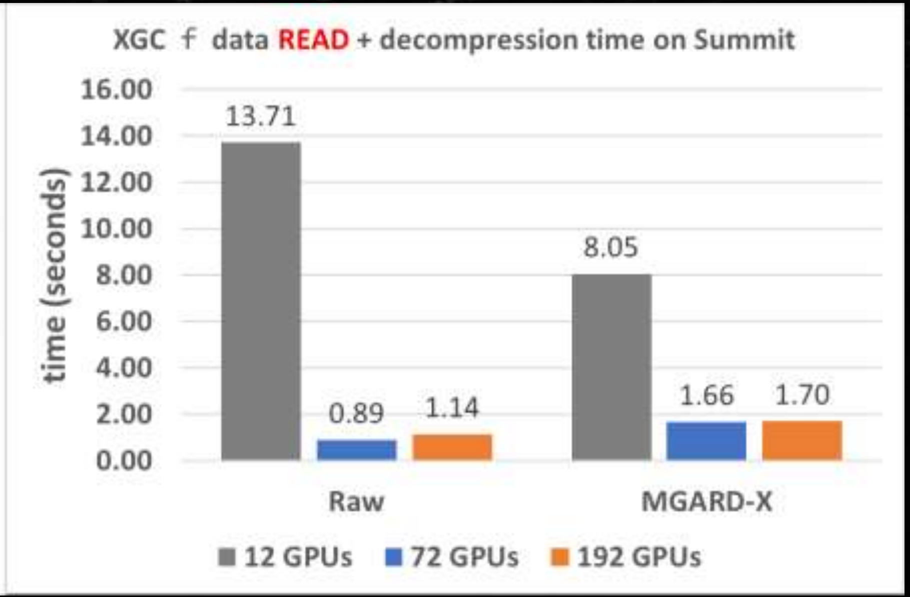


Operators that support GPU buffers:

- MGARD, ZFP
- The operators need to be built with GPU enable

XGC data compression on GPU



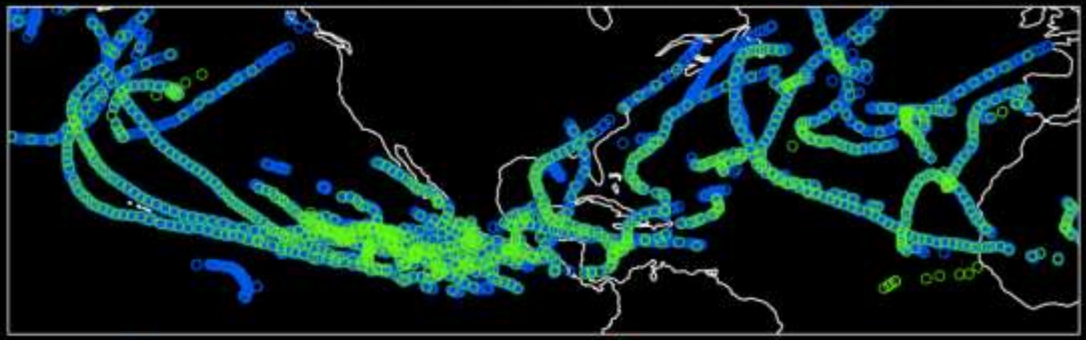


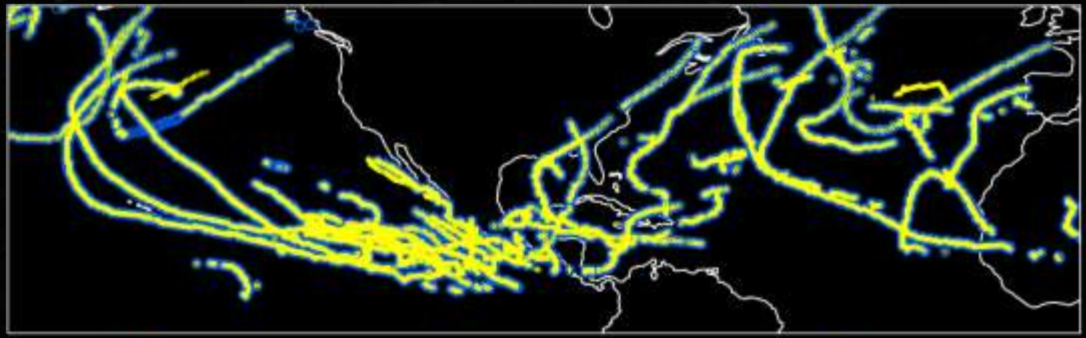
Cost of XGC ***f*** data compression in-place on GPU using MGARD. The GPU-Aware ADIOS is used for moving data between GPU and host memory for I/O purposes, allowing applications to seamlessly compress/decompress data directly on the GPU as part of I/O. This is a strong scaling test of a fixed amount of *f* data where MGARD achieves 13x reduction in file size. Reduction and writing is faster than writing the raw data, however, it still incurs some extra time to read and to reconstruct the data.

Norbert Podhorszki, Scott Klasky, Ana Gainaru, Qian Gong, Jieyang Chen, Sanjay Ranka. The Benefits of Data Reduction for Fusion Datasets. Poster at ICDDPS-4 conference, Apr 2023, Okinawa, Japan

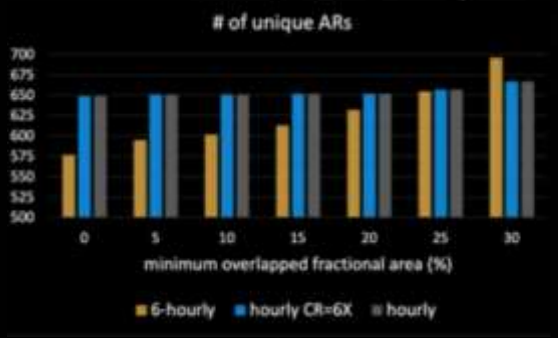
MGARD impact of Temporal Frequency on Climate Feature Tracking

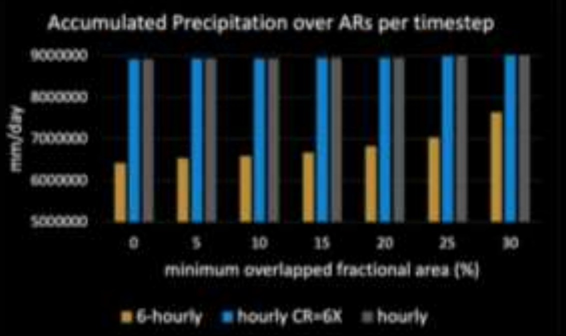
Compression on high-temporal resolution space helps to resolve temporal features





We ran E3SM and output the data every hour (compared to every 6 hours) and show that we can reduce the size by 23X while using MGARD → instead of output every 6 hours we can output every hour and have more accurate cyclone and atmospheric river prediction (using the TEMPEST EXTREME code), while reducing the output by 4X





¼ the storage footprint than 6-hourly data, 19.5X better prediction for AR, 1.8X for TC tracks

Gong, Q., et al. . Spatiotemporally adaptive compression for scientific dataset with feature preservation – a case study on simulation data with extreme climate events analysis, doi:10.1109/e-Science58273.2023.10254796

Optimising the Processing and Storage of Radio Astronomy Data

Scientific Achievement

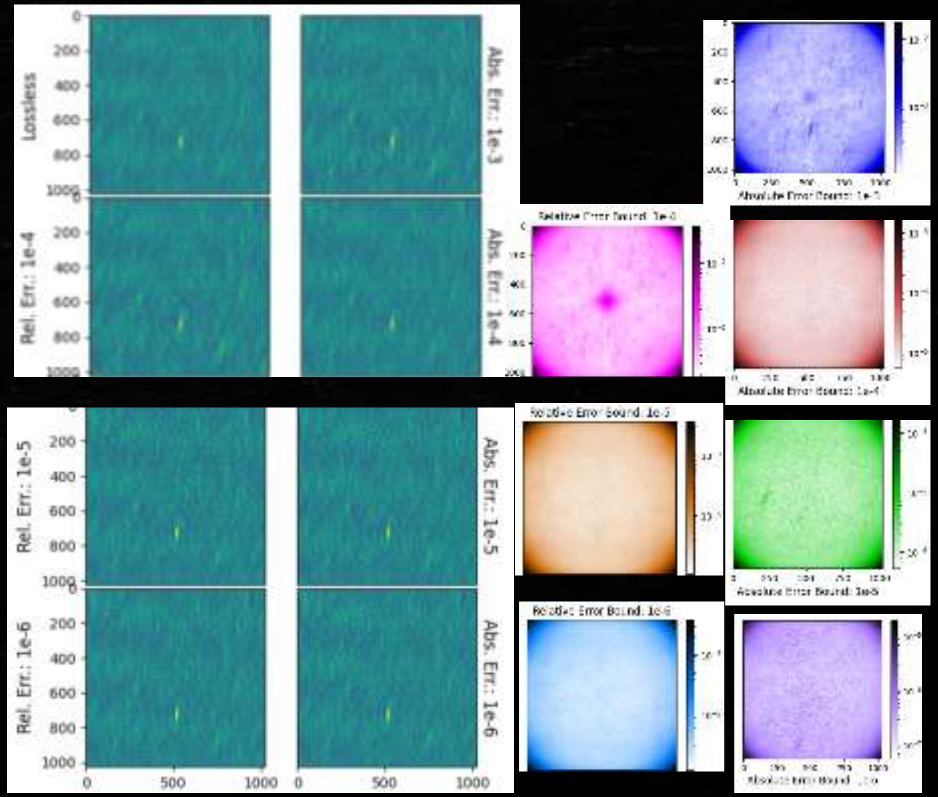
Square Kilometer Array (SKA), the world's largest radio telescope, is anticipated to collect 710 PB data per year, placing significant strains on I/O and storage. Utilizing data from a precursor telescope, we investigate lossy compression methods using the MGARD and ADIOS2 libraries. We demonstrate the improvements in I/O and storage and quantify the impacts on science quality using lossy compressed data.

Significance and Impact

- The advancements in radio telescope present great opportunities for scientific discovery also introduce challenges in data management and processing.
- Our study demonstrates that error-controlled lossy compression can significantly reduce I/O and storage costs while ensuring data accuracy.
- We quantified the impact of lossy compression and found that the SKA data can be compressed by 15x without impacting the integrity of scientific results.

Technical Approach

The raw SKA data was calibrated into a visibility grid, a PSF grid, and a PCF grid, then processed by the cdeconvoler application to generate final images. We compressed the calibrated data using various error bounds and investigated their impacts on the galaxy source after imaging process. We also evaluated the runtime acceleration and storage reduction achieved through compression.



On the left are images produced by cdeconvoler using the grid data compressed by MGARD at different error bounds. On the right are absolute residuals between the images produced using lossy compressed and uncompressed equivalent. Errors in data compressed by a relative error bound <1e-4 or an absolute error bound <1e-3 are not detectible – less than 1σ of the source spectral.

Online and Scalable Data Compression Pipeline With Guarantees on Quantities of Interest

Scientific Achievement

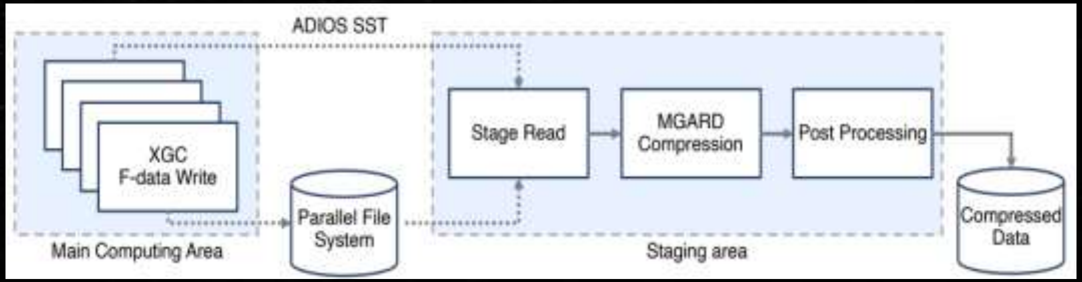
This study presents a high-performance pipeline capable of compressing data in concurrent with simulation, at merely 0.5% of additional computational cost, while ensuring the compression-incurred error on quantities of interest (QoIs) to be less than 10^{-8} at a compression ratio of 150X.

Significance and Impact

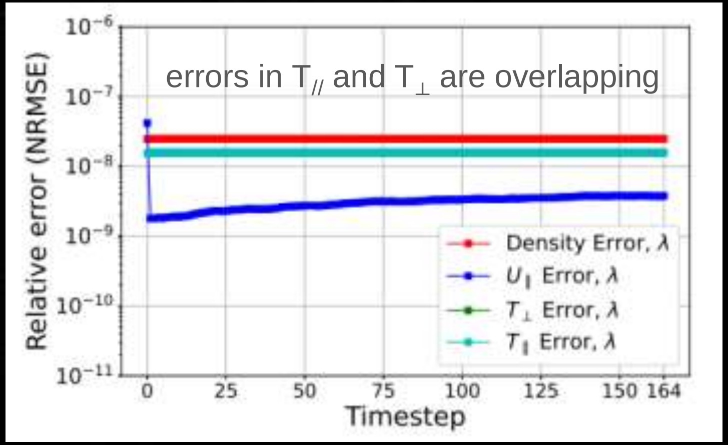
- The imbalanced growth among computing, networking, and storage capabilities necessitate the creation of compression techniques that can greatly reduce the data while accurately preserving derived QoIs.
- This study created a fast and scalable pipeline that allows concurrent execution of data compression in parallel with simulation and I/O through code coupling and staging. The approach allows compression, including the writing of required compression data, for the previous time step to be completed while the simulation proceeds with the current time step.
- Using XGC -- a fusion code producing hundreds of terabytes per simulation cycle -- as an example, this study demonstrates a highly parallel, scalable, and practical solution for applying on-the-fly data compression while guarantees the accuracy of QoIs that are critical to fusion

Technical Approach

- The study integrates a state-of-the-art compressor, MGARD, with post-processing for great data compression and error controls capabilities.
- By using staging nodes, the proposed method pipelines compression and simulation across timestep data which reduces computational overhead.



Data compression pipeline



Achieved errors in four XGC QoIs across 165 timesteps

Tania Banerjee, et al., "Online and Scalable Data Compression Pipeline with Guarantees on Quantities of Interest," IEEE 19th International Conference on e-Science (2023). DOI: 10.1109/e-Science58273.2023.10254934

Staging Options

Transfer mechanisms

- File based (BP4, BP5)
- Network based on the same resource (SST, SSC)
  - RDMA (libfabric, UCX)
  - MPI (one sided, two sided)
  - TCP/ RUDP
- Memory references
- WAN data transfer (DataMan,SST)
  - Streams – TCP, RUDP, RoCE

Placement options

- Same core (inline code)
- Different cores/same node
- Different nodes
- Different resource (LAN)
- Different resource (WAN)
- Hybrid (mixture of options)

Scheduling options

- Fully synchronous
- Fully asynchronous
- Hybrid

Refactoring options

- Prioritize which data gets moved first

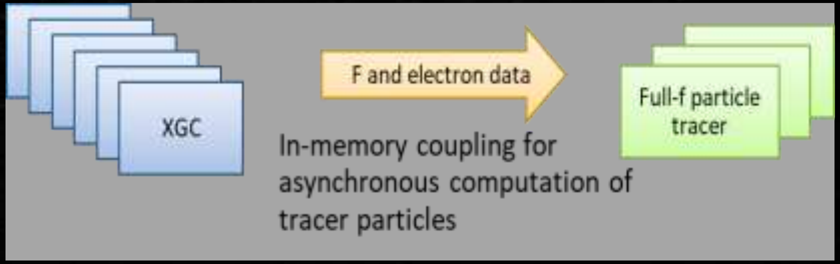
Storage options

- ADIOS-BP5
- HDF5

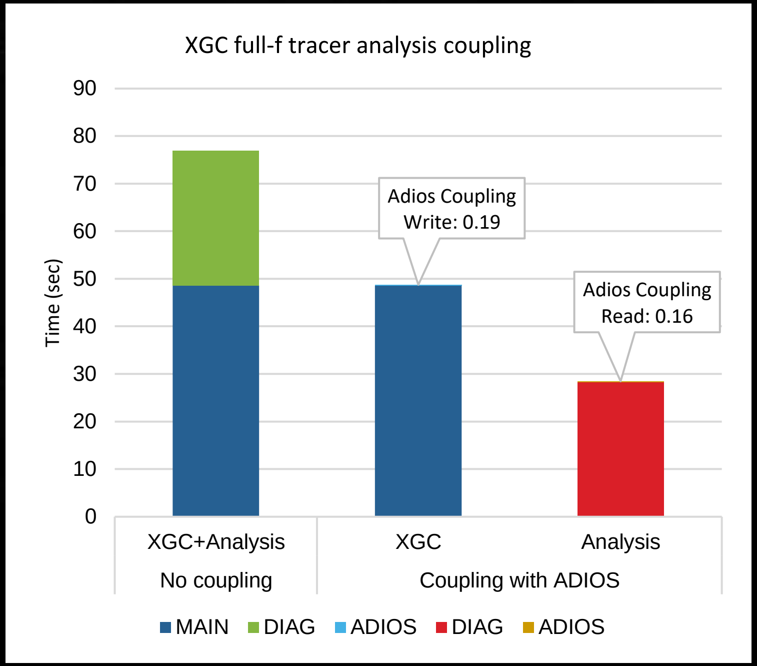
Gainaru, A., Klasky, S., et al., 2022, Understanding the Impact of Data Staging for Coupled Scientific Workflows, IEEE transactions on parallel and distributed systems. 2022.

Staging use case: off-load non-scaling code part

- Tracer particle analysis enables understanding of the transport characteristics spanning the pedestal and scrape-off layer
- It is costly to perform and is communication-heavy
- Asynchronously stage data to the tracer particle analysis running on additional nodes
  - Coupled data size: f0(95 GB) + E_rho/pot_rho(1.4GB)
- Reduced 36% of the XGC iteration time by using asynchronous services (only 0.4% time-overhead for coupling data)



Full-f particle analysis coupling



Full-f coupling performance on Summit with ADIOS using 4/1024 extra nodes

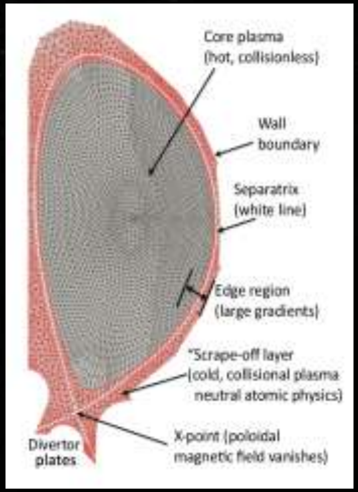
Choi, J. Y., Chang, C. S., Dominski, J., Klasky, Churchill, M., S., Merlo, G., Suchyta, E., ... & Wood, C. "Coupling exascale multiphysics applications: Methods and lessons learned". IEEE e-Science, 2018.

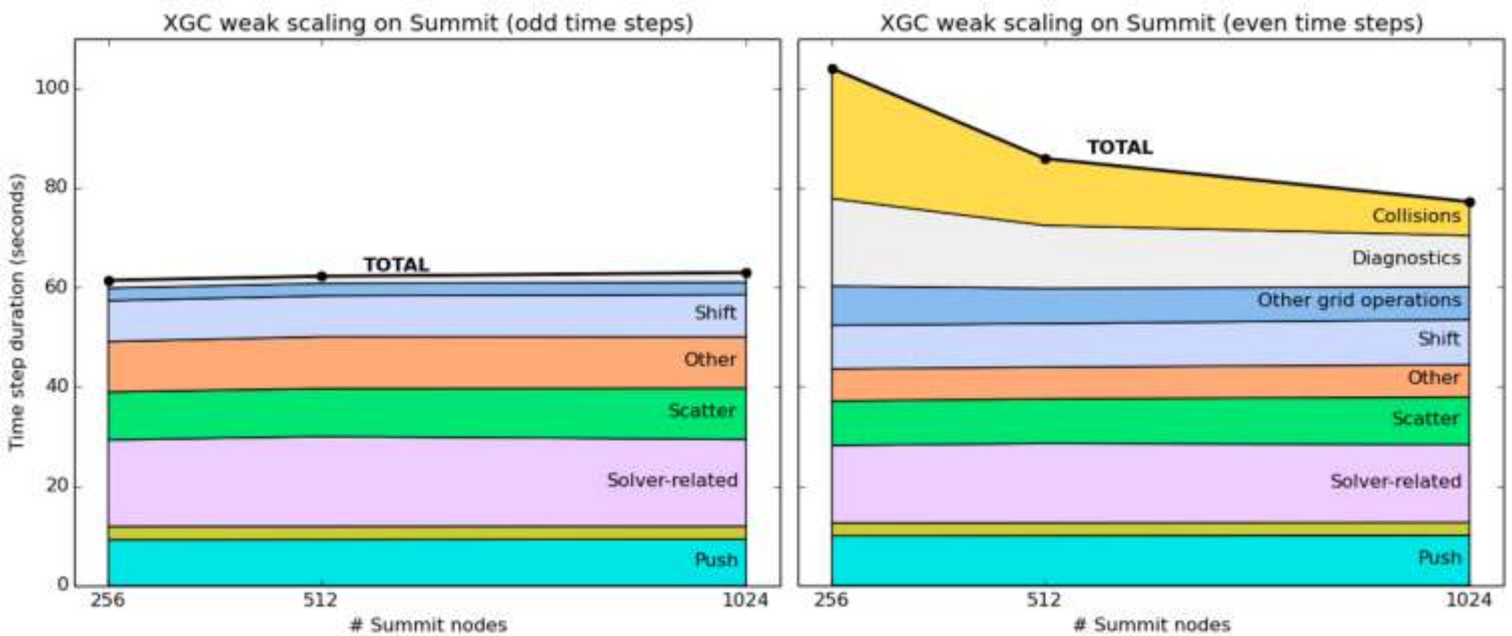
HBPS High-fidelity Boundary Plasma Simulation

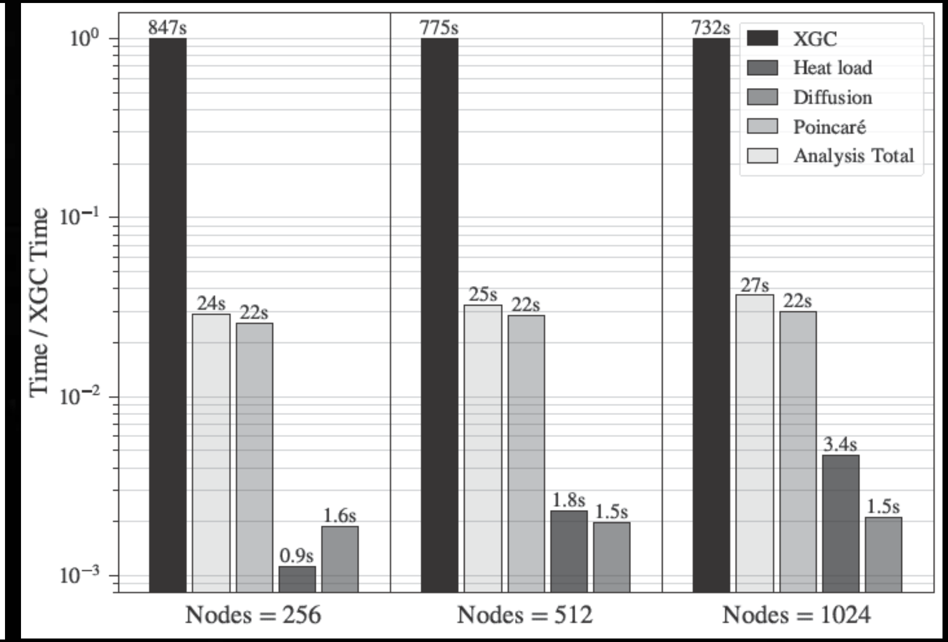
OAK RIDGE National Laboratory

Hybrid Analysis of Fusion Data for Online Understanding of Complex Science on Extreme Scale Computers

- We examine a complex workflow using XGC on Summit, with three in situ analysis for new scientific discovery
- We execute XGC along with three analysis routines (Poincaré surface plot, Head Load calculation, Diffusion Calculation)
- The overhead was 0.1% 1/1024 nodes





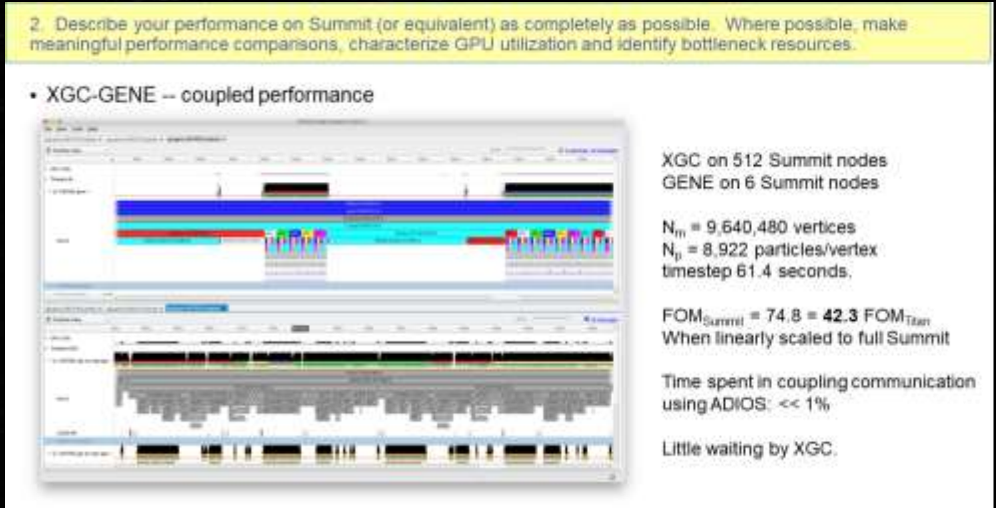


Suchyta, E., Klasky, S. , et al., Hybrid Analysis of Fusion Data for online understanding of complex science on extreme scale computers, Cluster 2023.

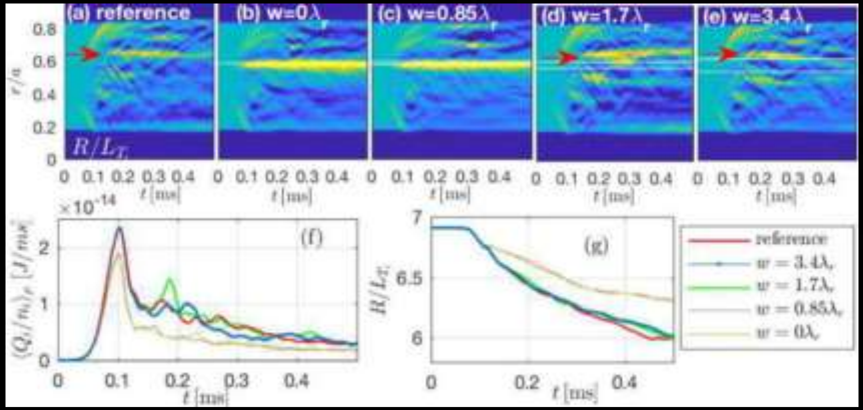
High-Fidelity Whole Device Modeling of Fusion Plasmas

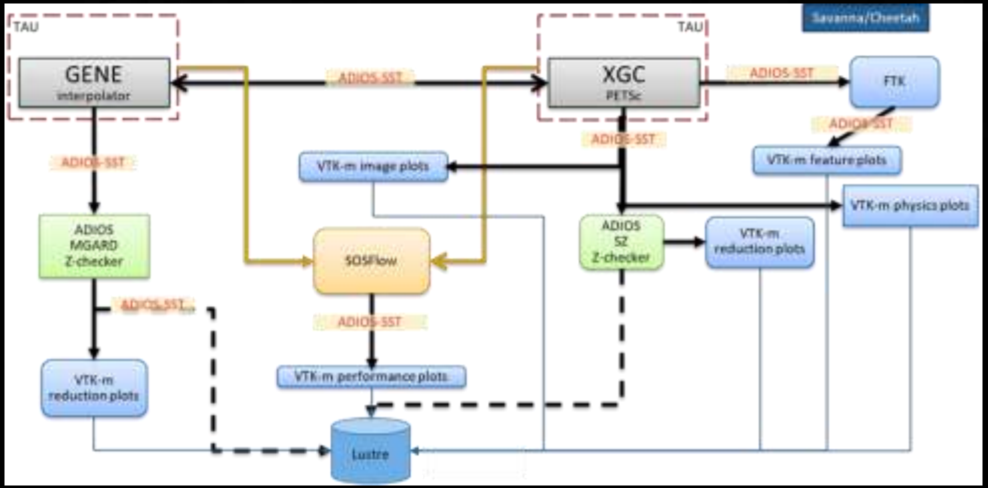
PI: Amitava Bhattacharjee, PPPL,
C. S. Chang, PPPL

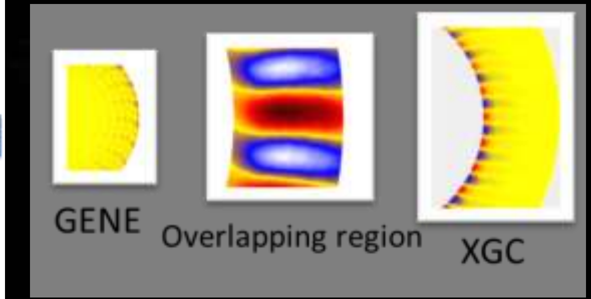
- Different physics solved in different physical regions of detector (spatial coupling)
- Core simulation: **GENE**
  Edge simulation: **XGC**
  Separate teams, **separate codes**
- Recently demonstrated first-ever successful kinetic coupling of this kind
- Data Generated by one coupled simulation is predicted to be > 10 PB/day on Summit



From FY21 WDMApp Review

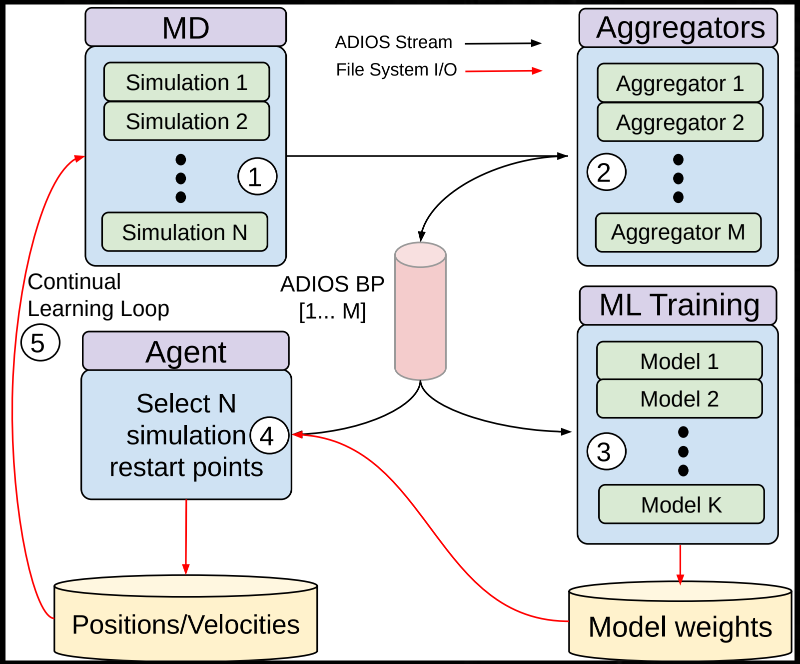


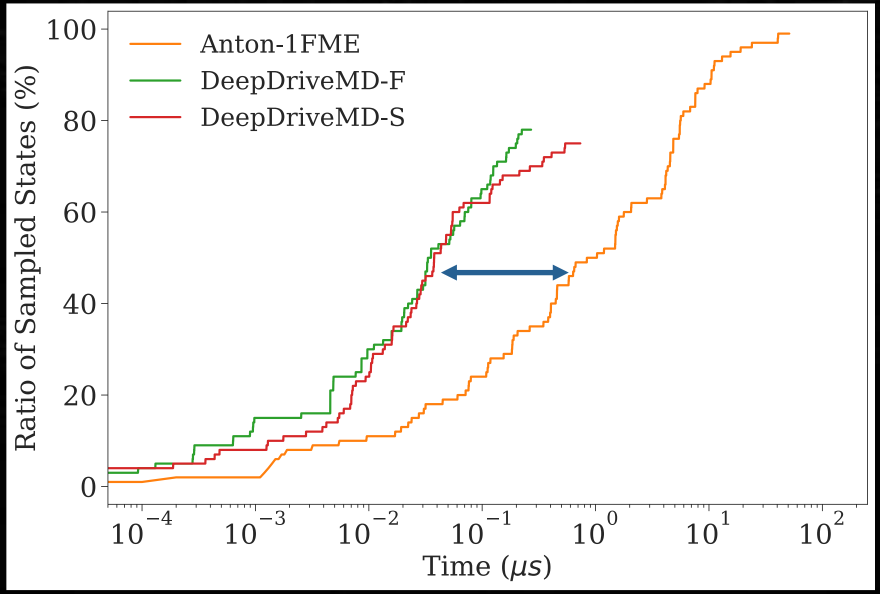


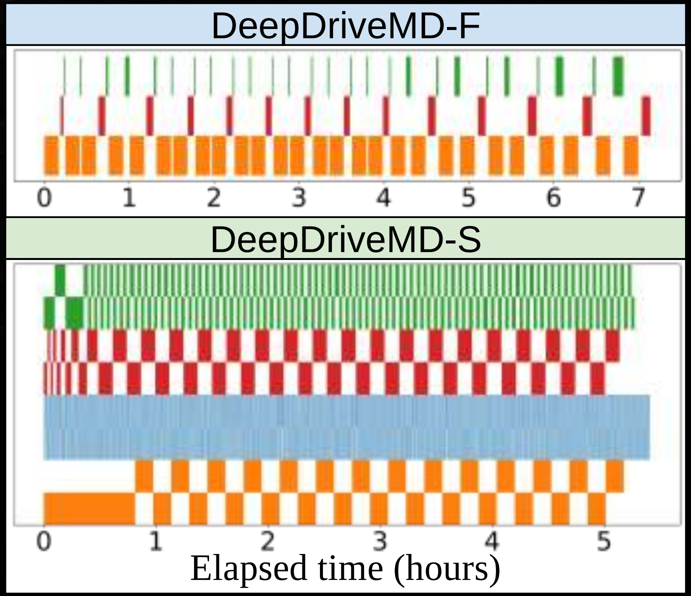


Dominski, J., et al. "Spatial coupling of gyrokinetic simulations, a generalized scheme based on first-principles." Physics of Plasmas 28.2 (2021): 022301.
Merlo, G., et al. "First coupled GENE–XGC microturbulence simulations." Physics of Plasmas 28.1 (2021): 012303.
Cheng, Junyi, et al. "Spatial core-edge coupling of the particle-in-cell gyrokinetic codes GEM and XGC." Physics of Plasmas 27.12 (2020): 122510.

DeepDriveMD: enhancing the scalability for streaming AI runs





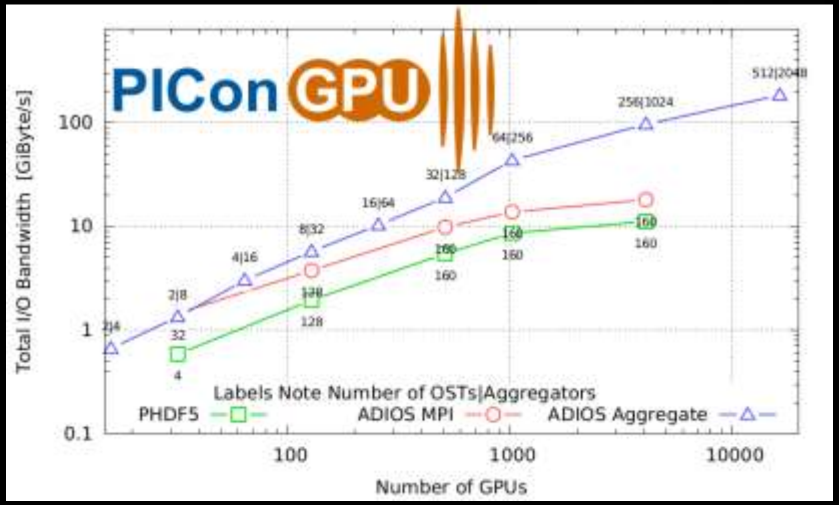


- DeepDriveMD-S: streaming implementation with ADIOS-BP
- Continual learning loop enabled by ADIOS constructs
- Streaming application of ML/AI
- Streaming runs have better resource utilization for protein folding simulations than static file system-based runs
- At least 2 orders of magnitude (100x) acceleration in sampling conformational states related to protein folding
- Faster time-to-solution enabled by streaming runs

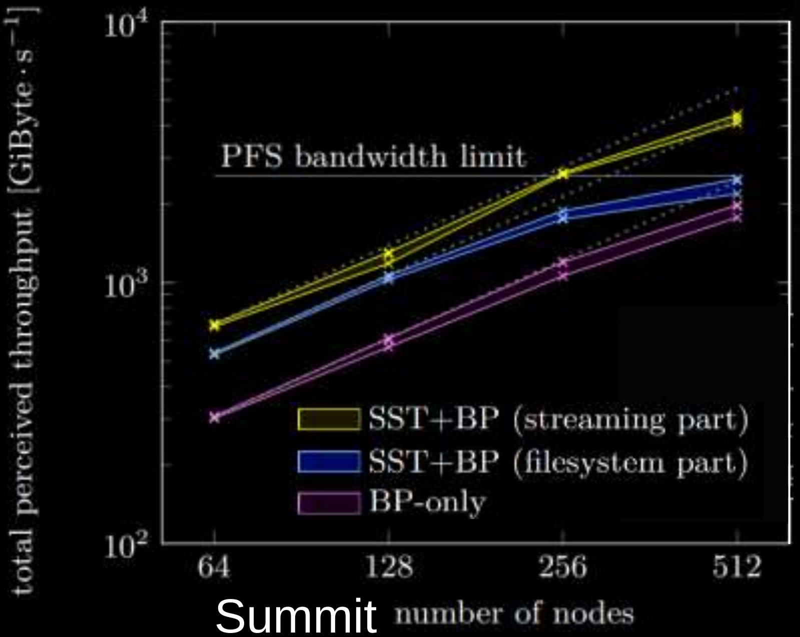
A. Brace *et al*., "Coupling streaming AI and HPC ensembles to achieve 100–1000× faster biomolecular simulations," *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, Lyon, France, 2022, pp. 806-816, doi: 10.1109/IPDPS53621.2022.00083.

Accelerator Physics: PIConGPU

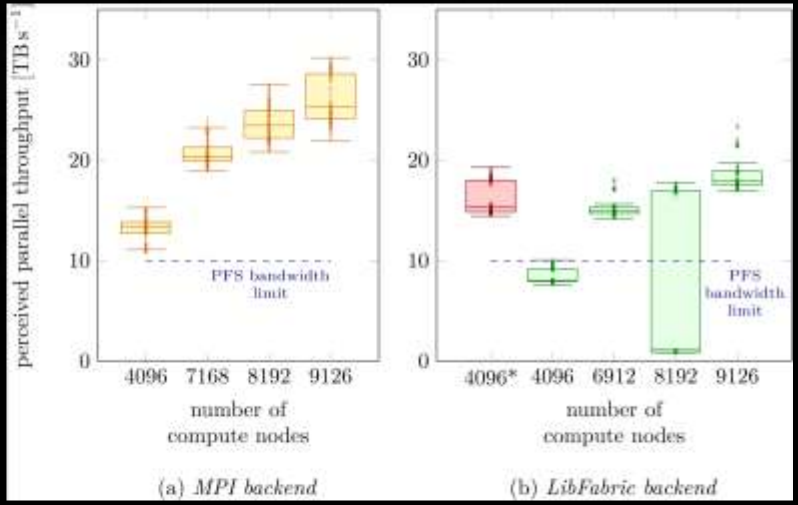
- We originally helped PIConGPU to achieve I/O performance on the OLCF Titan system
  - Allowed their code to get >10X I/O performance improvement
- Next, we utilized staging to increase the I/O bandwidth on Summit
- Now we are working on more in situ techniques for AI, Digital Twins on Frontier



Titan



Summit



Frontier

5.6 GB / node data output every step

Visualization services with Paraview/Catalyst/FIDES/VTK-M/ADIOS

- Fides is a visualization schema
- Fides maps ADIOS data arrays to VTK-M datasets, enabling viz using shared and distributed-memory parallel algorithms
- Catalyst provides in situ data analysis and visualization capabilities for ParaView

https://fides.readthedocs.io

- Interactive visualization in a GUI – post processing
- Batch visualization – post processing on compute nodes
- In situ (inline) interactive visualization in a GUI
- In situ (inline) batch visualization
- In situ (in transit) batch visualization

Using the same application that outputs data using ADIOS

Post hoc

Simulation → → Vis service

BP5, HDF5

In situ (inline)

Simulation | Vis service

ParaViewADIOSInSituEngine

In transit

Simulation → Vis service

SST, SSC staging and BP5 through file system

Pugmire, D. , et al. "Fides: a general purpose data model library for streaming data." ISC 202, Springer International Publishing, 2021.

Building applications with ADIOS

Build ADIOS2 on Frontier

https://adios2.readthedocs.io/en/latest/setting_up/setting_up.html

- Frontier environment

```
$ module load PrgEnv-XXXX
$ module load cray-python # or your favorite Python environment
$ module load cmake
```

- Get ADIOS source

```
$ wget https://github.com/ornladios/ADIOS2/archive/refs/tags/v2.10.1.tar.gz
$ tar xf v2.10.1.tar.gz
$ cd ADIOS2-2.10.1
$ mkdir build
```

- Build and install

```
$ cmake -DCMAKE_INSTALL_PREFIX=<adios_install_path>  -S . -B build
$ cmake --build build -j 8
$ cmake --install build
```

Load more modules or use
-DCMAKE_PREFIX_PATH="<path1>;<path2>;…" to point to extra libraries

Remember <*adios_install_path*>

Frontier

- Check what is supported in ADIOS

```
$ <adios_install_path>/bin/bpls -Vv
blps: ADIOS file introspection utility

Build configuration:
ADIOS version: 2.10.1
C++ Compiler:  GNU 12.2.0      Clang 17.0.3 (CrayPrgEnv)     Clang 17.0.0 (CrayPrgEnv)
Target OS:     Linux-5.14.21-150400.24.46_12.0.83-cray_shasta_c
Target Arch:   x86_64
Available engines = 11: BP3, BP4, BP5, HDF5, SST, SSC, Inline, MHS,
ParaViewADIOSInSituEngine, Null, Skeleton
Available operators = 4: BZip2, Blosc, ZFP, PNG
Available features = 18: DATAMAN, HDF5, HDF5_VOL, MHS, SST, MPI, PYTHON, BLOSC2, BZIP2,
PNG, ZFP, O_DIRECT, SODIUM, CATALYST, SYSVSHMEM, ZEROMQ, PROFILING, ENDIAN_REVERSE
```

Build ADIOS2 on Frontier with GPU support using Kokkos

See ADIOS2 source: scripts/build_scripts/build-adios2-kokkos-crusher.sh for guidance

Frontier environment

```
$ module load rocm
$ module load craype-accel-amd-gfx90a
```

- Get Kokkos source

```
$ wget https://github.com/kokkos/kokkos/archive/refs/tags/4.3.01.tar.gz
$ tar xf 4.3.01.tar.gz
$ cd kokkos-4.3.01
$ mkdir build
```

- Build and install

```
$ cmake -DCMAKE_BUILD_TYPE=Release  -DCMAKE_CXX_COMPILER=hipcc  -DKokkos_ENABLE_SERIAL=ON  -DKokkos_ARCH_ZEN3=ON  -DKokkos_ENABLE_HIP=ON  -DKokkos_ARCH_VEGA90A=ON  -DCMAKE_CXX_STANDARD=17  -DCMAKE_CXX_EXTENSIONS=OFF  -DCMAKE_POSITION_INDEPENDENT_CODE=TRUE -DBUILD_SHARED_LIBS=ON -DCMAKE_INSTALL_PREFIX=<kokkos_install_path>  -S . -B build
$ cmake --build build -j 8
$ cmake --install build
```

Add

-DCMAKE_PREFIX_PATH="*<kokkos_install_path>*" -DADIOS2_USE_Kokkos=ON

to the ADIOS configuration

Oak Ridge National Laboratory

Compile ADIOS2 codes

- CMake
  - Use MPI_C and ADIOS2 packages

```
CMakeLists.txt:
project(gray-scott C CXX)
find_package(MPI REQUIRED)
find_package(ADIOS2 REQUIRED)
add_definitions(-DOMPI_SKIP_MPICXX -DMPICH_SKIP_MPICXX)
...
target_link_libraries(gray-scott   adios2::cxx11_mpi   MPI::MPI_C)
```

  - Configure application by adding ADIOS installation to search path

```
cmake -DCMAKE_PREFIX_PATH=<adios_install_path>  -S <source_dir> -B <build_dir>
```

  - Available ADIOS2 targets: cxx11 c, fortran, cxx11_mpi, c_mpi, fortran_mpi

Compile ADIOS2 codes

- Makefile
  - Add ADIOS2 library paths to LD_LIBRARY_PATH
  - Use adios2_config tool to get compile and link options

```
ADIOS2_DIR = <adios_install_path>
ADIOS2_FINC=`${ADIOS2_DIR}/bin/adios2-config --fortran-flags`
ADIOS2_FLIB=`${ADIOS2_DIR}/bin/adios2-config --fortran-libs`
```

  - Codes that write and read

```
heatSimulation: heat_vars.F90 heat_transfer.F90 io_adios2.F90
	${FC}   -g -c -o heat_vars.o heat_vars.F90
	${FC}   -g -c -o heatSimulation.o heatSimulation.F90
	${FC}   -g -c -o io_adios2.o ${ADIOS2_FINC} io_adios2.F90
	${FC}   -g -o heatSimulation heatSimulation heat_vars.o io_adios2.o  ${ADIOS2_FLIB}
```

Summary

ADIOS brings a programming interface and a framework of many solutions to the generic problem of producing and consuming data

- The interface frees scientists from the limited scope of file-based data processing
  - Being fully applicable to file-based data processing
- Scalable IO: number of processes, variables and steps; and amount of data
- Offering a bridge from their scientific workflows that work now to the future, where they will extend their workflows with
  - More efficient data processing
  - Interactive visualization
  - Code coupling
  - On-the-fly AI training
  - Combining experimental data with simulation data