

ADIOS2 with Kokkos backend

Ana Gainaru, Norbert Podhorszki, Greg Eisenhauer,
Vicente Bolea, Scott Klasky

Kokkos User Group Dec 13, 2023

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



What to expect for the next 20 minutes

- What is ADIOS2
- Kokkos within ADIOS2
 - Kokkos applications using ADIOS2
 - API for storing/streaming files
 - Streaming Kokkos::View **--DEMO--**
 - ADIOS2 using Kokkos
 - Some performance numbers
 - Querying on derived variables **-- DEMO --**
- Kokkos wish list

What is ADIOS2



- 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

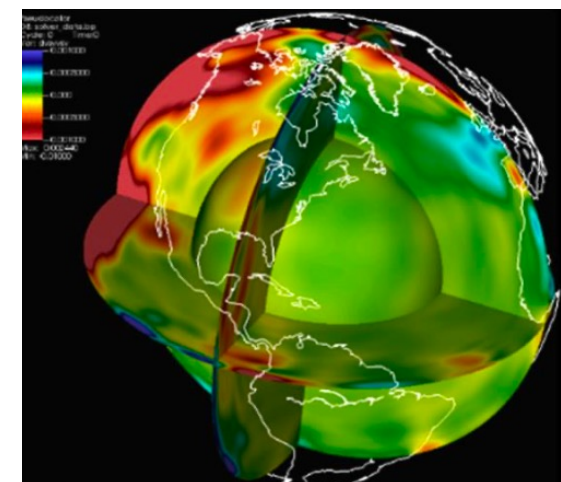
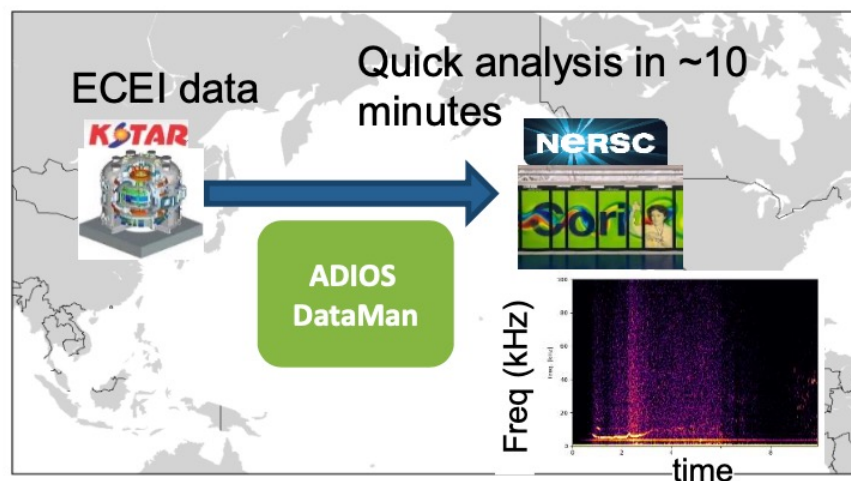
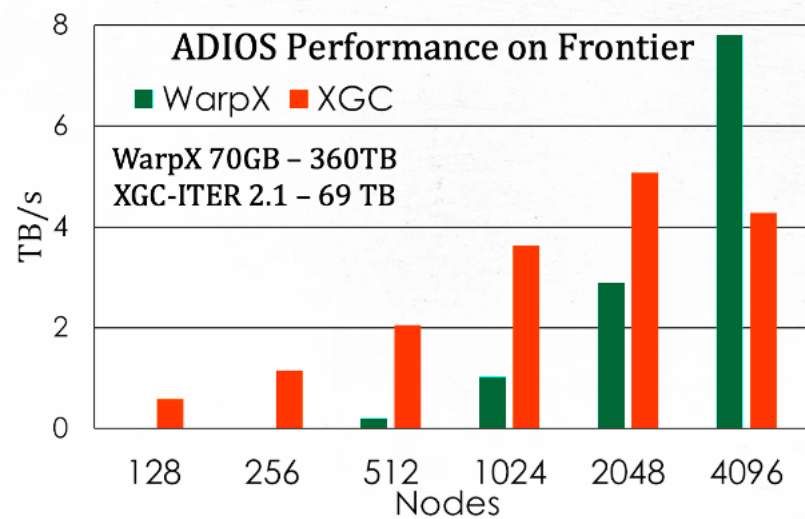
Application	Nodes/GPUs	Data Size per step	I/O speed
SPECFEM3D	3200/19200	250 TB	~2 TB/sec
GTC	512/3072	2.6 TB	~2 TB/sec
XGC	512/3072	64 TB	1.2 TB/sec
LAMMPS	512/3072	457 GB	1 TB/sec

<https://github.com/ornladios/ADIOS2>

Contributors

A few of our applications

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



A bit more on ADIOS2



- **Publish/subscribe API**

- Define an ADIOS variable
 - With a certain global and local shape
- Add an operator
- Publish data
 - Aggregated in internal buffers
- Subscribe to data

- **I/O engines**

- Keep the same code and switch the data management strategy

```
auto adiosVar = io.DefineVariable<float>(
    "var_name", shape, start, count);

adios2::Operator mgardOp =
    adios.DefineOperator("mgardCompressor",
adios2::ops::LossyMGARD);

adiosVar.AddOperation(mgardOp,
    {{adios2::ops::mgard::key::tolerance, tolerance}});

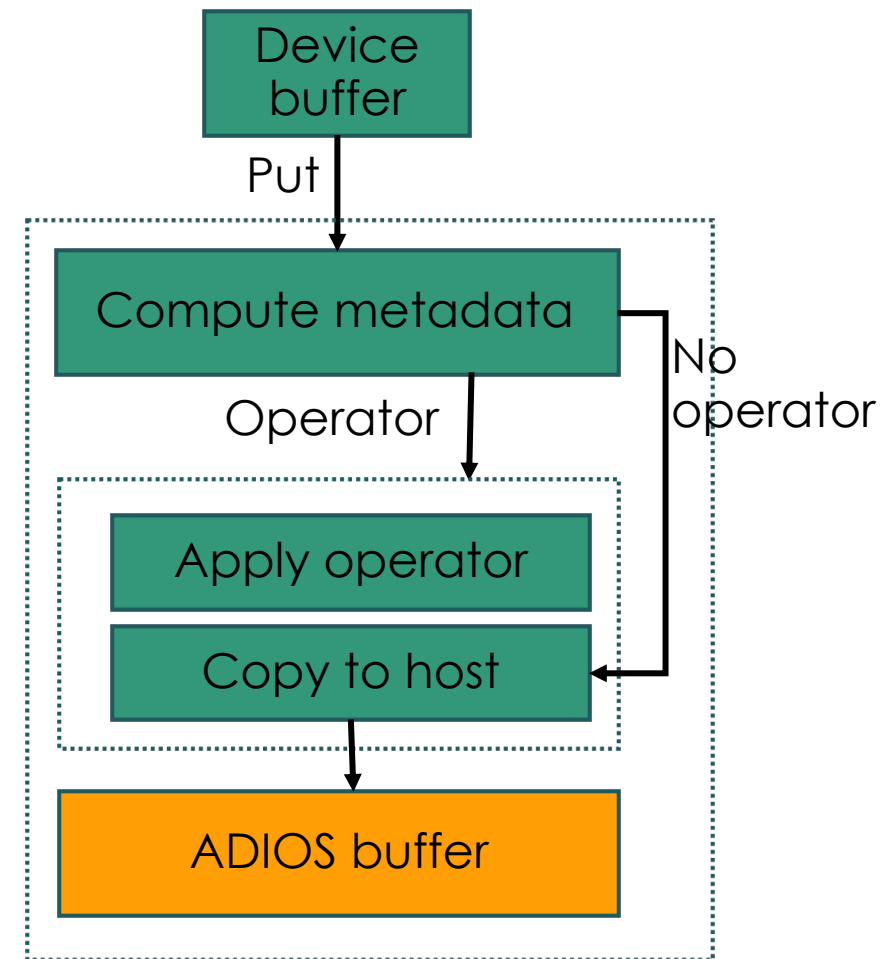
bpWriter.Put(adiosVar, userData);
```

```
$ bpls gs.bp -l
double U 100*{64, 64, 64} = 0.0907619 / 1
double temp 100*{64} = 0 / 26.74846
int32_t step 100*scalar = 10 / 1000

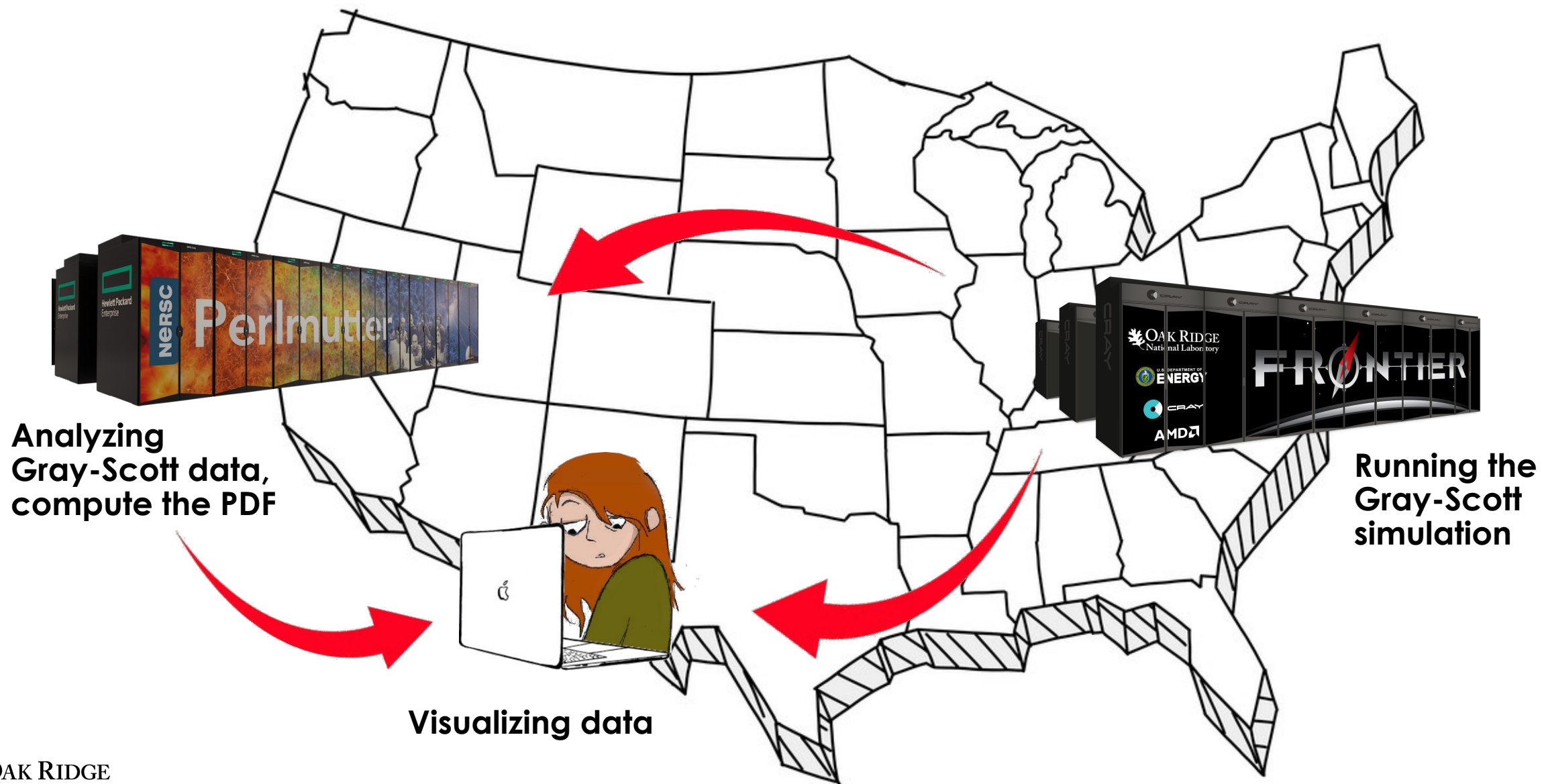
$ bpls gs.bp -la
double temp_units attr = "Celsius"
```

GPU-aware ADIOS2

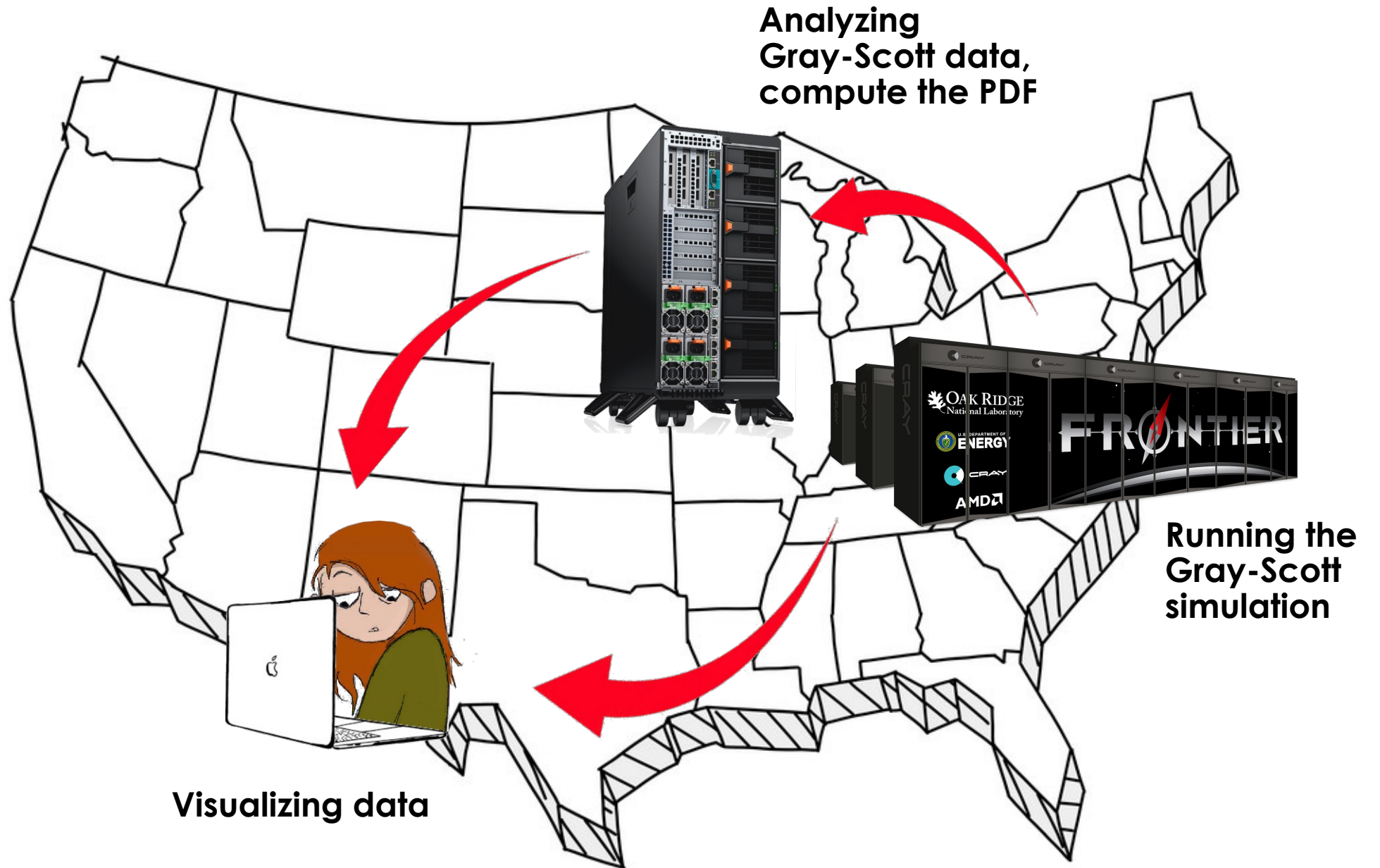
- Publish/subscribe directly GPU pointers
 - For Kokkos::View we extract the memory space and layout
- Internals
 - Copy the data to adios2 internal buffers
 - Compute metadata
 - Min/Max of blocks of data
 - Layout is handled by the adios2 variable dimensions



Demo remote access



Demo remote access



Demo 1

The image is a composite of several elements related to a scientific simulation:

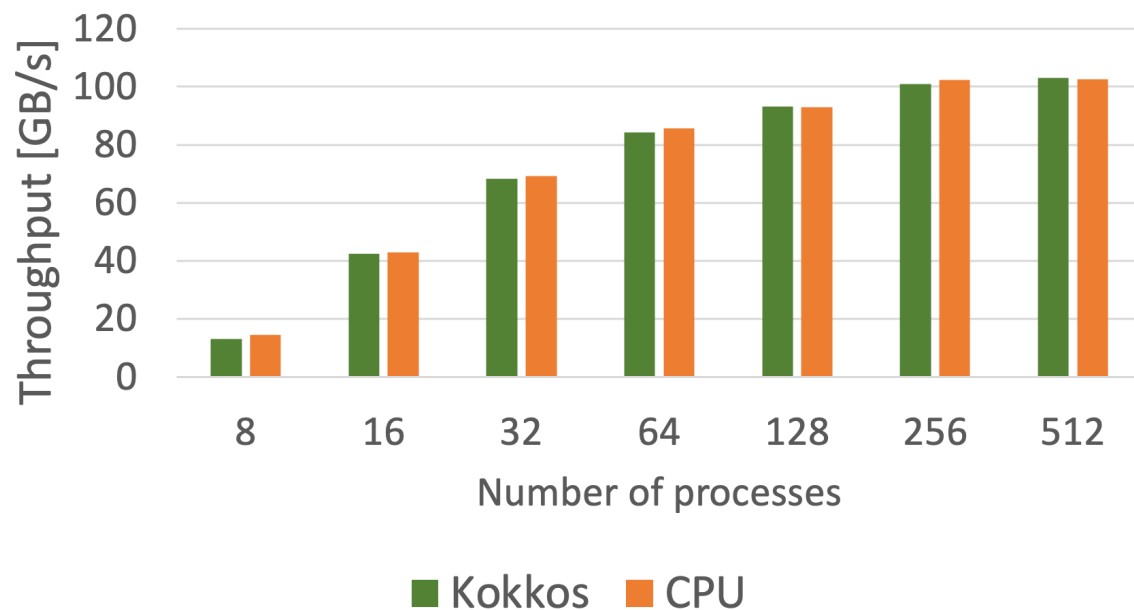
- Figure 1:** A 2D heatmap titled "U, yz plane, step 84". The x-axis is labeled "y" and the y-axis is labeled "z", both ranging from 0 to 60. The plot shows four distinct peaks of high intensity (red/yellow) arranged in a 2x2 grid, with a color scale below ranging from 0.4 to 0.8.
- Terminal Windows:** Several terminal windows are visible. One shows a table of system resources:

1	N/A	N/A	2788	G	/usr/lib/xorg/Xorg	4MIB
1	N/A	N/A	166797	G	/usr/lib/xorg/Xorg	4MIB
2	N/A	N/A	2788	G	/usr/lib/xorg/Xorg	53MIB
	N/A	N/A	166797	G	/usr/lib/xorg/Xorg	155MIB
	N/A	N/A	166968	G	/usr/bin/gnome-shell	10MIB
	N/A	N/A	167503	G	/usr/lib/firefox/firefox	39MIB

Another terminal window shows the execution of a script: `ana@system76: ~/adios/Gray-Scott-Kokkos/build$./gray-scott_pdf-calc gs.bp gs-pdf.bp 100`. A third terminal window shows the command `ana@system76: ~/adios/Gray-Scott-Kokkos/build$`.
- System 76:** A server rack is shown with the text "System 76 (in Oak Ridge)".
- Frontier:** A server rack is shown with the text "Frontier".
- Ana's Laptop:** A laptop is shown with the text "Ana's Laptop" and a cartoon character sitting at it.

Performance

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

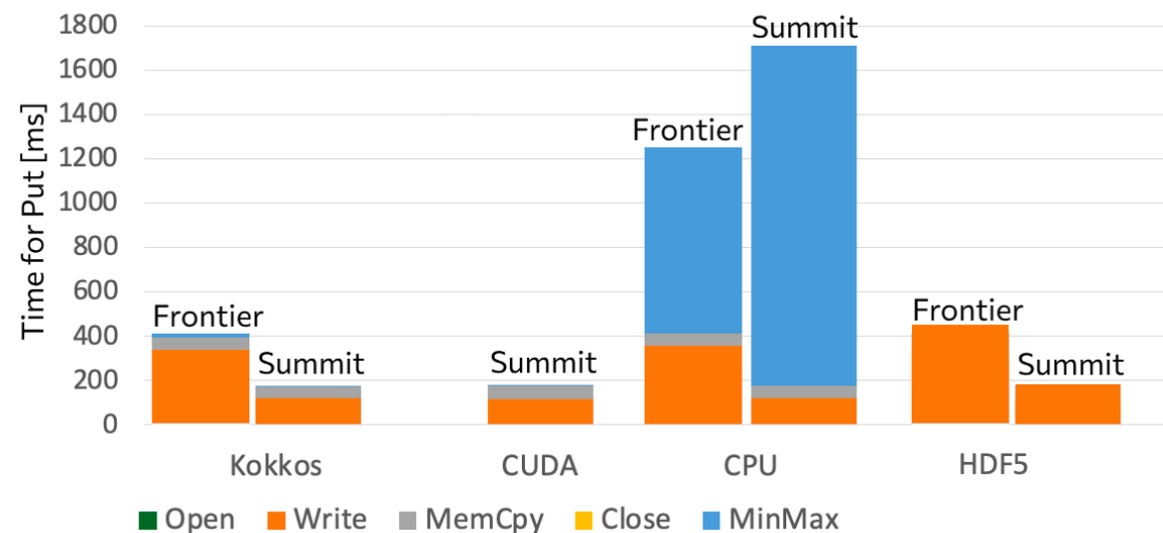
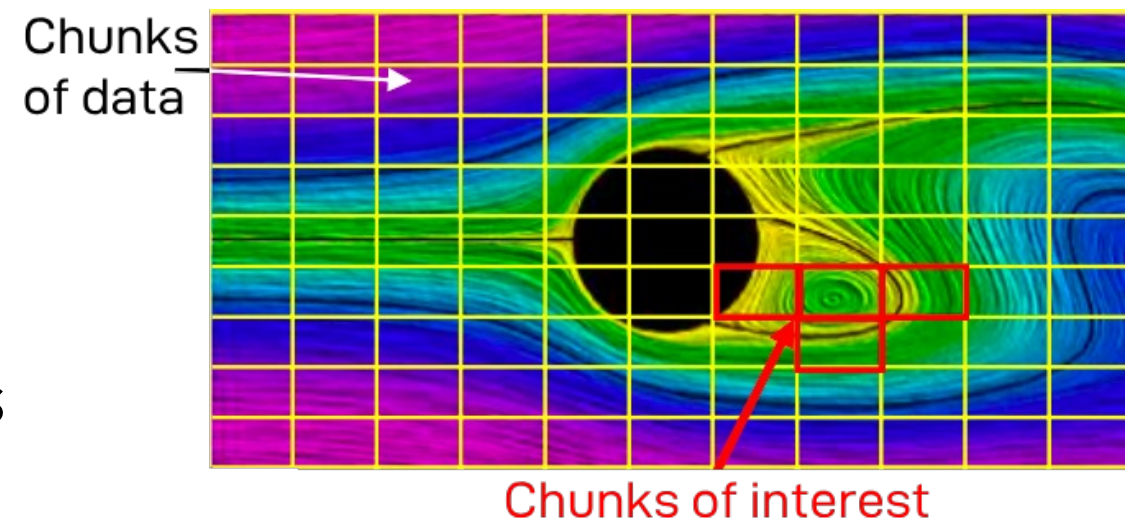


- * Results for weak scaling on Summit, 64GB of data per node
- * We measure the overall write throughput for all nodes.

- Memory footprint
 - CPU backend
 - For chunks $> 4\text{MB}$
 - 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

Query

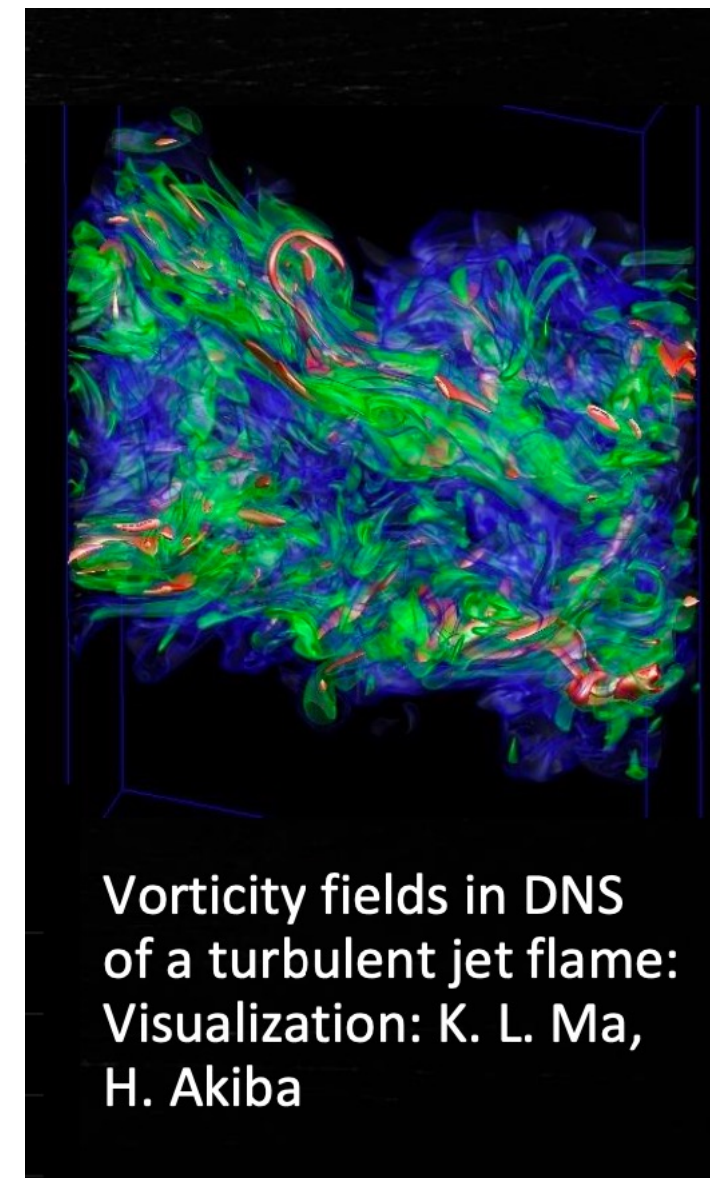
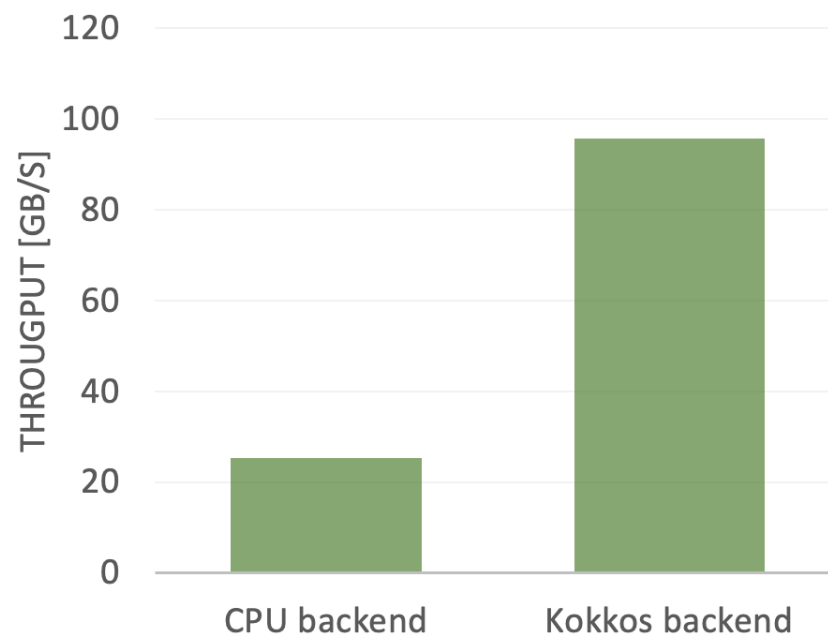
- Collecting min/max
 - Allows querying on the read side
 - Granularity of the chunk size affects how much more data we store
 - Trade-off with how much data is read
- Applications can define **Quantities of Interest**
 - E.g. curl, magnitude, derivative
 - ADIOS2 compute and stores metadata for derived variables



Performance

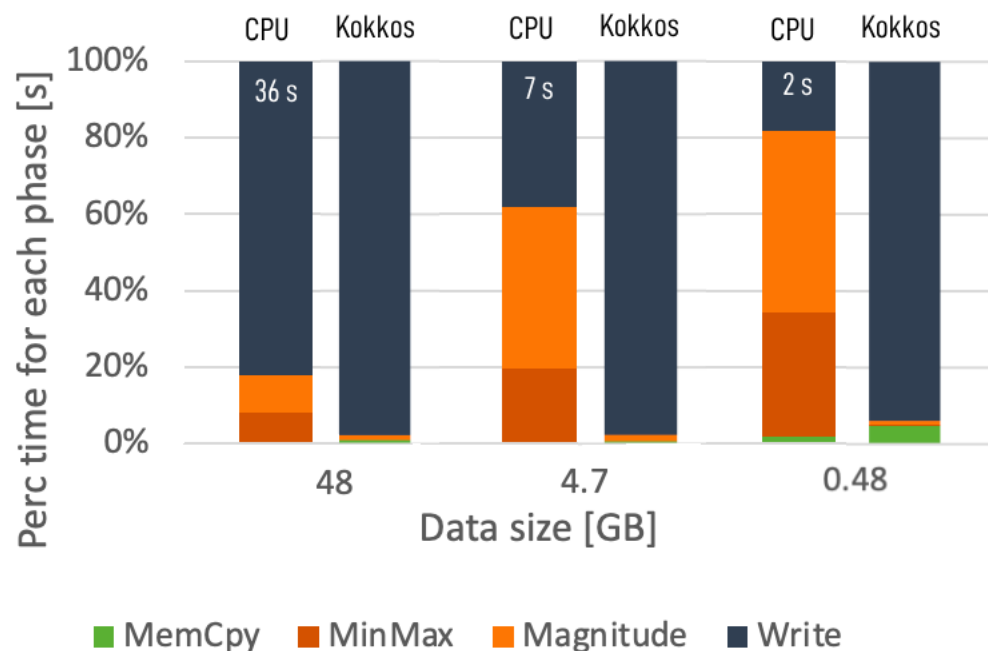
- Computing min/max and magnitude for S3D data
 - 229 nodes (7200 processes) on Frontier

Total amount of data: 1.582 TB



Performance

- Profiling the CPU/Kokkos backends



```
parallel_for("ADIOS::ComputePartialMagnitude",
            Kokkos::MDRangePolicy<Kokkos::Rank<2>>> (
                {0, 0}, {dimensions, size}),
            KOKKOS_LAMBDA(int i, int j)
            {
                auto val = data(i, j);
                Kokkos::atomic_add(&mag(j), val * val);
            });

parallel_for("ADIOS::ApplySqrtToMagnitude",
            Kokkos::RangePolicy<>(0, size),
            KOKKOS_LAMBDA(int i)
            {
                mag(i) = Kokkos::sqrt(mag(i));
            });
```

- Memory footprint

Size with magnitude

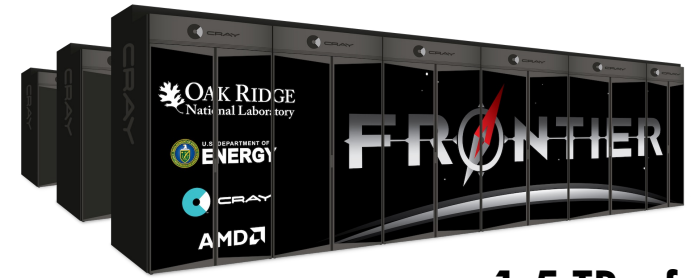
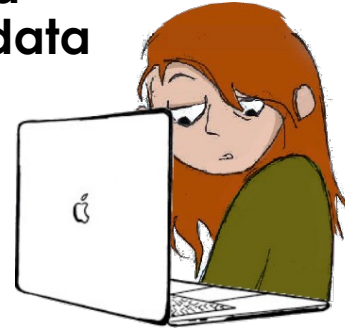
1.518 TB

Size without magnitude

1.582 TB

Demo query

Query and visualize data



1.5 TB of S3D data with derived metadata



A composite image showing the workflow. At the top, two terminal windows are open. The left terminal shows a shell prompt and a directory listing. The right terminal shows a command being executed, resulting in a 3D visualization of data. Below the terminals is a large window displaying a 3D visualization of the data. The visualization shows a complex, multi-layered structure with a color gradient from blue to orange/red, representing different data values. A coordinate system with X, Y, and Z axes is visible at the bottom left of the visualization window.

Wish list from Kokkos

- Would be great
 - Get what device was set
 - Math functions

- Not very probable
 - Detecting if a pointer was allocated on the device
 - I/O direct to storage
 - Only for CUDA

```
#ifdef ADIOS2_HAVE_KOKKOS_CUDA
int device_id;
cudaGetDevice(&device_id);
settings.set_device_id(device_id);
#endif
```

Thank you



- Tutorial

https://users.nccs.gov/~pnorbert/ADIOS_tutorial_SC23.pdf

- <https://adios-io.org/>

- <https://github.com/ornladios/ADIOS2>

Ana Gainaru

gainarua@ornl.gov



Backup slides

Kokkos applications using ADIOS2

- Layout

- ADIOS2 variables use by default LayoutRight (except for Fortran codes)
- Based on the memory space
 - Internally we reverse the dimensions of the adios2 variable
 - Zero cost
 - Put/Get on different memory space might see transposed data
- Write/read files, streaming

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

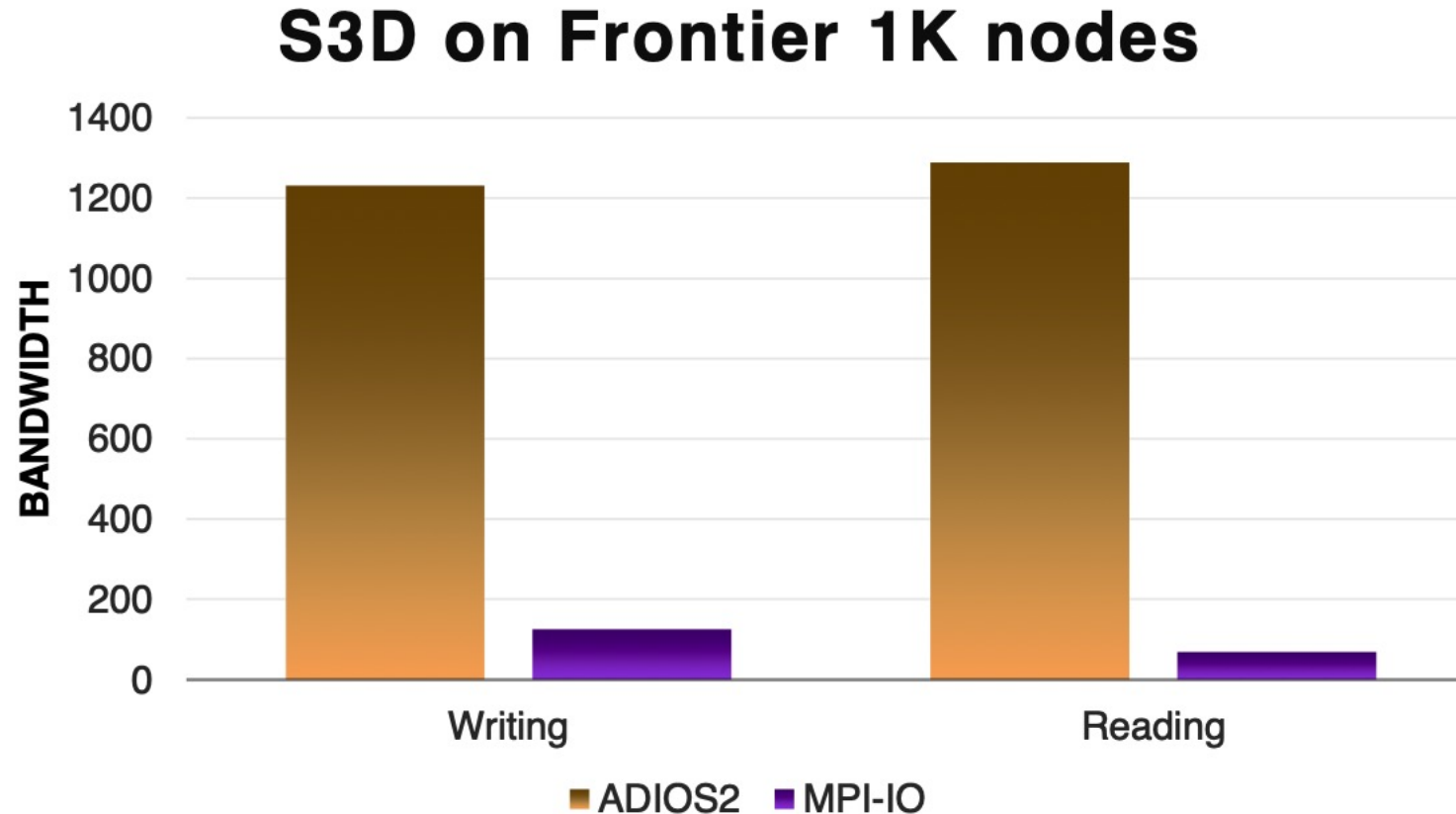
Kokkos::parallel_for("initializeData",
    Kokkos::MDRangePolicy< Kokkos::Rank<2> >({0, 0}, {Nx, Ny}),
    KOKKOS_LAMBDA(int x, int y)
    {
        data(x, y) = static_cast<float>(x);
    });
```

```
$ ./ReadKokkosView
Read on memory space: HIP
1 1 1 1
2 2 2 2
3 3 3 3
4 4 4 4
5 5 5 5
6 6 6 6
```

```
$ ./bin/bpls -I Kokkos.bp/ -d data -n 6
float data {3, 6} = 1 / 6
(0,0) 1 2 3 4 5 6
(1,0) 1 2 3 4 5 6
(2,0) 1 2 3 4 5 6
```

S3D Frontier results

1k nodes, 32 processes per node, ~6.5 GB/node



DataMan code

- ZeroMQ for data transfer
- Parameters
 - TransportMode: fast or reliable
 - MaxStepBufferSize: the default buffer size is 128 MB
 - Threading: true for reader, false for writer

```
adios2::ADIOS adios(MPI_COMM_WORLD);
adios2::IO dataManIO = adios.DeclareIO("WritePDF");
dataManIO.SetEngine("DataMan");
dataManIO.SetParameters({{"IPAddress", "127.0.0.1"},
                          {"Port", "12306"},
                          {"Timeout", "5"},
                          {"RendezvousReaderCount", "1"}});

dataManWriter.Put(adiosVar, KokkosData, adios2::Mode::Sync);
```