ADIOS at scale

Presenter: Norbert Podhorszki
Oak Ridge National Laboratory, Computer Science and Mathematics Division

Excalibur - US extreme data workshop and tutorials

Cambridge, UK

July 21, 2023

I/O Challenges

- Write/read a TB dataset frequently enough for science
- Write a 60 TB checkpoint every half an hour (particle codes)
- Write/Read data from O(100k) processes (CPU-only codes, Arm systems)
- Collect data from thousands of applications to train AI/ML (DeepDriveMD)
- Couple two/more parallel applications in every iteration (Fusion Whole Device Modeling)
- Write a few PBs of data produced by an ensemble occupying the entire machine (Earth-scale tomography)
- Staging data processing pipeline that can process data TB/s (SKA epoch pipeline on Summit)
- Replace costly parts of simulation with a surrogate model tightly coupled to the simulation using staging (correlated electrons simulation, WRF, fusion, particle accelerators)
- Scalable I/O library in number of processors, number of variables, number of steps, frequency of steps

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

Approach

- Easy-to-use, high performance I/O abstraction to allow for on-line/off-line memory/file data subscription service
- Sustainable solution that works 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

API:

- Each process describes what data it has/needs
- Forced to declare the I/O phase

I/O strategy:

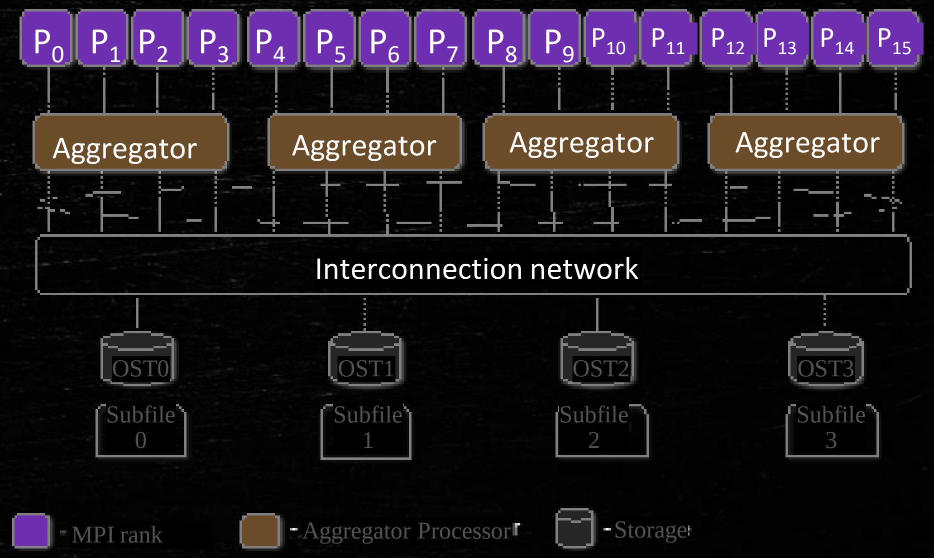
Media, aggregation, number of sub-files, target file-system hacks, file format not expressed at the code level

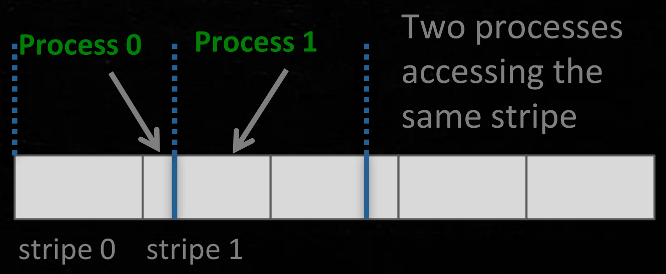
Godoy, W. F., Klasky, S. ,et al. (2020). ADIOS-2: The adaptable input output system. a framework for high-performance data management. SoftwareX, 12, 100561.

File format + API designed for scalability

- Avoid latency (of small writes): Buffer data for large bursts – use a type of self-describing log file format
- Avoid accessing a file system target from many processes at once
  - Aggregate to a small number of actual writers:
  - Avoid lock contention
  - Striping correctly & writing to subfiles
- Avoid global communication wherever possible

Liu, Q., Klasky, S., et al. "Hello ADIOS: the challenges and lessons of developing leadership class I/O frameworks." Concurrency and Computation: Practice and Experience 26.7 (2014): 1453-1473.





Application	Nodes/GPUs	Data Size/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

BP File format

- Dataset is a directory, not a single file
  - Data is stored in multiple files
- Arrays (variables) are stored as chunks, that can be compressed/decompressed individually
  - One arrays spans all files, multiple arrays (chunks) are stored in each file
  - Don't confuse this with HDF5/Zarr's static chunk concept
- Metadata is stored separately from data (separate files)
- Metadata is a binary format and is big
  - information per chunk has to be stored to be able to find, locate and retrieve it

Why use ADIOS over other technologies

- Scalable performance
  - designed to be a parallel IO library,
  - one application can utilize the bandwidth of the entire file system (when running at large scale)
  - proven by users for many PBs datasets, tens of thousands of nodes, daylong runs, thousands of steps
- Scalable (distributed) compression routines
  - lossless and lossy, blosc, bzip2, zfp, sz, mgard
- Safe to append data into single dataset during run, at scale
  - new writes cannot corrupt existing data
  - entire output “step” is either in or not
  - application failure will not corrupt previously written data/file
- Able to read data from files being written
  - previous steps only, not the one under construction
  - stable semantics: data in a published step is always completely available

ADIOS Scaling for large parallel file systems

- There are a **few** options for changing the performance of ADIOS for your code on all HPC systems
  - Aggregation – choosing the number of sub-files for maximizing the filesystem bandwidth
  - Appending multiple steps into a single output for minimizing the filesystem metadata overhead
  - Asynchronous write with BP5 engine
  - Choosing the “best” ADIOS engine/storage system (staging, NVRAM-flush) for minimizing the variability

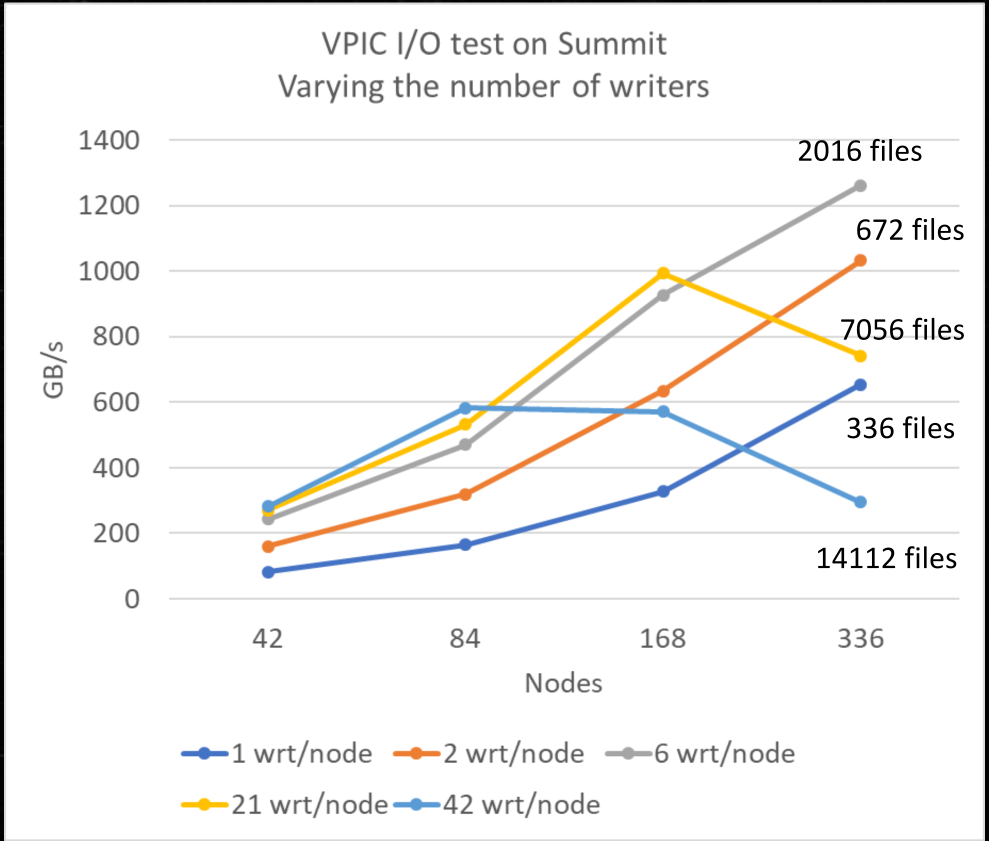
ADIOS Scaling for large parallel file systems: number of files

- Not good:
  - Single, global output file from many writers (or for many readers)
    - Bottleneck at file access
  - One file per process
    - Chokes the metadata server of the file system at create
    - Reading from different number of processes is very difficult
- Good:
  - Create K subfiles where K is proportioned to the capability of the file system, not the number of writers/readers
- ADIOS BP engine options
  - **NumAggregators**
  - **AggregatorRatio**
- ADIOS default settings
  - One file per compute node

```
<io name="SimulationOutput">
    <engine type="BP5">
        <parameter key="NumAggregators" value="2048"/>
    </engine>
</io>
```

Example: VPIC I/O test on Summit

chart courtesy of Junmin Gu, LBNL

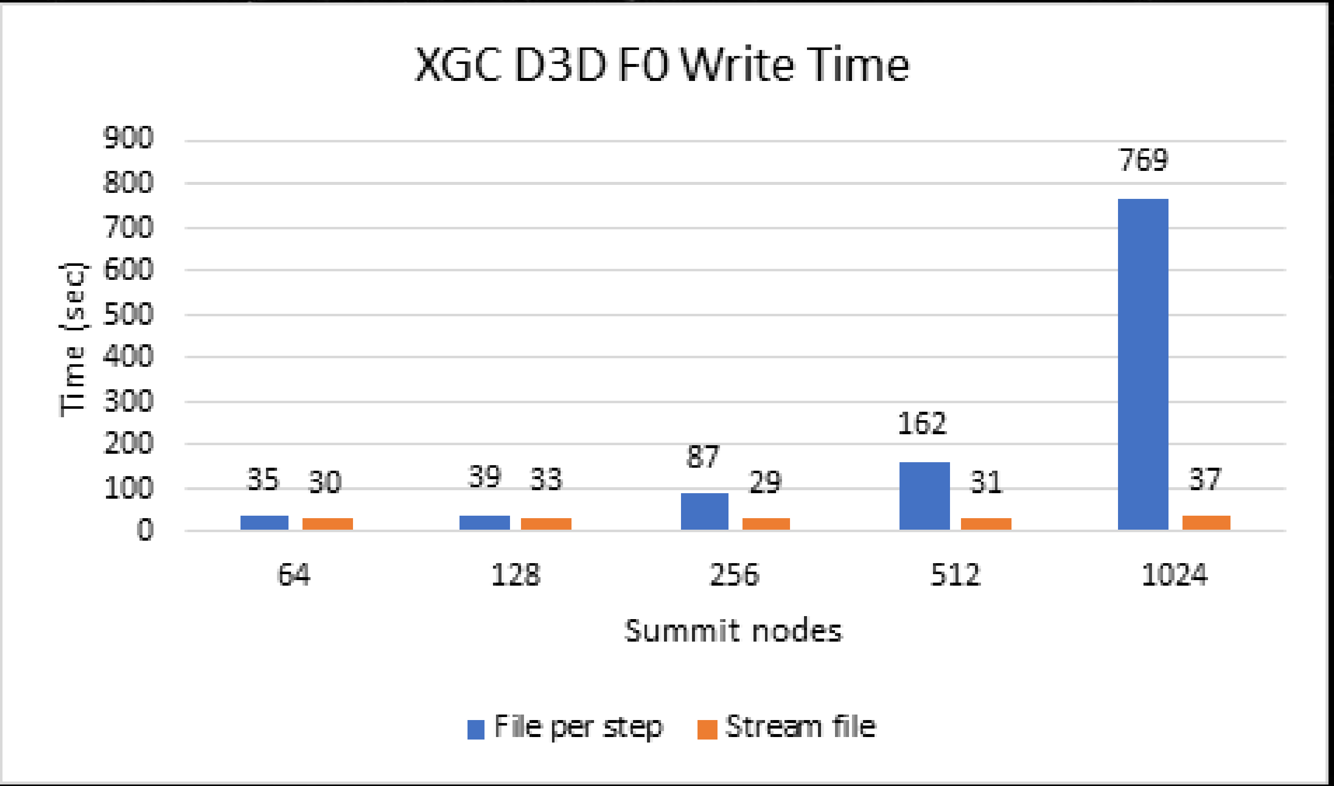


- A fixed aggregation ratio breaks down as we scale up the nodes
- Best options here:
  - 42 nodes – 42*42=1764 subfiles (1:1)
  - 84 nodes – 84*42=3528 subfiles (1:1)
  - 168 nodes – 168*21=3528 subfiles (1:2)
  - 336 nodes – 336*6=2016 subfiles (1:7)
- Summit general guidance
  - One subfile per GPU (6 per node) is a good start but apps usually have one MPI task/GPU, so keep the total number of subfiles below 4000

Application	Nodes/GPUs, 6 tasks/node	Data Size per step	I/O speed	ADIOS NumAggregators
SPECFEM3D_GLOBE	3200/19200	250 TB	~2 TB/sec	3200 (1:6 aggregation ratio)
GTC	512/3072	2.6 TB	~2 TB/sec	3072 (1:1 aggregation ratio)
XGC	512/3072	64 TB	1.2 TB/sec	1024 (1:3 aggregation ratio)
LAMMPS	512/3072	457 GB	1 TB/sec	512 (1:6 aggregation ratio)

ADIOS Scaling for large parallel file systems: number of steps

- Another aspect of number of files: number of output steps
- New output every step -> many files over the course of simulation
  - If the rate of steps stresses the file system, write performance will drop
  - Actually, the total time of creation of files will add up
- Appending multiple output steps into same file is better



XGC 100 output steps in 1245 second simulation, 40 GB per step (4TB total)

One file per node created in each step

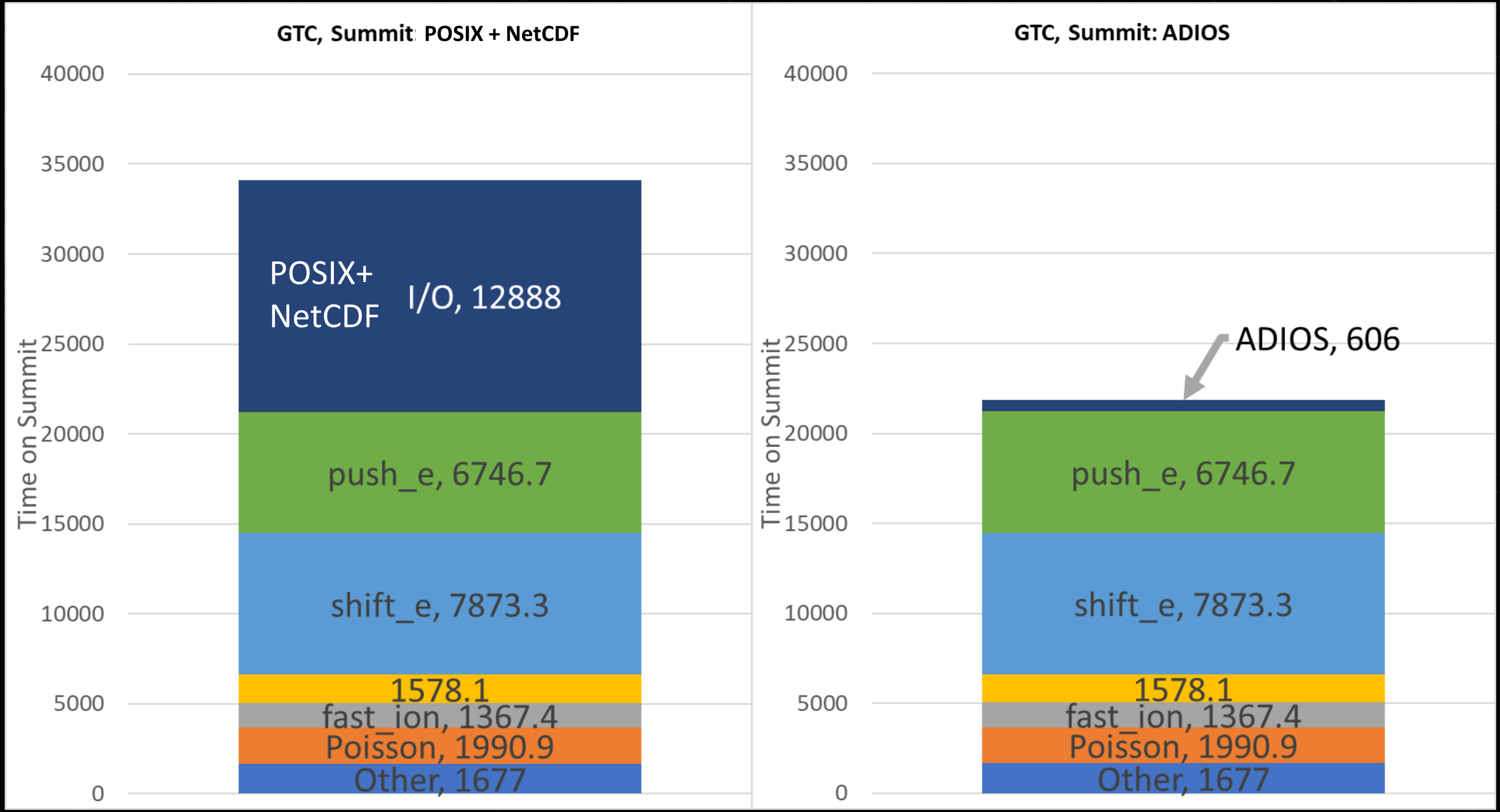
Single output for all steps (one file per node)

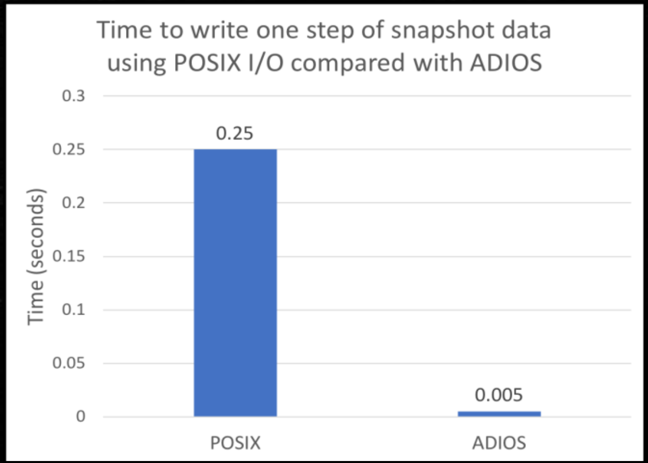
Oak Ridge National Laboratory

Optimized GTC, a fusion PIC code, I/O on Summit

PI: Zhihong Lin, UC Irvine

- Change to ADIOS I/O: Total simulation time reduced from 9.5 hours to 6.1 hours on 1024 nodes on Summit





* average cost when measuring 10 000 steps

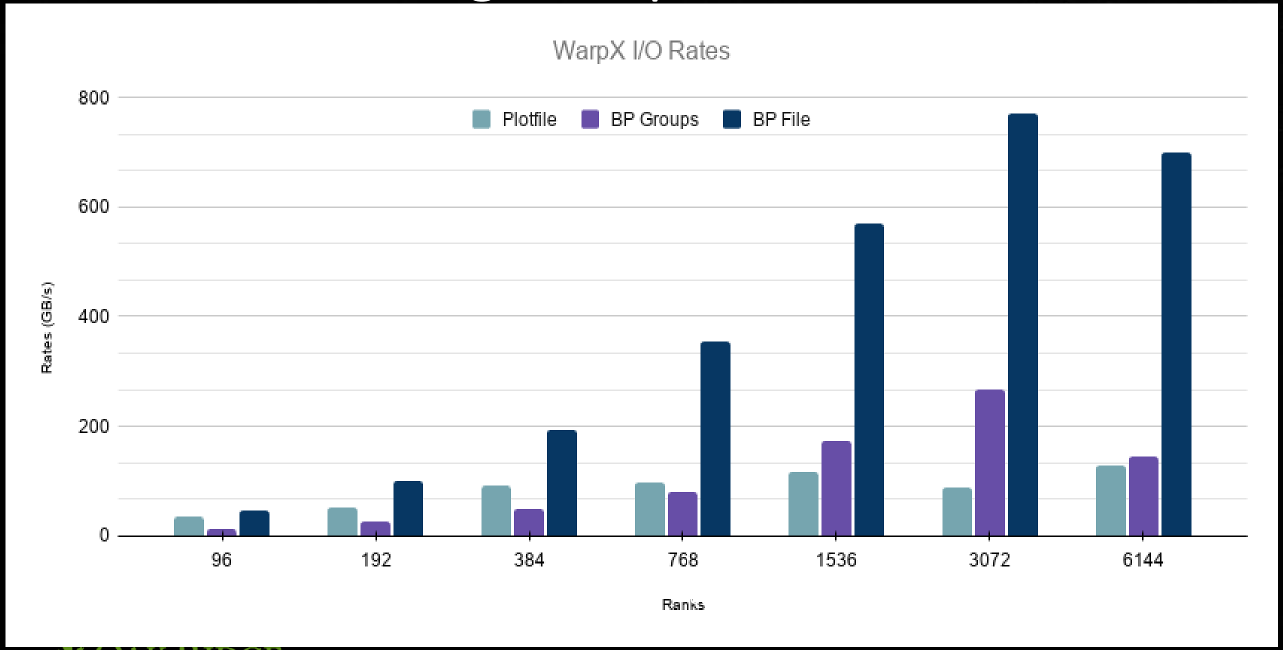
OAK RIDGE National Laboratory

WarpX example: appending + aggregation

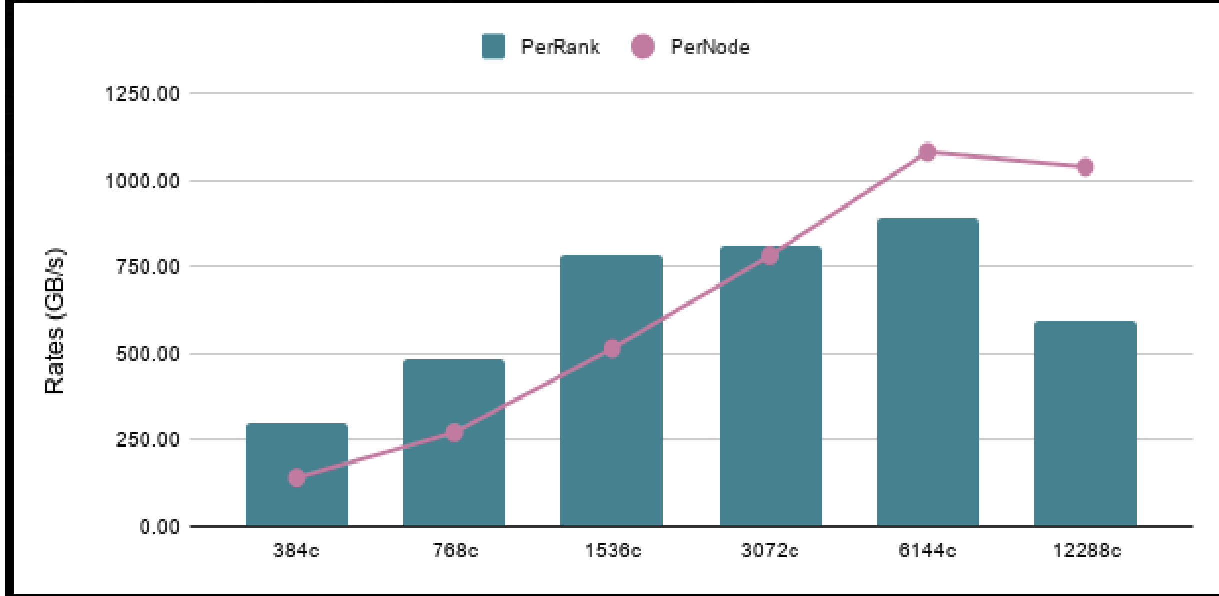
PI: Jean-Luc Vay, LBNL

- One file per compute node (6 processes on Summit)
- Better performance at 512 nodes and over
- One file per rank at smaller jobs

WarpX on Summit, Original vs
ADIOS new output per step vs
ADIOS single output



WarpX on Summit, 6 rank per node setup
240 TB on 1024 nodes (6144 cores)

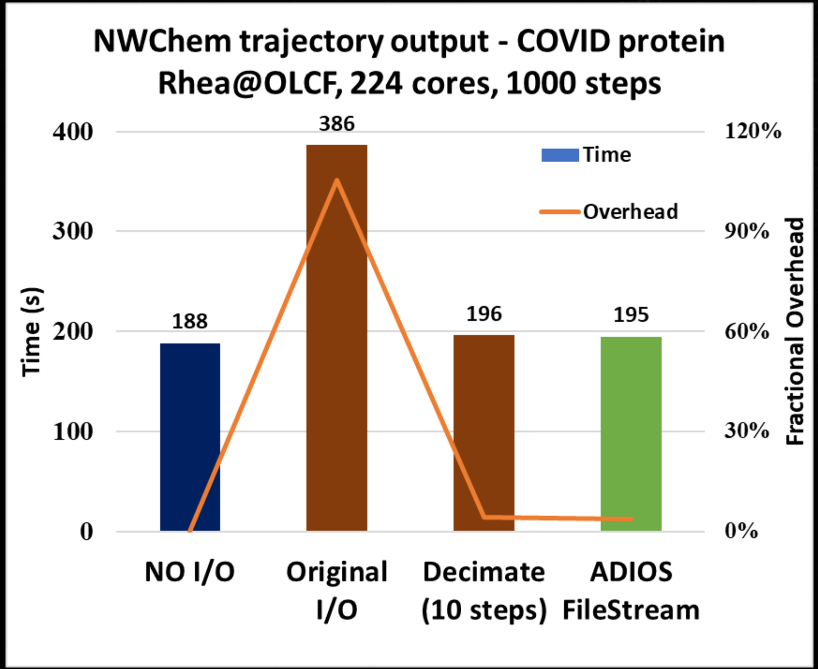


240 TB on 1024 nodes (6144 cores)

OAK RIDGE
National Laboratory

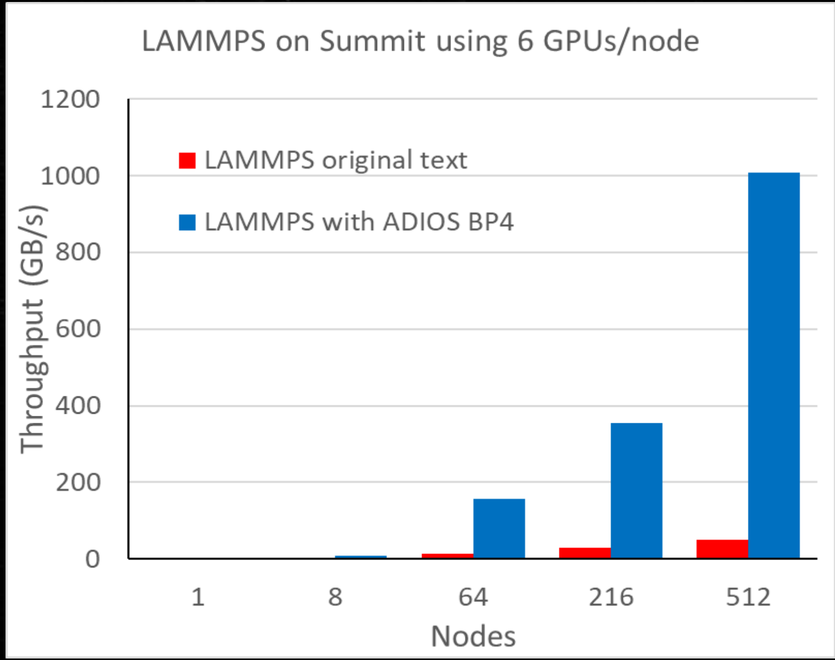
Single process I/O examples: NWChem, LAMMPS

- Moves data between processes as part of preparation for I/O
  - “I am doing POSIX/Fortran I/O on rank 0”
  - While gathering data, no one is writing
- Single file output – not utilizing available bandwidth



ADIOS can write all steps out with little cost (here every 0.2 seconds)

Summit 512 nodes
12B atoms, 5 TB



https://github.com/lammps/lammps/tree/master/src/USER-ADIOS

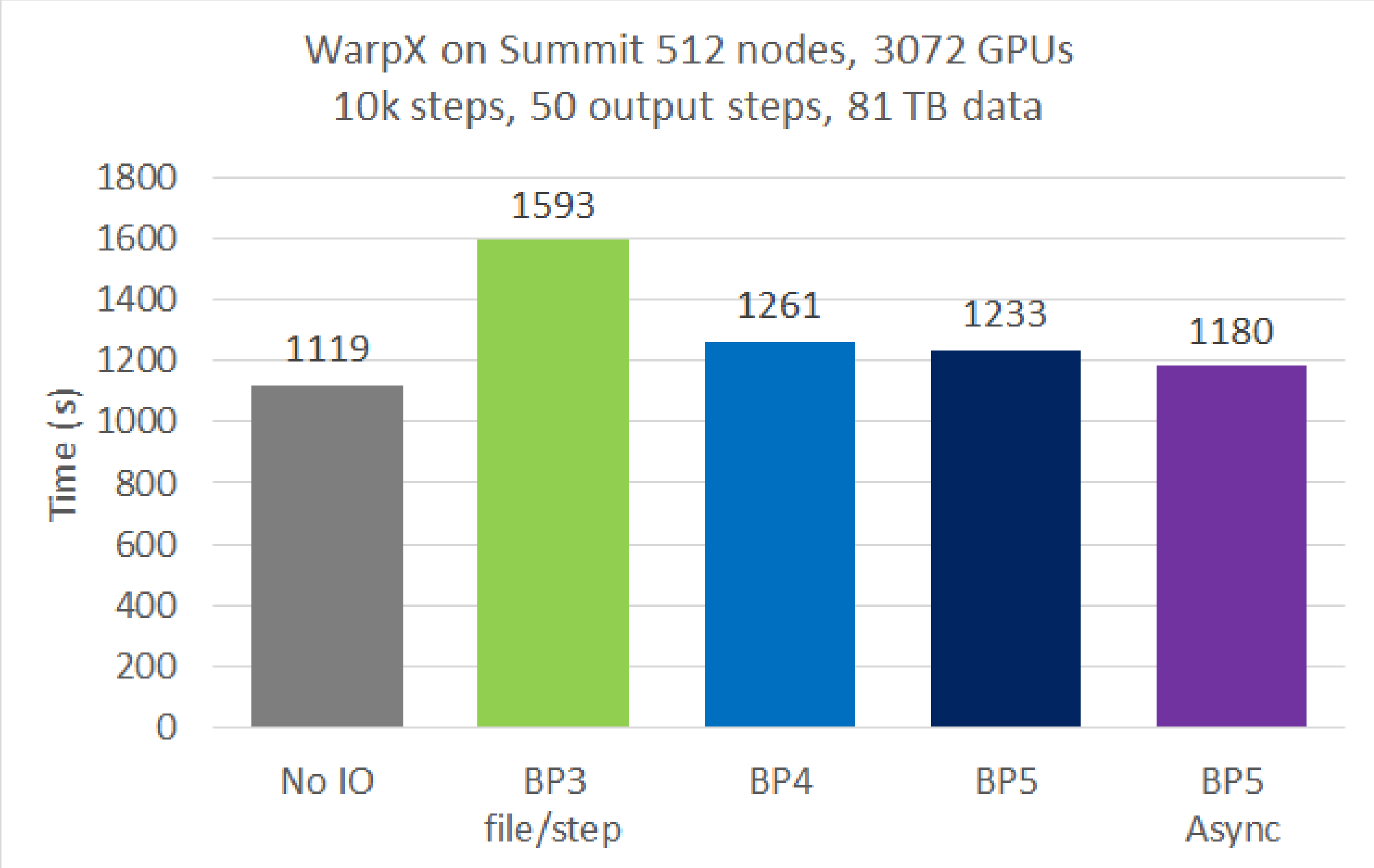
- USER-ADIOS package in LAMMPS for dump commands

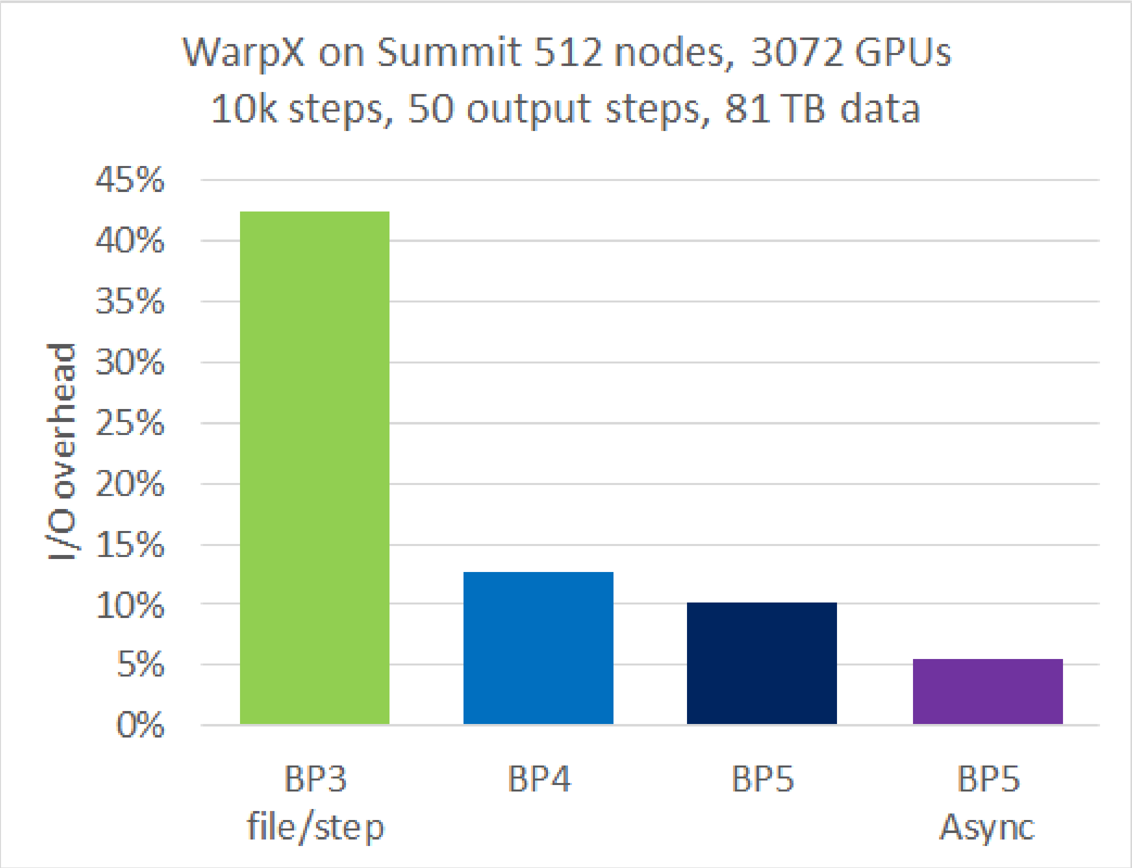
  dump atom/adios   dump custom/adios
- Output goes into an I/O stream

  BP4 file by default   Can use staging engines

Asynchronous write to storage (BP5 engine, 2.8.0 release)

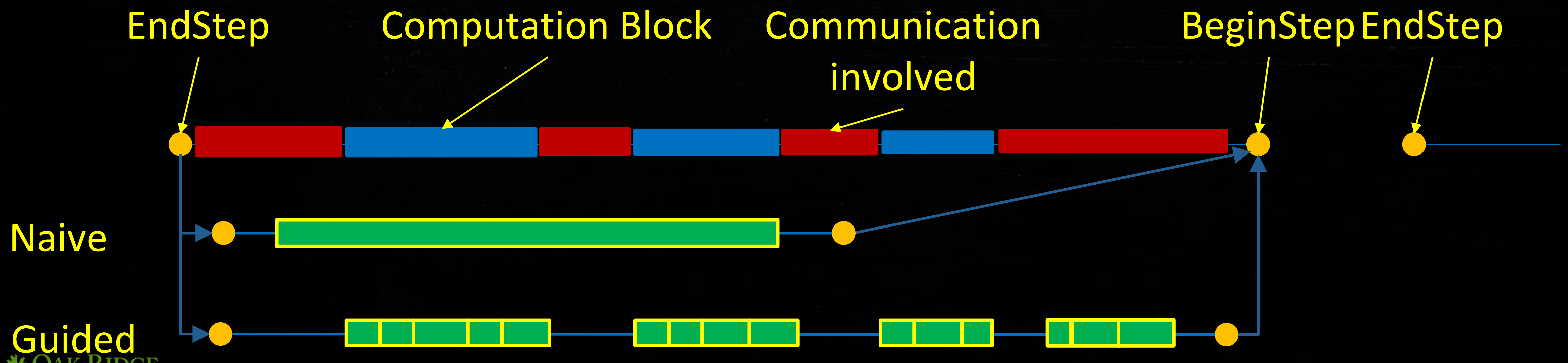
- User friendly On/Off option
  - No need to modify the user code
- Only data writing is async, metadata gathering and writing is still sync





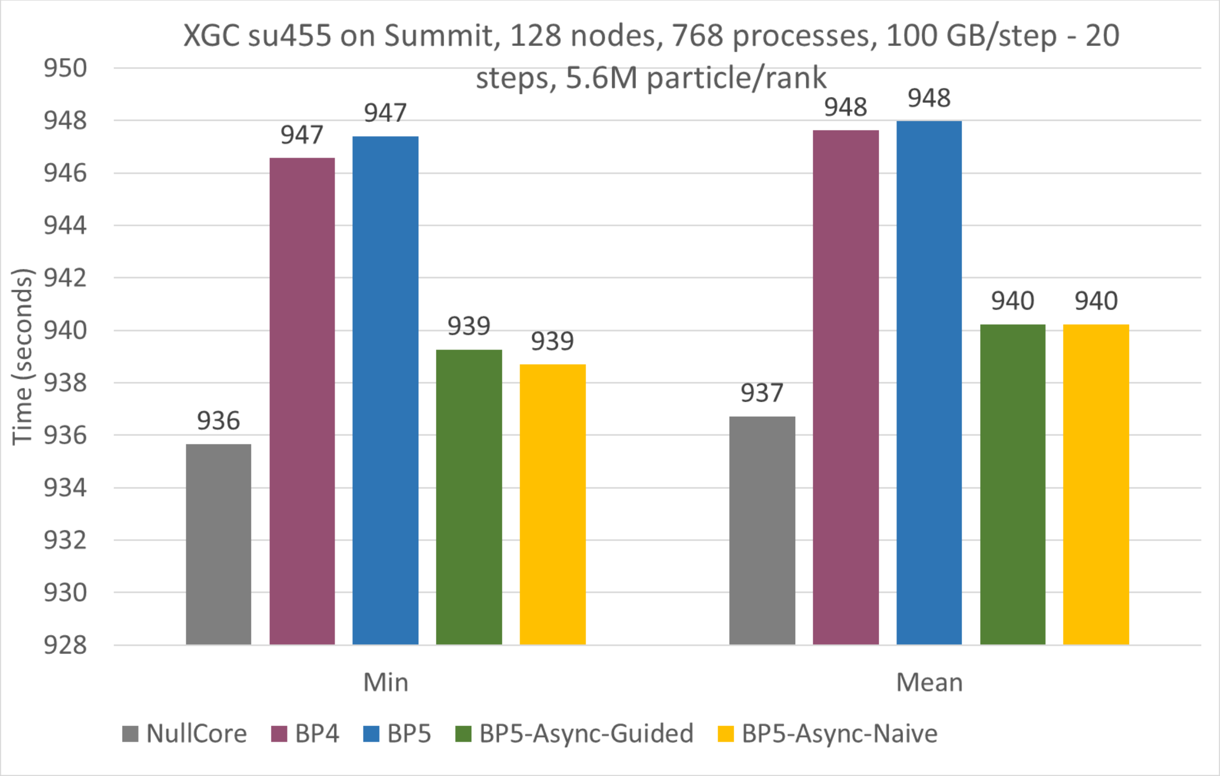
Async strategies

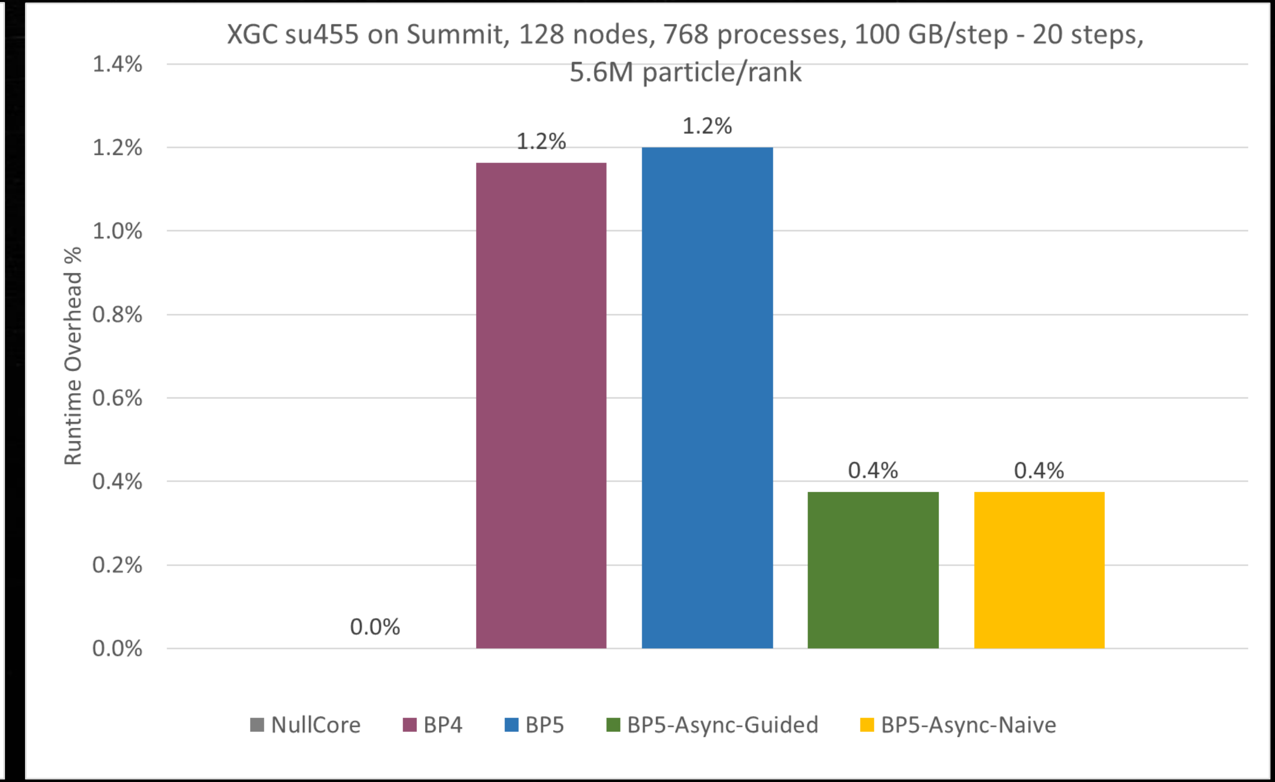
- Naive
  - dump data without thinking
- Guided
  - Application augmented with EnterComputationBlock/ExitComputationBlock pairs
  - Attempt writing during computation blocks
  - Naïve dump when running out of (forecasted) computation blocks



Async IO with XGC only small data

- This is a small D3D run case and quick test

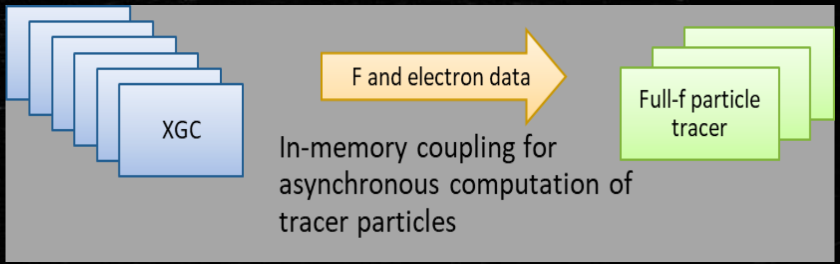




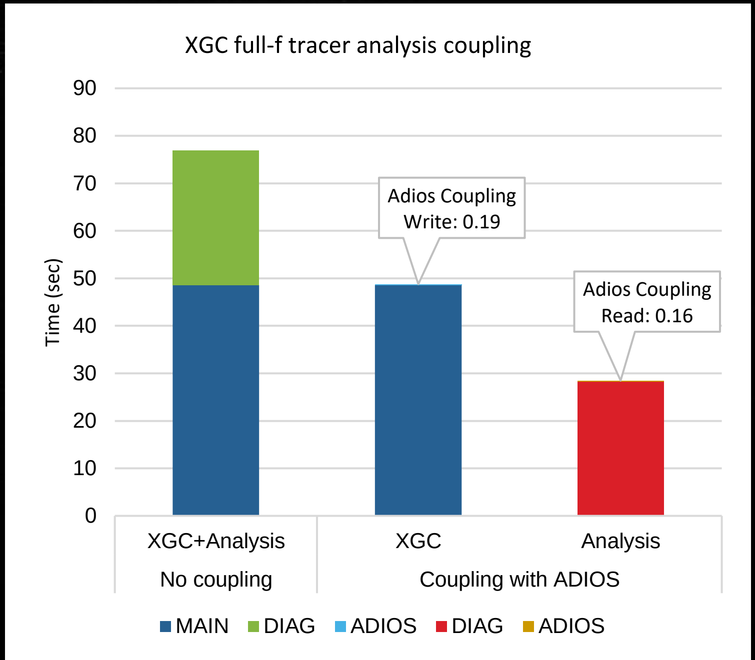
OAK RIDGE
National Laboratory

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

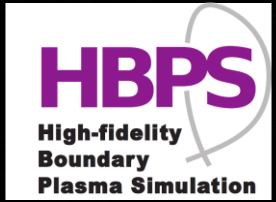
- Tracer particle analysis enables understanding of the transport characteristics spanning the pedestal and scrape-off layer
- It is costly to perform and is communication-heavy
- Asynchronously stage data to the tracer particle analysis running on additional nodes
    - Coupled data size: f0(95 GB) + 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., et al. "Coupling exascale multiphysics applications: Methods and lessons learned". IEEE e-Science, 2018.

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
- 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