OAK RIDGE National Laboratory

I/O with Kokkos

Ana Gainaru

Kokkos Tea-Time

Jan 15, 2025

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

U.S. DEPARTMENT OF ENERGY

ADIOS
easy-to-use, fast, scalable, and portable I/O

gainarua@ornl.gov

Summary

- **The ADIOS I/O framework**
- **Kokkos applications using ADIOS**
  - Store and stream data
  - Campaigns and querying
  - Remote access to monitor the performance
- **ADIOS using Kokkos**
  - GPU-backend
  - Derived variables

What is ADIOS2

ADIOS

- High performance I/O abstraction to allow for on-line/off-line memory/file data subscription service
  - Declarative, **publish/subscribe API** is separated from the I/O strategy
  - **I/O engines** provide different strategies for data movement
  - **Operators** can be added to data transfers
  - Metadata is computed for **queries** on the reader side

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

https://github.com/ornladios/ADIOS2

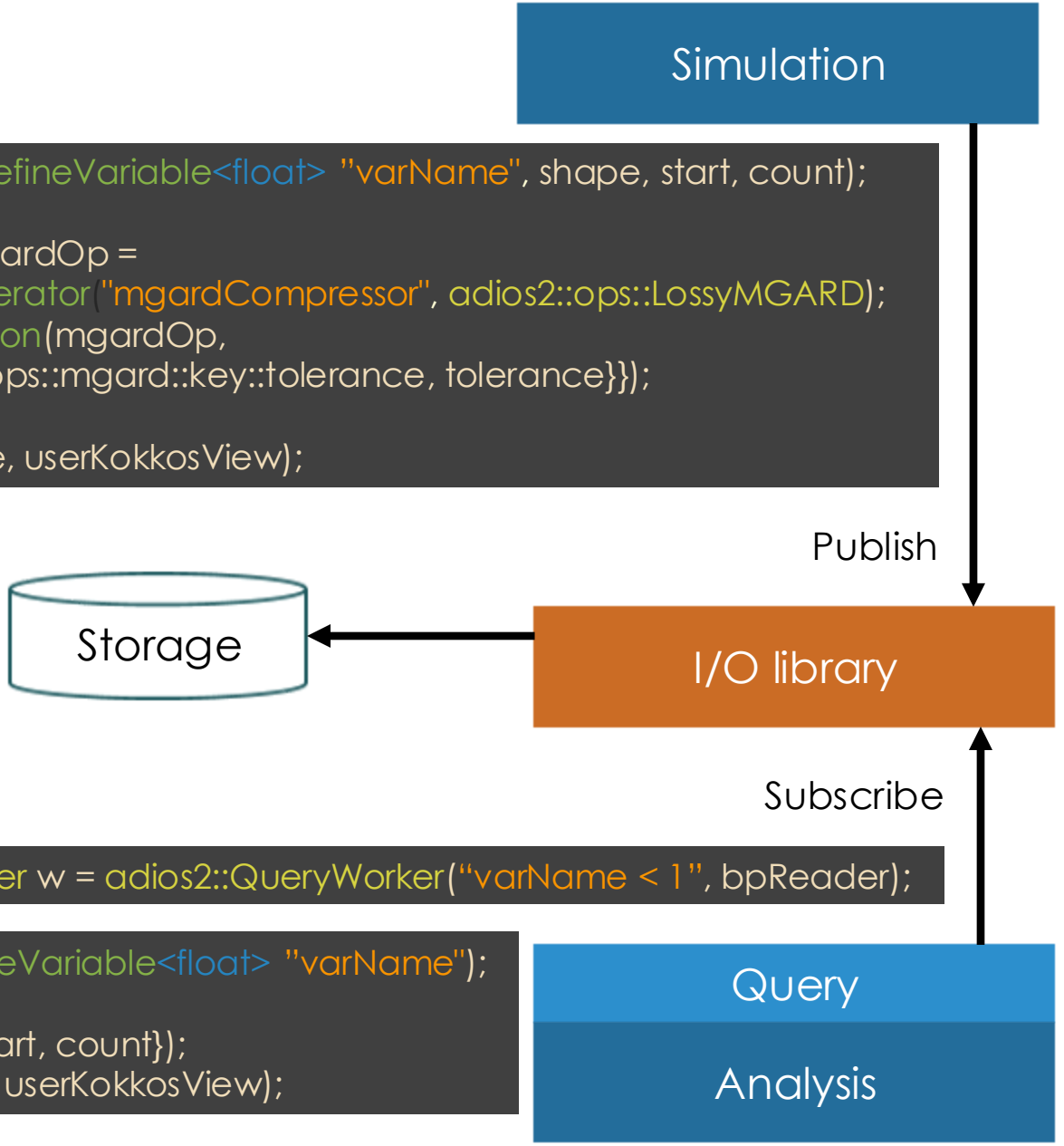
Contributors

OAK RIDGE National Laboratory, kitware, BERKELEY LAB, GEORGIA TECH, RUTGERS

A bit more on ADIOS

- **Publish API**
  - Define an ADIOS variable
    - With a certain global and local shape
  - Add an operator
  - Publish data
    - Aggregated in internal buffers
- **Subscribe API**
  - Subscribe to data
  - Query data
  - Attach an accuracy
- **I/O engines**
  - Keep the same code
  - Switch the data management solution

```
auto variable = io.DefineVariable<float> "varName", shape, start, count);

adios2::Operator mgardOp =
      adios.DefineOperator("mgardCompressor", adios2::ops::LossyMGARD);
variable.AddOperation(mgardOp,
                {{adios2::ops::mgard::key::tolerance, tolerance}});

bpWriter.Put(variable, userKokkosView);
```




```
adios2::QueryWorker w = adios2::QueryWorker("varName < 1", bpReader);
```

```
auto variable = io.InquireVariable<float> "varName");

variable.SetSelection({start, count});
bpReader.Get(variable, userKokkosView);
```

Self-describing Scientific Data

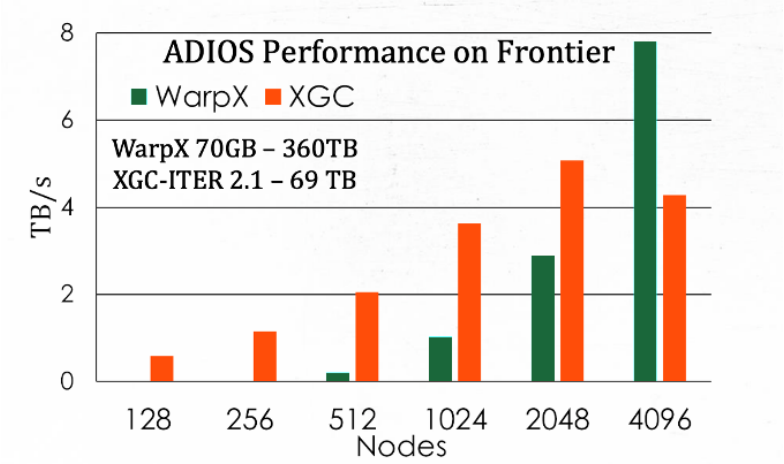
```
double   BOUT_VERSION scalar = 5.2                    double   dx           {68, 20} = 0.2 / 0.2
double   Bxy          {68, 20} = 1 / 1                double   dy           {68, 20} = 1 / 1
string    Bxy/cell_location attr   = "CELL_CENTRE"    double   dz           {68, 20} = 0.2 / 0.2
string    Bxy/direction_y   attr   = "Standard"       double   g11          {68, 20} = 1 / 1
string    Bxy/direction_z   attr   = "Average"        int32_t  nx           scalar = 68
string    Bxy/source        attr   = "Coordinates"    int32_t  ny           scalar = 16
double   G1           {68, 20} = 0 / 0                int32_t  nz           scalar = 64
double   G2           {68, 20} = 0 / 0                double   phi          143*{68, 20, 64} =
double   G3           {68, 20} = 0 / 0                                      -0.139167 / 0.0899946
double   J            {68, 20} = 1 / 1                string   run_id       scalar =
int32_t  MXG          scalar = 2                           "cfc9cd3d-3ec1-4238-8fa0-f75f97a9c949"
int32_t  iteration    143*scalar = -1 / 141           double   t            143*scalar = 0 / 142
 …
double   n            143*{68, 20, 64} =
                      -0.185305 / 0.0961174
```

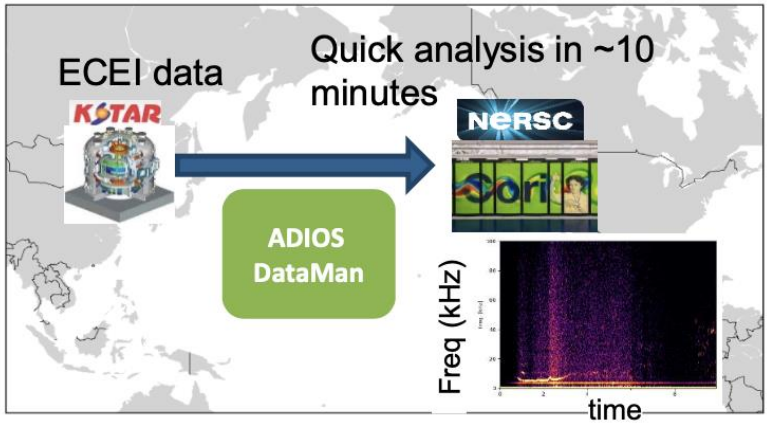
143 output steps of a 3D array of double type and 68x20x64 dimensions, named n
global min = -3.76192 max = 4.05582

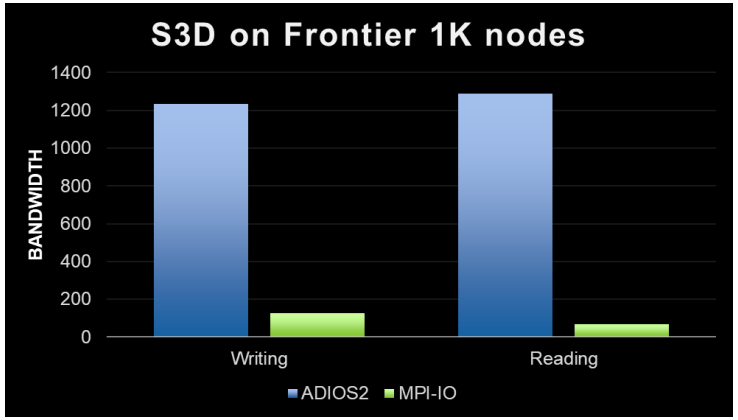
- BOUT++ hasegawa-wakatani-3d example, partial list of variables

A few of our applications

- Wind Turbine (GE)
- Accelerator Physics (PIConGPU, WarpX)
- Fusion (GTC, GE, M3DC1, XGC, GENE, KSTAR)
- Cancer research
- Combustion (S3D)
- Climate (E3SM)
- Radio astronomy (SKA)
- Seismic Tomography Workflow
- Molecular dynamic (DeepDriveMD)







GPU-aware

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

- Build ADIOS2 with Kokkos support –D ADIOS2_USE_Kokkos=ON

- The user can provide a memory space
  - If not set ADIOS2 will detect automatically the memory space

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

```
Kokkos::View<float **, MemSpace> gpuData("data", Nx, Ny);
 bpWriter.Put(data, gpuData);
```

- ADIOS2 saves pointers to data and copies data to internal CPU buffers
  - Computes metadata for each Get/Put using CUDA kernels

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

Overhead for detecting where buffers are allocated

Summary

- The ADIOS I/O framework
- **Kokkos applications using ADIOS**
  - Store and stream data
  - Campaigns and querying
  - Remote access to monitor the performance
- ADIOS using Kokkos
  - GPU-backend
  - Derived variables

~/work/adios/ADIOS2-main/ADIOS2-clean/build-kokkos/adios2 ⌥⌘2

gainaru@login05.frontier build]$

Frontier

ana@system76: ~/adios/Gray-Scott-Kokkos/build ⌥⌘4

ana@system76:~/adios/Gray-Scott-Kokkos/build$

System 76 (in Oak Ridge)

~/work/adios/Gray-Scott-Kokkos/build 🔔 ⌥⌘

95j][mac122730][±][main ↑1 {6} S:1 ?:5 ✗][~/work/adios/Gray-Scott-Kokkos/build]

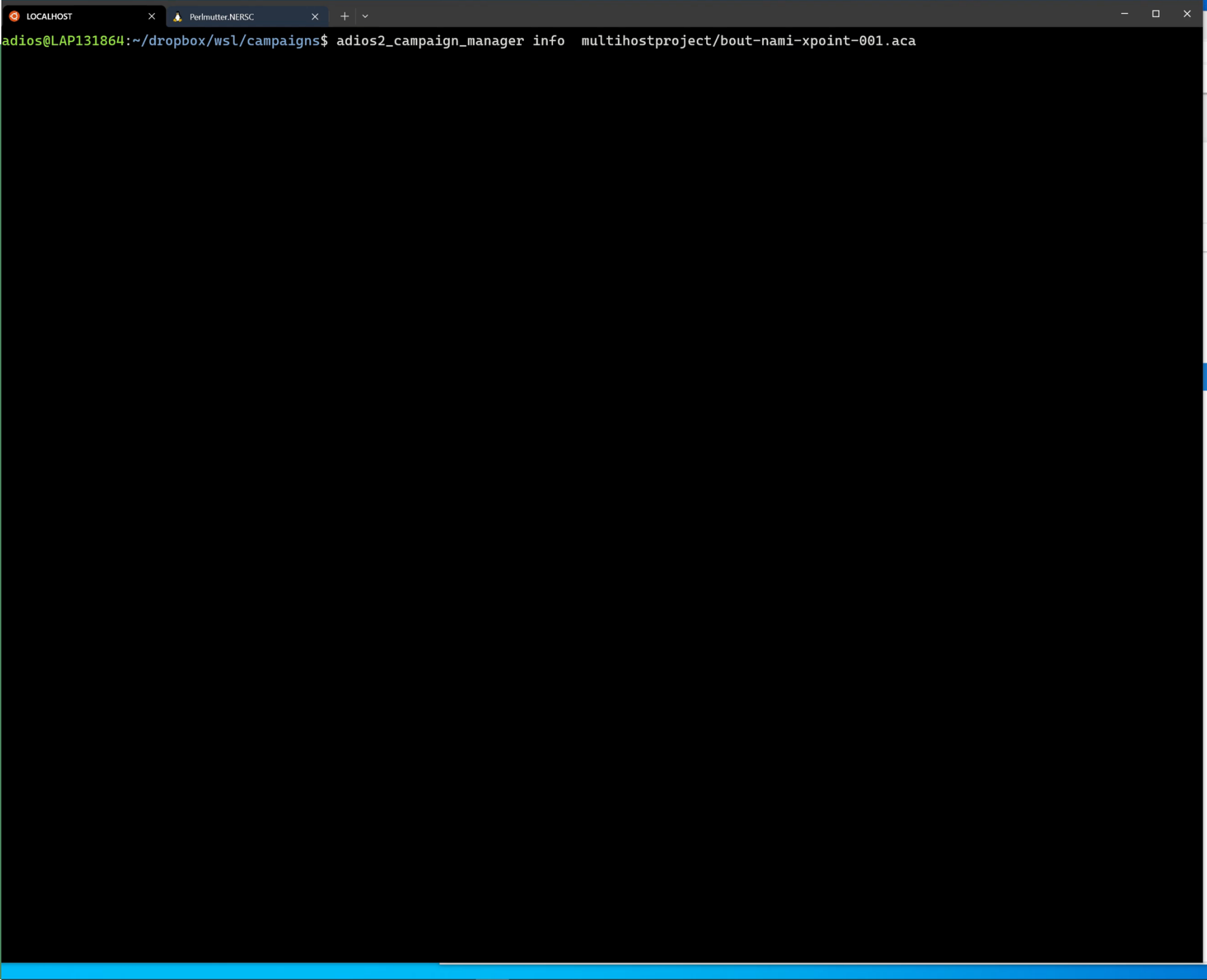
Ana's Laptop

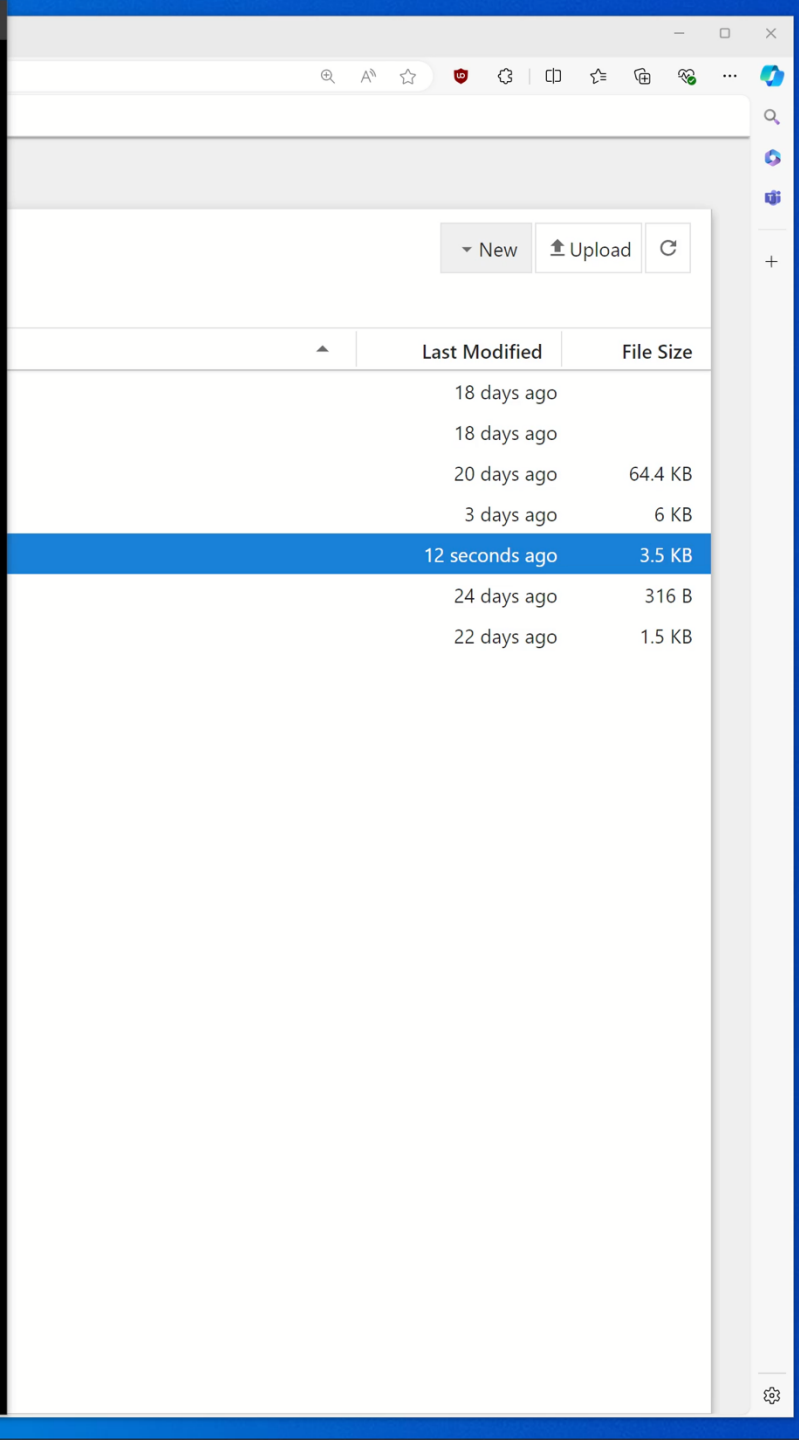
Remote access and campaigns

- Keep track of all datasets in a campaign
  - Location for remote access
  - Available variables and metadata
  - Metadata management using SQLite
- Query or subscribe across multiple streams / files
  - Remote / local access

Performance traces with TAU could also be included in the campaign

```
$ adios2_campaign_manager info paper_sc24.aca

info archive
ADIOS2 Campaign Archive, version 0.1, created on 2024-07-23 16:38:13
hostname = OLCF     longhostname = frontier.olcf.ornl.gov
    dir = /path/on/frontier/to/run1
            dataset = sim.bp    created on 2024-07-23 13:35:18
            dataset = pdf.bp    created on 2024-07-23 14:21:42
hostname = NERSC     longhostname = perlmutter-p1.nersc.gov
    dir = /path/on/perlmutter/to/run2
            dataset = sim.bp    created on 2024-07-22 10:25:05
```


LOCALHOST
Perlmutter.NERSC
adios@LAP131864:~/dropbox/wsl/campaigns$ adios2_campaign_manager info multihostproject/bout-nami-xpoint-001.aca

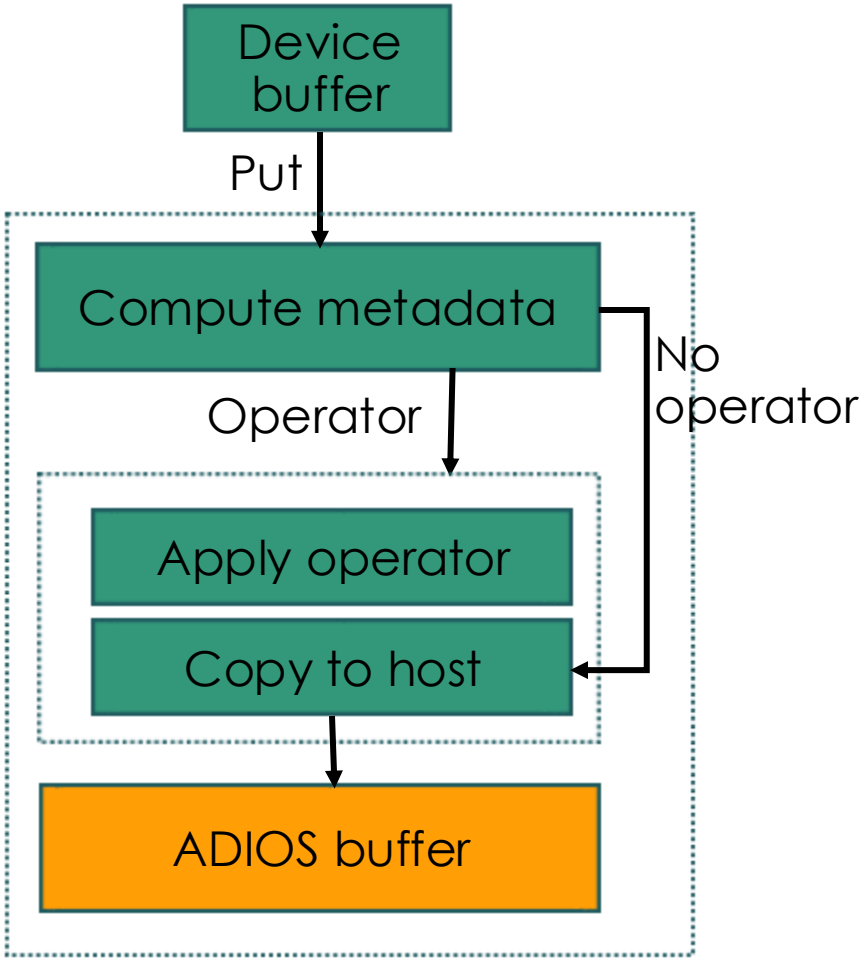

New
Upload
Last Modified
File Size
18 days ago
18 days ago
20 days ago
64.4 KB
3 days ago
6 KB
12 seconds ago
3.5 KB
24 days ago
316 B
22 days ago
1.5 KB

Summary

- **The ADIOS I/O framework**
- **Kokkos applications using ADIOS**
  - Store and stream data
  - Campaigns and querying
  - Remote access to monitor the performance
- **ADIOS using Kokkos**
  - GPU-backend
  - Derived variables

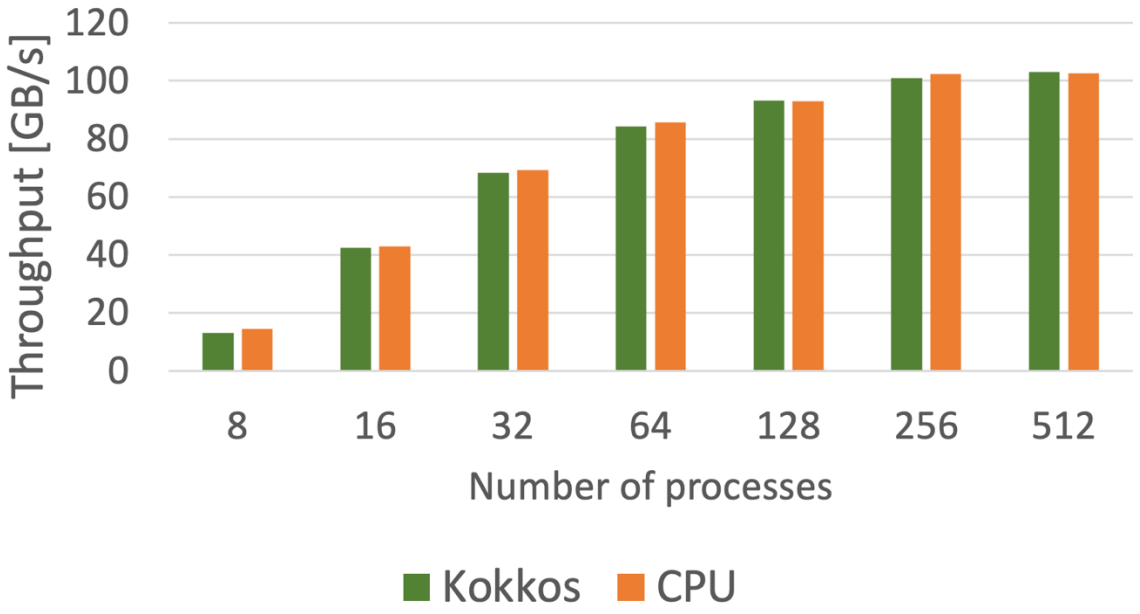
GPU-aware ADIOS2

- Publish/subscribe directly GPU pointers
  - For Kokkos::View we extract the memory space and layout
- Internals
  - Copy the data to adios2 internal buffers
  - Compute metadata
    - Min/Max of blocks of data
  - Layout is handled by the adios2 variable dimensions



Performance

- When not collecting any metadata
  - Kokkos has the same performance as the CPU backend



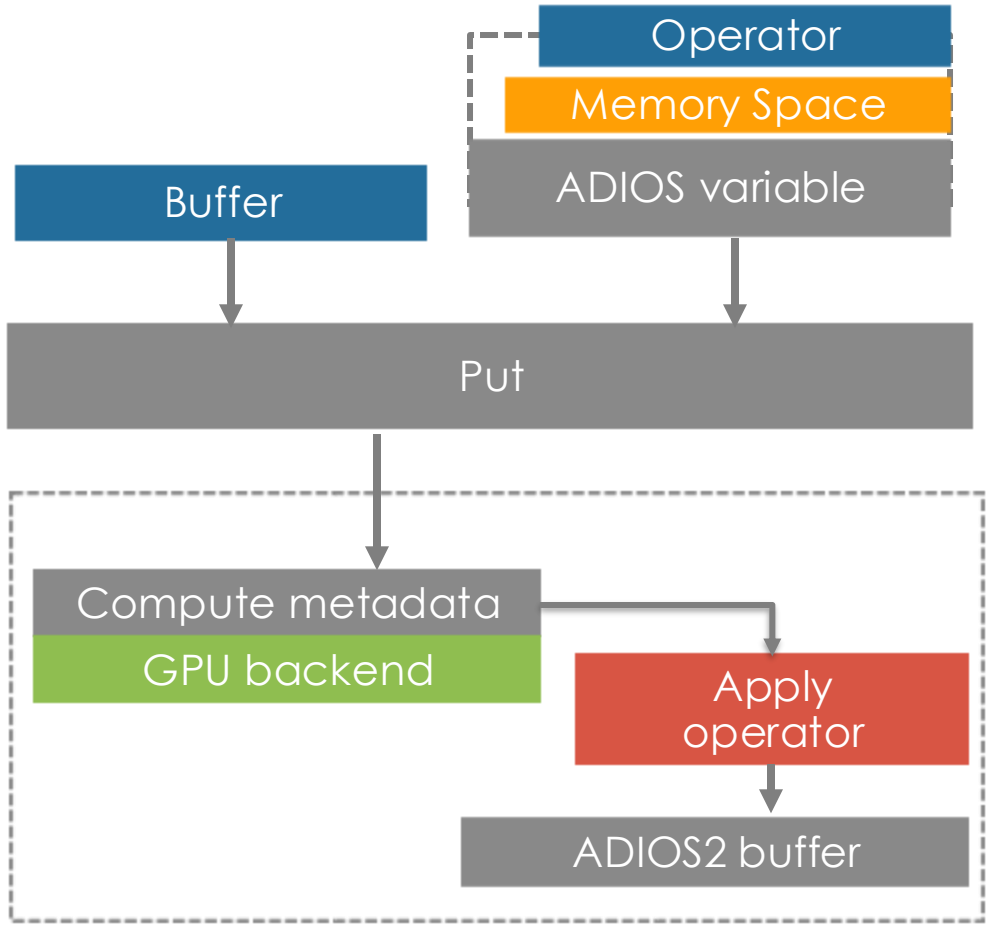
- Memory footprint
  - CPU backend
    - For chunks > 4MB
      - Move data directly from the user buffer
  - Kokkos backend
    - ADIOS2 always uses internal buffers to hold the GPU data
    - Currently we do not handle memory accessible from the Host

* Results for weak scaling on Summit, 64GB of data per node
* We measure the overall write throughput for all nodes.

Compression with GPU-aware I/O

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

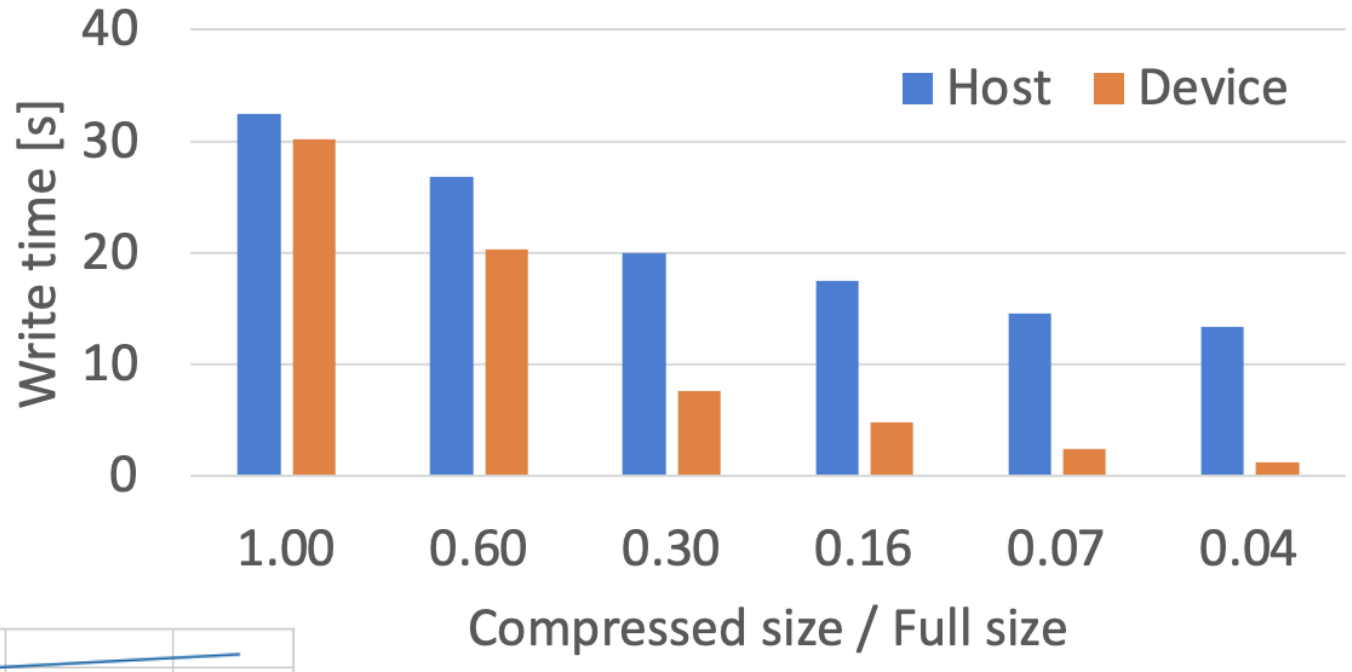
```
auto var = io.DefineVariable<double>(”test", shape, start, count);

// 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);
```



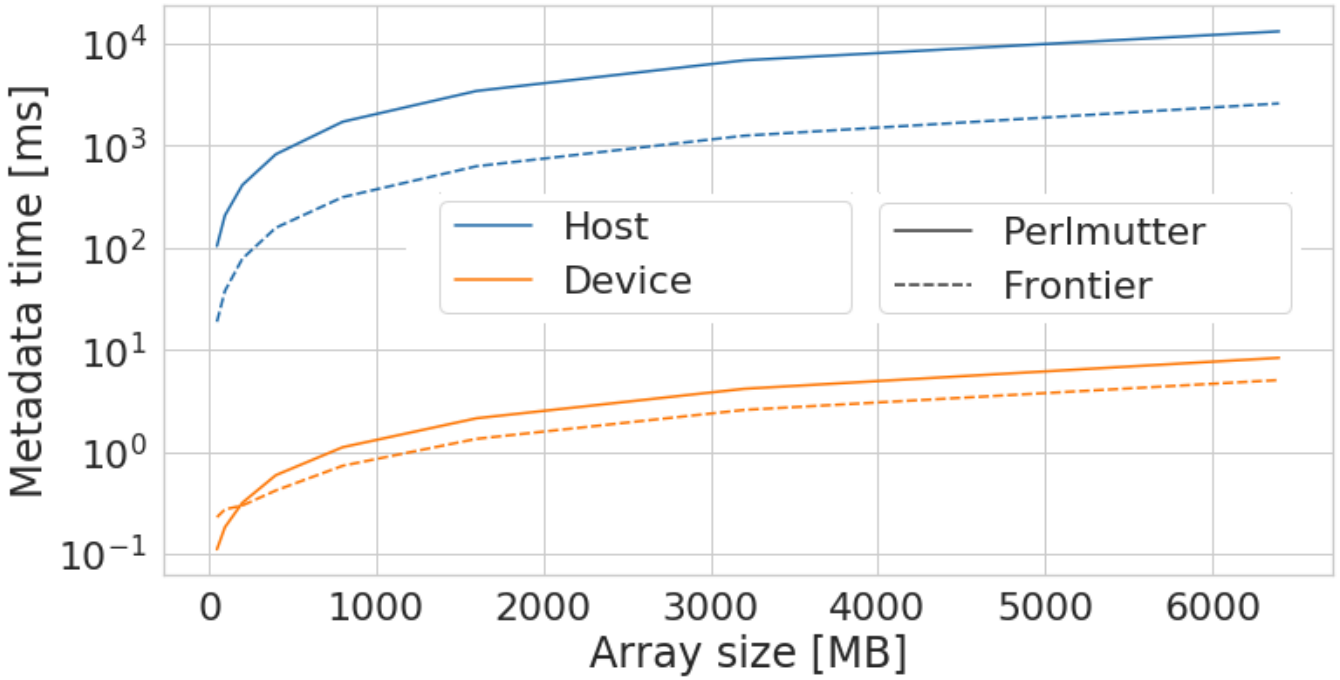
Operators that support GPU buffers:

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

Basic performance

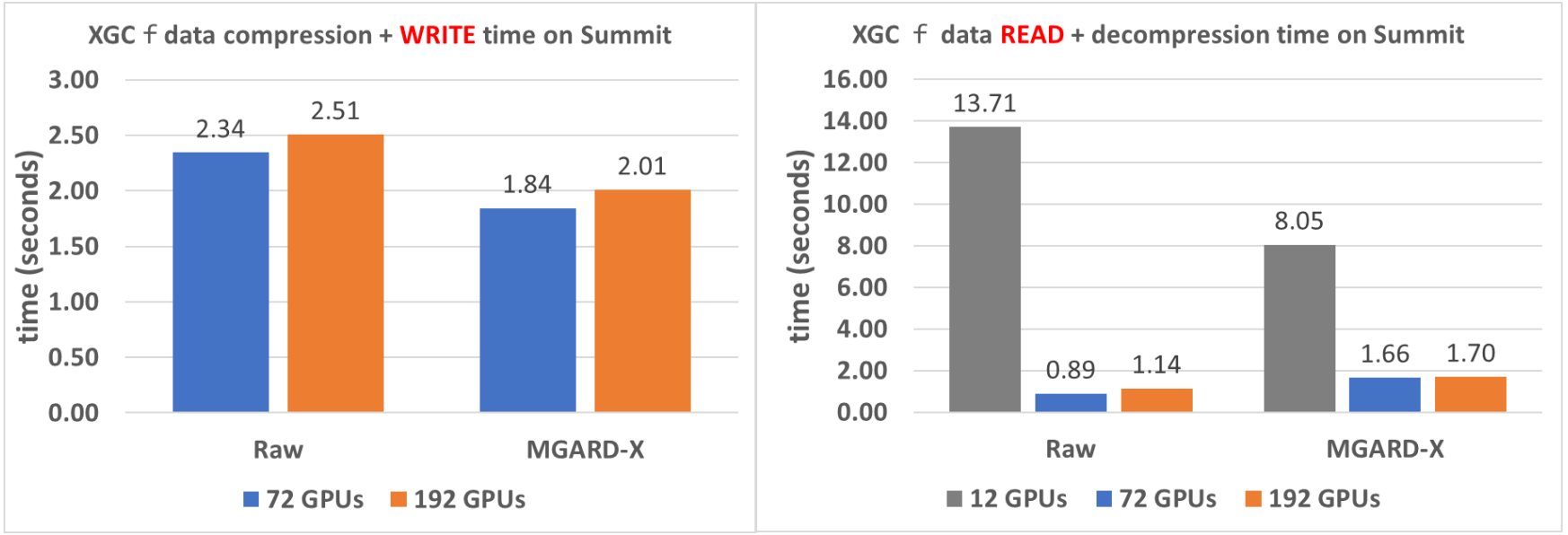


ZFP compression time when writing compressed data for different compression rates using the CPU vs Kokkos backends



Metadata time when computing the min/max using the CPU vs Kokkos backends

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.

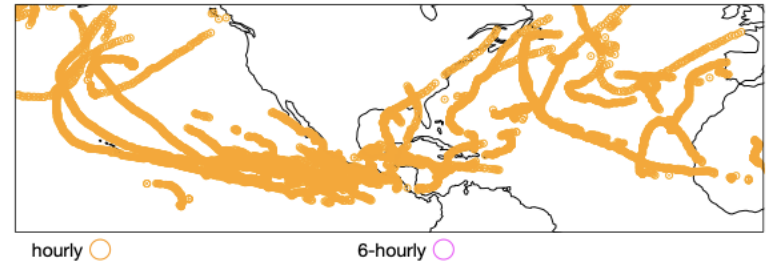
Norbert Podhorszki, Scott Klasky, Ana Gainaru, Qian Gong, Jieyang Chen, Sanjay Ranka. The Benefits of Data Reduction for Fusion Datasets. Poster at ICDDPS-4 conference, Apr 2023, Okinawa, Japan

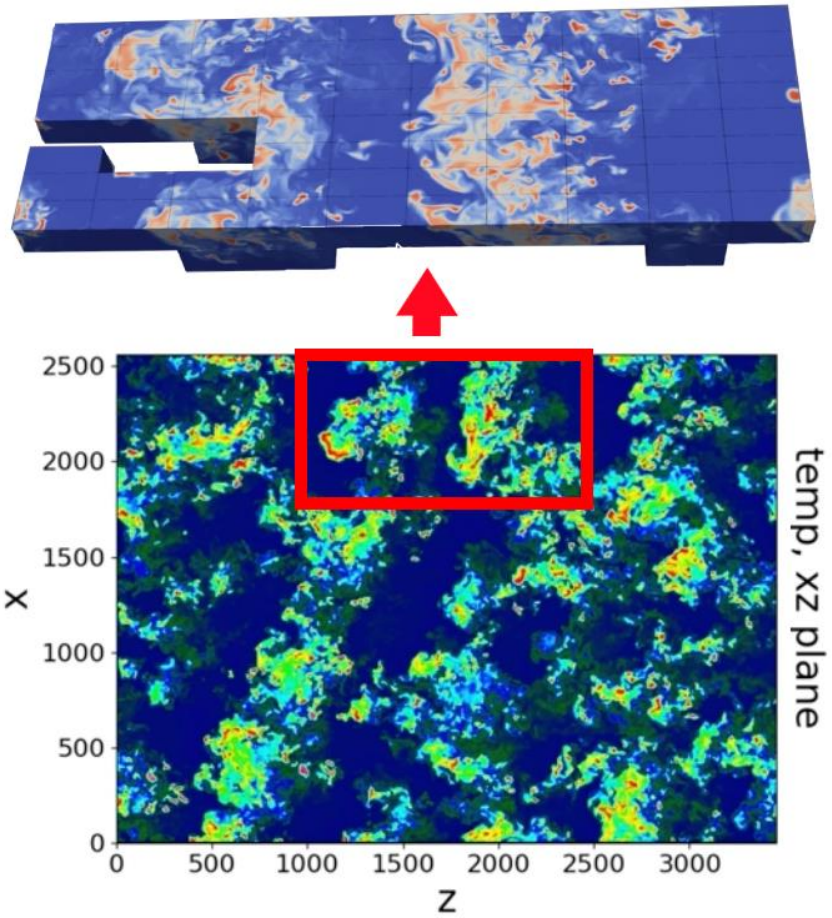
Derived quantities

- Data or quantities of interest
  - Not specifically the result of the principal calculation of the application
  - Can be computed or extrapolated (derived) from primary data

- Why are they needed
  - Queries and analysis

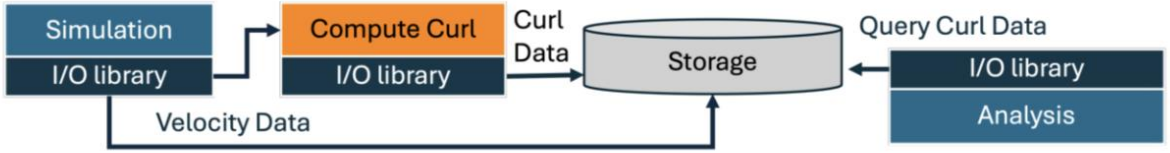
- Example S3D
  - 1.5 TB per step (including temp, velocity, species information, etc
  - Visualize 2D slices of temperature
    - Query on magnitude (instead of velocity) to identify areas of interest
  - Analysis in-situ or on a remote server (or scientist laptop)





Derived variables

- Offload derived variable computation to ADIOS2
  - **Writer side** solutions
    - **Store**
    - Workflows include analysis computing and storing the required derived data
  - **Reader side** solutions
    - **Expression**
    - Analysis codes computing derived variables on the fly (e.g. Paraview)
  - **Hybrid**
    - **Stats**
    - Store only stats for derived variables



(a) Write side solution

(b) Read side solution

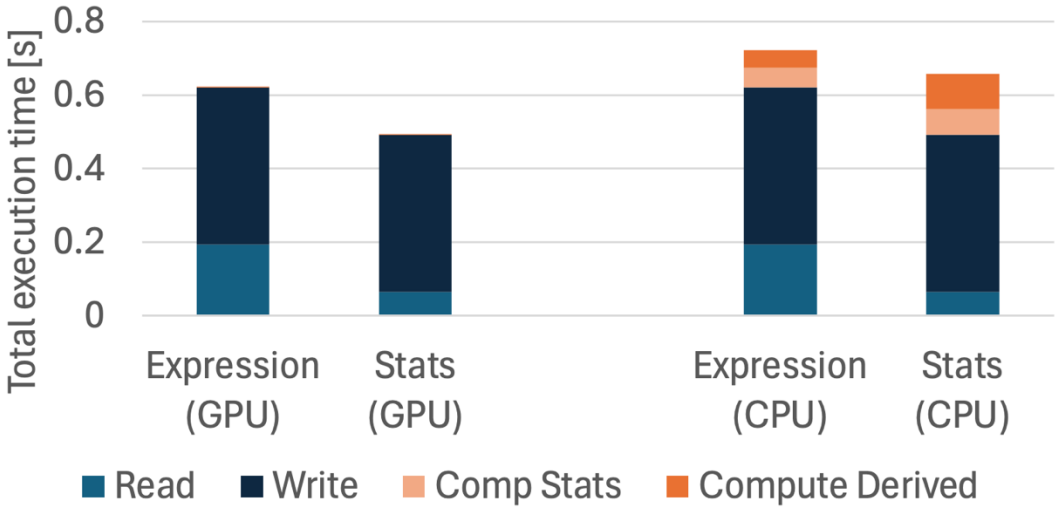
```
auto velocity = bpIO.DefineVariable<float>(
                              "velocity", shape, start, count);

bpIO.DefineDerivedVariable("derived/magnitude",
                           "v = velocity \n"
                           "magnitude(v)",
                           adios2::DerivedVarType::StatsOnly);
```

S3D

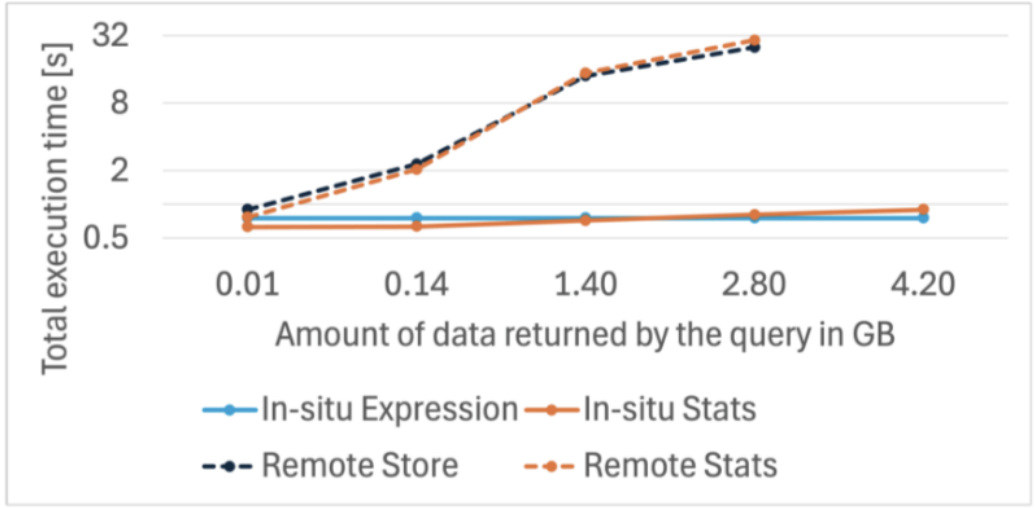
- The magnitude derived variable has a size equal to the number of particles
  - The Store strategy adds 64 GB of data for each simulated step
  - For 900 ranks the stats are12 MB

- The Expression strategy requires storing 256 GB on the remote site

The Stats strategy has similar results with Expression for in-situ and Stats for remote access



(a) Execution time on Frontier for writing S3D data and reading in-situ a small portion of the data (around 1.5%)

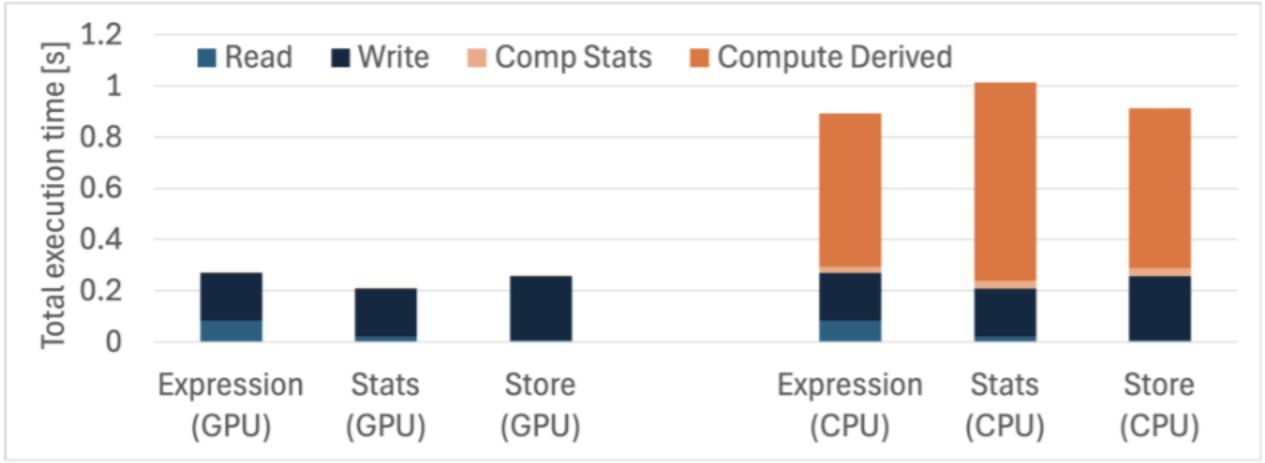


(b) In-situ and remote analysis of S3D data when the study includes querying multiple areas of interest

E3SM

- The size of the curl variables is 4 GB
  - The Store strategy adds 28 GB
  - The stats for 96 ranks is 1 MB
- Stats strategy is 1.5x slower
  - Curl has high complexity
  - The curl values are needed by the analysis

(a) Total time as the analysis is querying more data



(b) Breakdown of the cost for 1GB of data

Conclusion

ADIOS

- Applications using Kokkos can use ADIOS directly to stream/store Kokkos::Views
  - TAU has an the option of publishing ADIOS variables which could allow monitoring the performance in real time
  - Remote access and querying is available post-mortem or in real time

- ADIOS uses a Kokkos backend
  - Allows GPU buffers to be transferred to the compression library

- Derived variables
  - Computed on the GPU, hiding the computation cost
  - Allows for hybrid and adaptive solutions

- Some links
  - https://adios-io.org/
  - https://github.com/ornladios/ADIOS2

Ana Gainaru
gainarua@ornl.gov