RNTUPLE AND DAOS

(BUT REALLY A COLLECTION OF TOPICS…)

ROB ROSS
MATHEMATICS AND COMPUTER SCIENCE DIVISION
ARGONNE NATIONAL LABORATORY
RROSS@MCS.ANL.GOV

WHO AM I?

- I’m a computer scientist with a background in HPC storage, I/O, and communication (MPI)
  https://www.mcs.anl.gov/research/projects/mochi/
  https://www.mcs.anl.gov/research/projects/darshan/
  https://parallel-netcdf.github.io/
- I lead a team of computer scientists and AI/ML experts helping DOE scientists use HPC resources
  http://rapids-scidac.org
- I’m co-leading storage-related activities with Peter VG in HEP-CCE
  https://www.anl.gov/hep-cce

WHY AM I HERE?

- Gain a better understanding of HEP workflows, how HPC might play a role, where computer science challenges lie
- Learn personalities, organizations, and terminologies ☺
- Help align potential HEP-CCE activities with ROOT, RNTuple, and experiment efforts
- Serve as a resource for HPC I/O knowledge/tools

WHAT’S DARSHAN AND WHY SHOULD I CARE?

WHAT'S DARSHAN?

- **Darshan is a scalable HPC I/O characterization tool. It captures a concise picture of application I/O behavior with minimal overhead.**
- Deployed at DOE supercomputing sites (ALCF, OLCF, and NERSC)
- No changes to code or development process

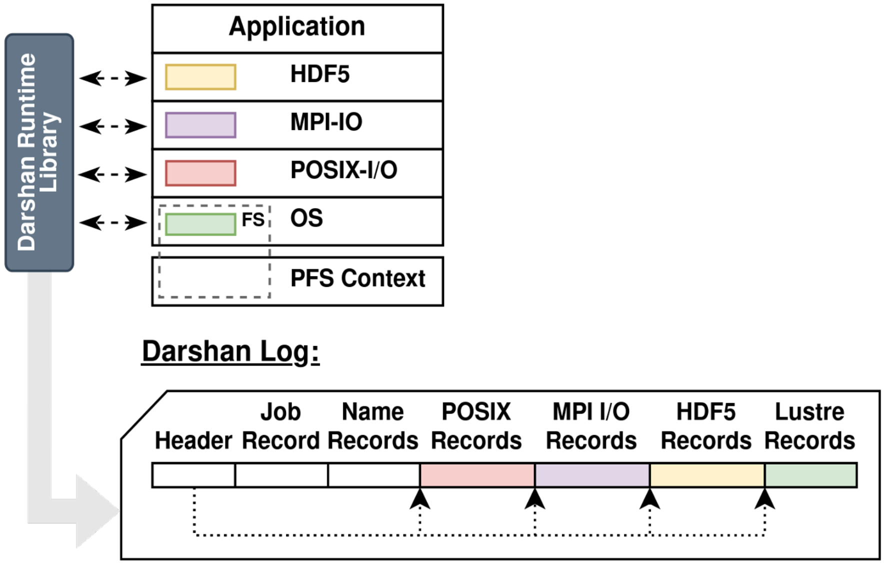


Figure courtesy Jakob Luettgau (UTK)

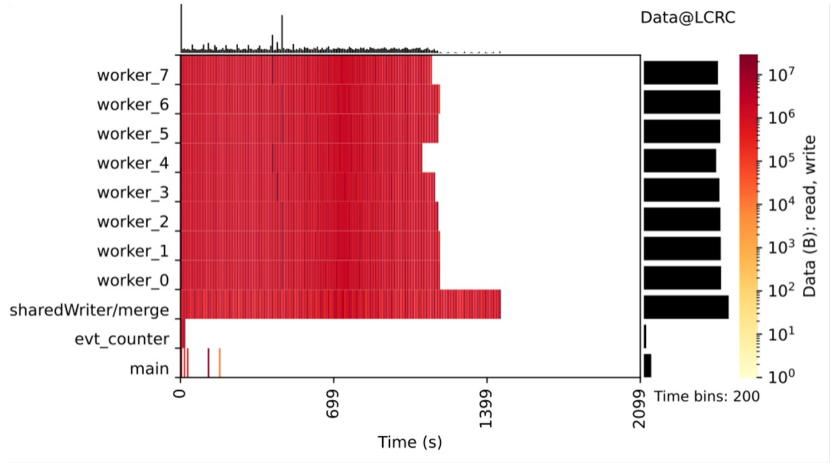
- Negligible performance impact: just "leave it on"
- Includes counters, timers, histograms, etc.
- **We routinely learn things about application I/O behavior using Darshan.**

https://www.mcs.anl.gov/research/projects/darshan/

Argonne NATIONAL LABORATORY

DARSHAN ANALYSIS TOOLS

- Tools and interfaces to inspect and interpret log data
  - PyDarshan command line utilities like the new job summary tool
  - Python APIs for usage in custom tools, Jupyter notebooks, etc.



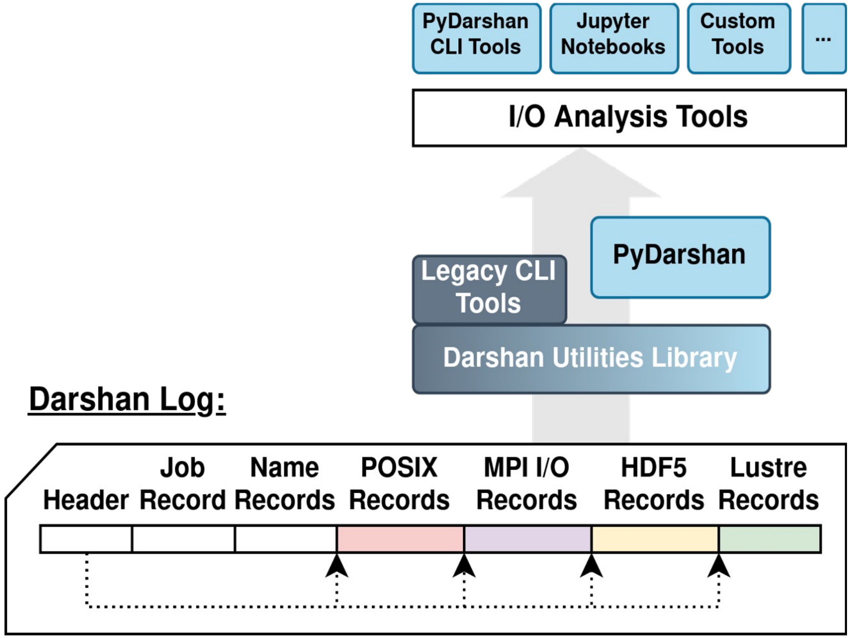
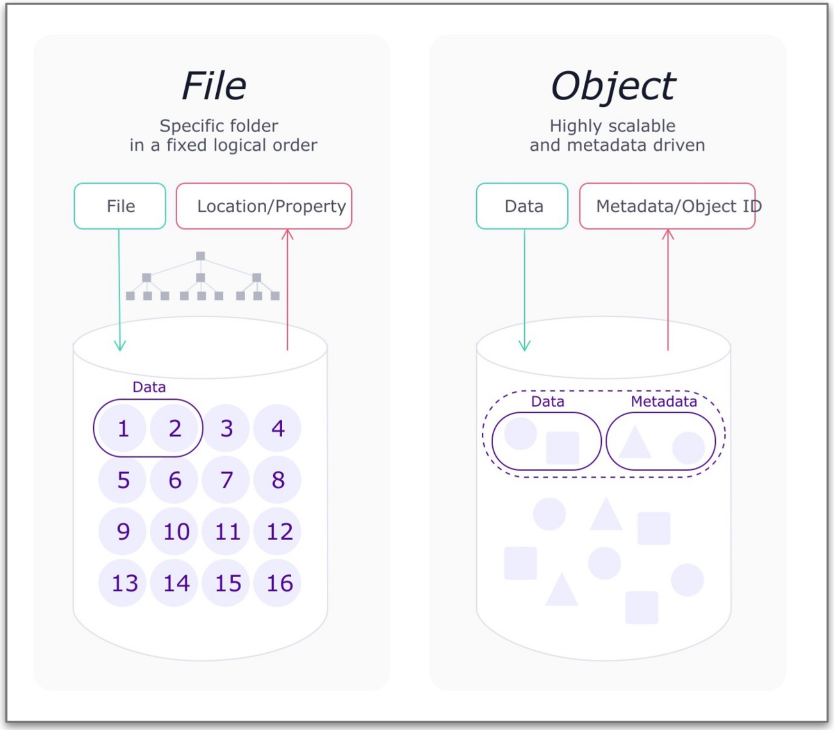


Figure courtesy Jakob Luettgau (UTK)

OBJECTS, DAOS, ROOT (RNTUPLE), AND DARSHAN

OBJECT STORAGE: WHAT IS IT?

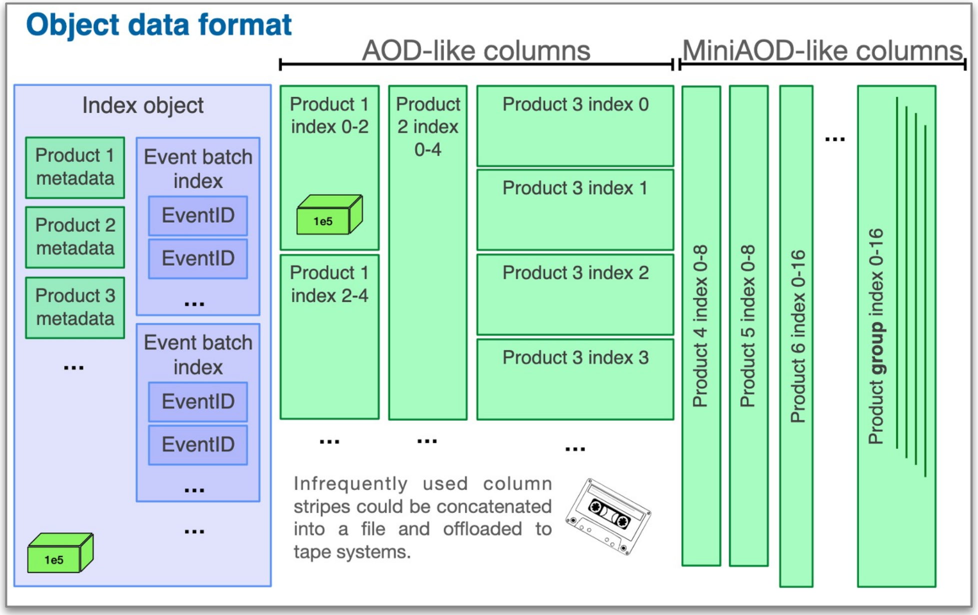
- "Object storage" is a term used to describe a collection of storage technologies that focus on the ability to store, reference, and access large collections of unstructured data
- Unlike a file system, how you find things is generally handled separately (e.g., via DB)
- There are lots of flavors of object storage being used today in different contexts:
  - Cloud storage (e.g., S3): large, immutable objects, HTTP access
  - Distributed filesystems (e.g., RADOS): large, mutable and byte-addressable objects with file system access (i.e., Ceph)
  - HPC storage (e.g., DAOS): semi-structured, mutable, byte-addressable objects with key/value access



https://www.scaleway.com/en/blog/understanding-the-different-types-of-storage/

STORING ROOT DATA IN OBJECTS

- Numerous potential advantages for using in HEP:
  - Allow fine-grained versioning, avoiding replication of unchanged objects
  - Facilitate user-driven data augmentation, to subset of events
  - These methods of referencing save storage space
- Object storage activities on HPC side as well
  - DAOS
  - HDF5 over objects
  - Data lakes for AI applications

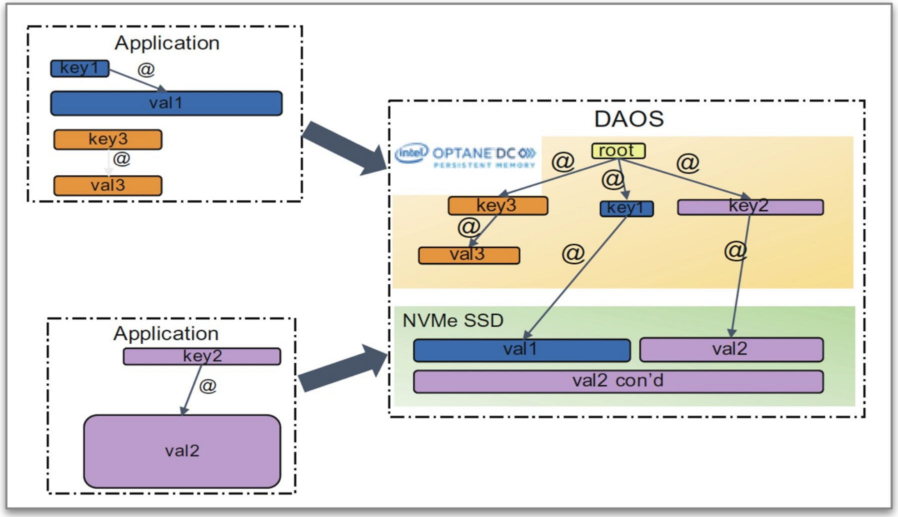


The FNAL team is investigating mapping CMS datasets into Ceph objects. The approach is not specific to Ceph, although different mappings might be more advantageous on specific underlying technologies.

Bo Jayatilaka, Christopher Jones, Nicholas Smith, "Using CEPH Object store with ROOT serialization in CMS", December 2022. https://indico.fnal.gov/event/57189/contributions/254706/attachments/162368/214598/ncsmith-uscms-objectstoresQ4.pdf

DISTRIBUTED ASYNCHRONOUS OBJECT STORAGE (DAOS)

- DAOS is an object storage service developed for use on persistent memory technologies as a very high performance online storage layer
- Data model includes both key:value objects and array objects
- Array objects can be used to streamline storage of large multidimensional arrays with record addressability
- Access can be via POSIX or directly via custom API
  - Custom API, array objects, striping all provide opportunities for optimization beyond a "standard" object store

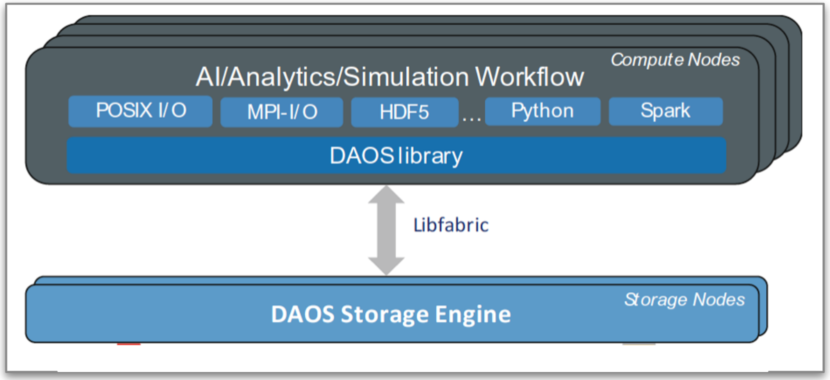


Example of keys and references employed in a DAOS volume. Array objects preserve record addressability that is incredibly valuable in many HPC contexts (e.g., HDF5 arrays).

Zhen Liang, Johann Lombardi, Mohamad Chaarawi, Michael Hennecke,"DAOS: A Scale-Out High Performance Storage Stack for Storage Class Memory," June 2020.
https://doi.org/10.1007/978-3-030-48842-0_3

OBJECT STORES, DAOS, AND RNTUPLE

- HEP-CCE will study RNTuple DAOS implementation using Darshan
- Darshan already provides initial support for characterizing DAOS storage access
- IOS has successfully used Darshan for current HEP workflows using ROOT
- Aligns with, and will benefit from, other activities to understand and tune DAOS use by team members
- **We will look at how to incorporate TTPerfStats and RNTuple performance data into analysis!**



A variety of user APIs have already been developed for using DAOS from applications, including POSIX, HDF5, and ROOT.

Zhen Liang, Johann Lombardi, Mohamad Chaarawi, Michael Hennecke,"DAOS: A Scale-Out High Performance Storage Stack for Storage Class Memory," June 2020. https://doi.org/10.1007/978-3-030-48842-0_3

BONUS: A BIT ABOUT HPC I/O

Aurora

Argonne's upcoming exascale supercomputer will leverage several technological innovations to support machine learning and data science workloads alongside traditional modeling and simulation runs.

PEAK PERFORMANCE
≥2 Exaflop DP

Intel® Xe ARCHITECTURE-BASED GPU
Data Center GPU Max Series

INTEL® XEON® SCALABLE PROCESSOR
Intel Xeon CPU Max Series

PLATFORM
HPE Cray EX

Argonne NATIONAL LABORATORY
U.S. DEPARTMENT OF ENERGY
intel
Hewlett Packard Enterprise
Aurora

Compute Node
2 Intel® Xeon® CPU Max Series processors; 6 Intel® Data Center GPU Max Series GPUs; Unified Memory Architecture; 8 fabric endpoints; RAMBO

GPU Architecture
Intel® Data Center GPU Max Series; Tile-based chiplets, HBM stack, Foveros 3D integration, 7nm

CPU-GPU Interconnect
CPU-GPU: PCIe
GPU-GPU: Xe Link

System Interconnect
HPE Slingshot; Dragonfly topology with adaptive routing

Network Switch
25.6 Tb/s per switch, from 64–200 Gbs ports (25 GB/s per direction)

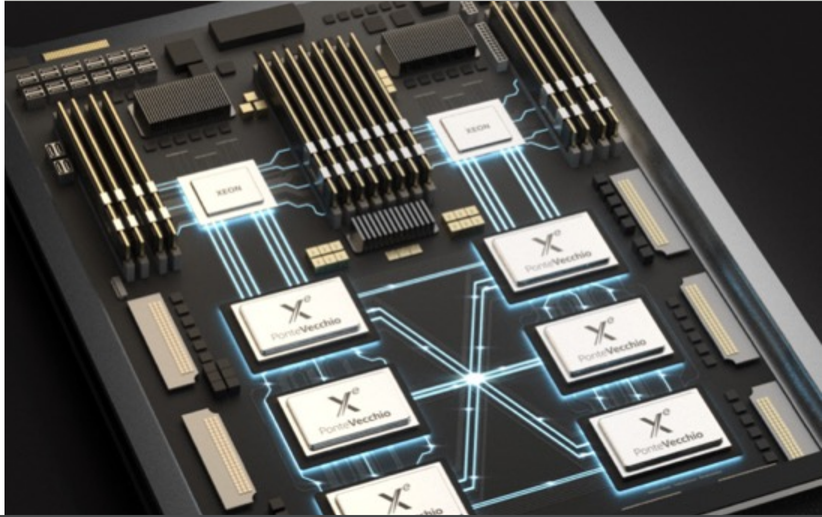
High-Performance Storage
≥230 PB, ≥25 TB/s (DAOS)

Programming Models
Intel oneAPI, MPI, OpenMP, C/C++, Fortran, SYCL/DPC++

Node Performance
>130 TF

System Size
>10,000 nodes

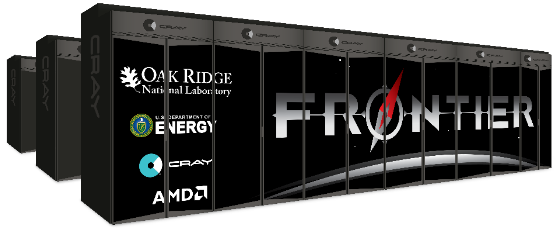
"Up to 56 cores"



From K. Riley, SciDAC PI Meeting, Sept. 2023

Frontier Overview

Extraordinary Engineering



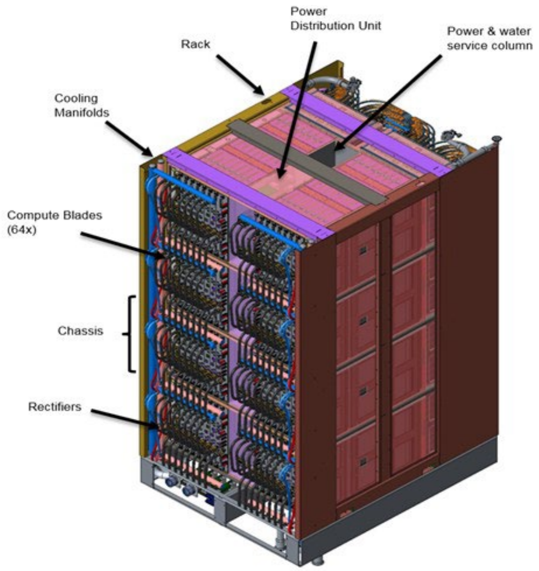
System

- 2 EF Peak DP FLOPS
- 74 compute racks
- 29 MW Peak Power
  (5 MW idle/ 18 MW average)
- 9,472 nodes
- 9.2 PB memory
  (4.6 PB HBM, 4.6 PB DDR4)
- Cray Slingshot network with dragonfly topology
- 37 PB Node Local Storage
- 716 PB File storage
- 4,000 ft^2 foot print

Built by HPE

Olympus rack

- 128 AMD nodes
- 8,000 lbs
- Supports 400 KW



Powered by AMD

AMD node **"64 cores"**

- 1 AMD "Trento" CPU
- 4 AMD MI250X GPUs
- 512 GiB DDR4 memory on CPU
- 512 GiB HBM2e total per node
  (128 GiB HBM per GPU)
- Coherent memory across the node
- 4 TB NVM
- Infinity Fabric fully connects GPUs & CPU
- 4 Cassini NICs connected to the 4 GPUs

Compute blade

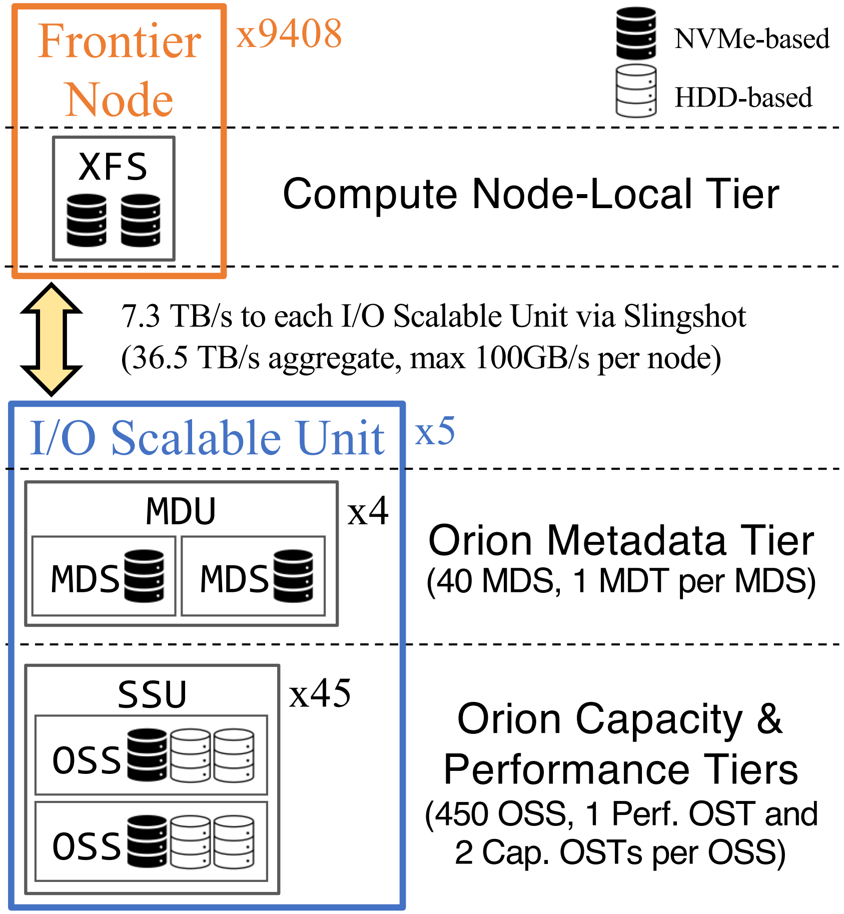
- 2 AMD nodes

All water cooled, even DIMMS and NICs

From A. Geist, SciDAC PI Meeting, Sept. 2023

Orion Tiered Architecture

- Capacity Tier:
  - 679 PB
  - RD/WR: 5.5/4.6 TB/s
  - 47,700 18 TB HDD
- Performance Tier:
  - 11.5 PB
  - RD/WR: 10 TB/s
  - 5,400 3.2 TB NVMe
- Metadata Tier:
  - 10 PB
  - RD/WR: 0.8/0.4 TB/s
  - 480 30.7 TB NVMe

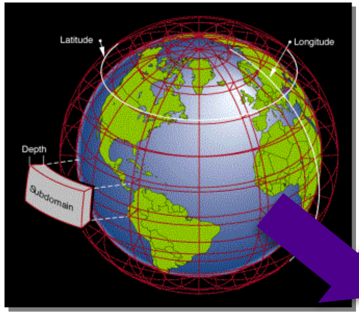

Frontier Node
x9408
NVMe-based
HDD-based
XFS
Compute Node-Local Tier
7.3 TB/s to each I/O Scalable Unit via Slingshot
(36.5 TB/s aggregate, max 100GB/s per node)
I/O Scalable Unit
x5
MDU
x4
MDS
MDS
Orion Metadata Tier
(40 MDS, 1 MDT per MDS)
SSU
x45
OSS
OSS
Orion Capacity & Performance Tiers
(450 OSS, 1 Perf. OST and 2 Cap. OSTs per OSS)

From M. Brim, ATPESC, Aug. 2023

Pre-Production Application I/O Performance on Orion

- Default progressive file layout is good for file-per-process
  - WarpX (ADIOS) - File-per-process @ 4,096 nodes (32k processes, 1.5 GiB/process) achieved ~7.9 TiB/sec for simulation output writing
  - GTC (ADIOS) - File-per-process @ 2,048 nodes (16k processes, 2 GiB/process) achieved ~5.2 TiB/sec for checkpointing three datasets
- But not so great for large single-shared-file
  - Flash-X (HDF5) - Shared-file @ 512 nodes (28k processes, 29 GiB/node) got only 6 GiB/sec
    - Using Capacity tier only with wide-striping improved this by 20x

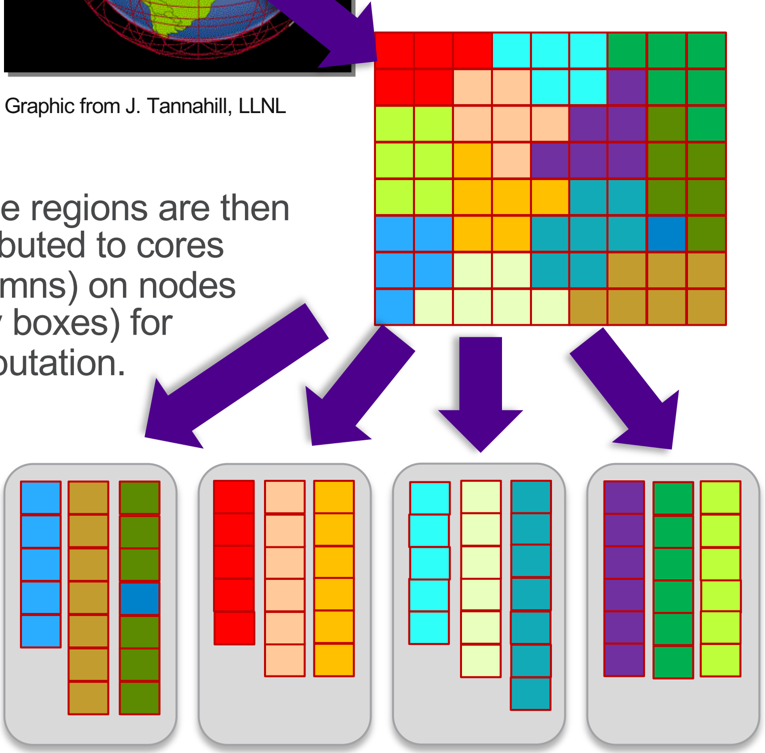
From M. Brim, ATPESC, Aug. 2023

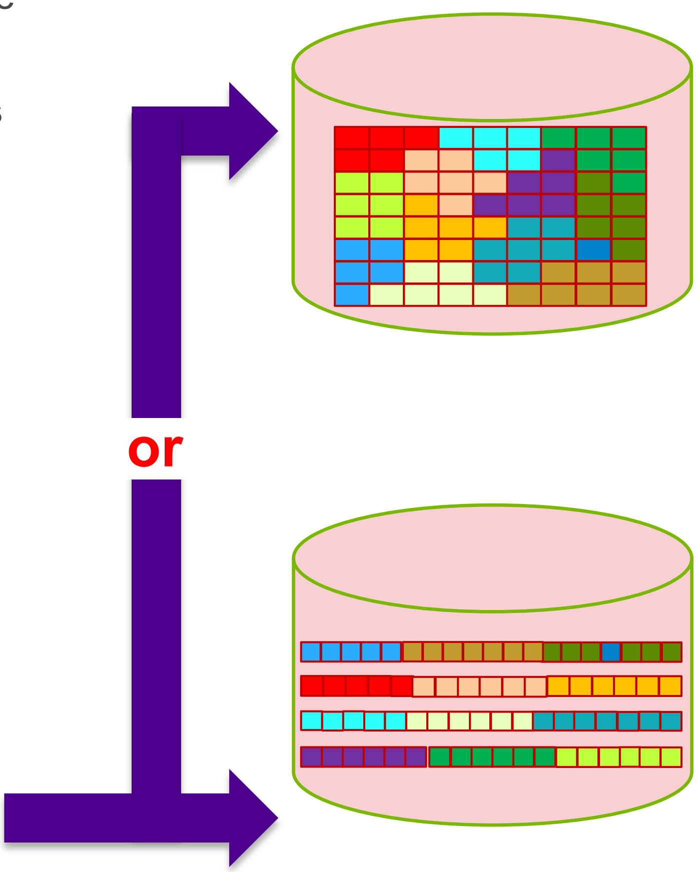


Graphic from J. Tannahill, LLNL

Typical simulations divide up the region being simulated into chunks, then group those chunks into similar amounts of work.

These regions are then distributed to cores (columns) on nodes (grey boxes) for computation.





To prepare data for analysis, a code can write in a ***canonical*** view by processing the data while it is in memory, resulting in a better organized dataset.

E.g., Traditional use of netCDF.

When speed of writing is the priority, ***blobs*** of data are written from each node into individual files that must usually be post-processed for analysis.

E.g., HDF5 Log VOL and ADIOS.

COMPARING I/O STRATEGIES

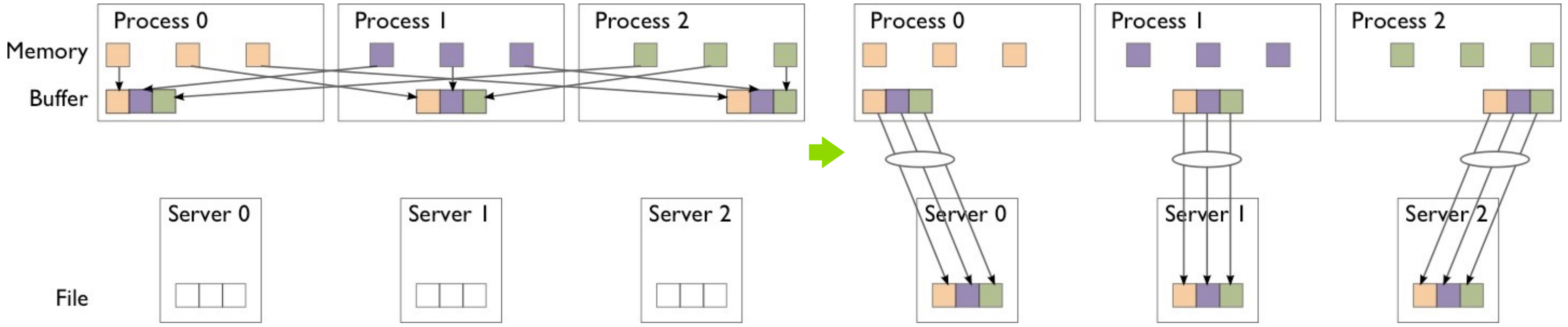
- **For traditional "canonical" output into a netCDF format, PnetCDF is substantially faster than NetCDF4**
- "Blob" formats are faster for writing, especially for large numbers of time steps
  - **Directly writing into HDF5 with DataLib HDF5 Log VOL retains advantages of HDF5 ecosystem while achieving best in class performance**
- Who could use our HDF5 Log VOL?
  - Any HDF5 user could switch to our plug-in with no code changes.
  - Accelerated HDF5 opens up HDF5 use to teams who have found performance inadequate in the past.
  - *Note that the HDF5 Log VOL cannot solve the NetCDF4 API problems!*

	Cori @ NERSC	NetCDF4 (on HDF5)	PnetCDF	HDF5 Log VOL	ADIOS
		Canonical		Blob	
Case F	# MPI processes	21600 on 258 nodes			
	# sub-files	1		258	
	Write amount in GiB	21.3			
	# write flushes		25	1	1
	End-to-end time (sec)	**> 3600 (!)**	**11.64**	**5.01**	**4.62**
Case G	# MPI processes	9600 on 115 nodes			
	# sub-files	1		115	
	Write amount in GiB	80.0			
	# write flushes		1	1	1
	End-to-end time (sec)		**20.04**	**1.55**	**3.16**
Case I	# MPI processes	1344 on 16 nodes			
	# sub-files	1	1	16	32
	Write amount in GiB	86.1			
	# write flushes		240	1	1
	End-to-end time (sec)		**217.06**	**24.53**	**37.40**

- Case I: 1/2 degree, 25 KM global res., 240 time steps

AVOIDING LOCK CONTENTION

- **To avoid lock contention when writing to a shared file, we can reorganize data between processes**
- *Two-phase I/O* splits I/O into a data reorganization phase and an interaction with the storage system (two-phase write depicted):
  - Data exchanged between processes to match file layout

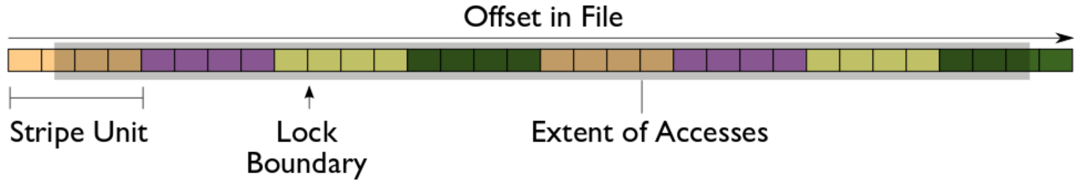


Phase 1: Data are exchanged between processes based on organization of data in file.

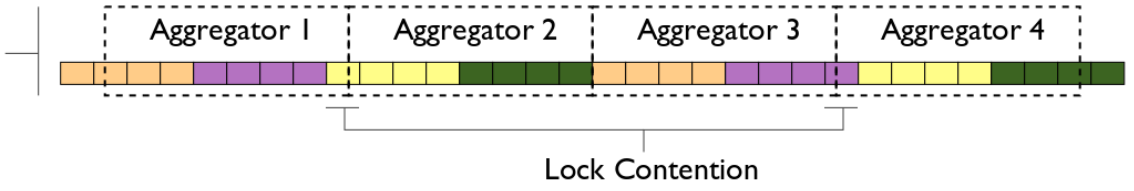
Phase 2: Data are written to file (storage servers) with large writes, no contention.

TWO-PHASE I/O ALGORITHMS

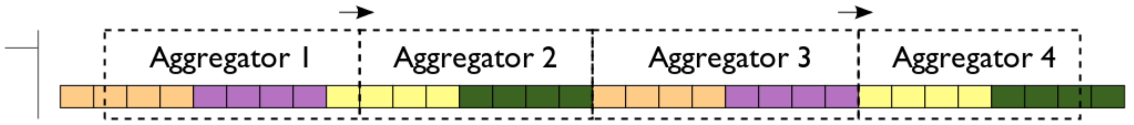
Imagine a collective I/O access using four aggregators to a file striped over four file servers (indicated by colors):



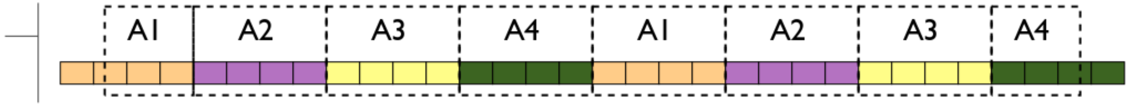
One approach is to evenly divide the region accessed across aggregators.



Aligning regions with lock boundaries eliminates lock contention.



Mapping aggregators to servers reduces the number of concurrent operations on a single server and can be helpful when locks are handed out on a per-server basis (e.g., Lustre).



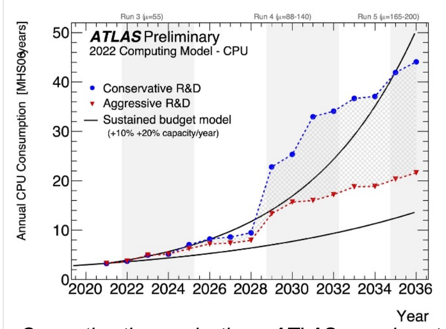
For more information, see W.K. Liao and A. Choudhary, “Dynamically Adapting File Domain Partitioning Methods for Collective I/O Based on Underlying Parallel File System Locking Protocols,” SC2008, November 2008.

Improving I/O in the Sherpa and Pythia event generators (HEP)

With the HEP-ASCR Partnership (program) – *if applicable*

Scientific Achievement

Experiments at the Large Hadron Collider generate complex final states that must be simulated at high precision to identify the physics principles which determine how subatomic particles interact. Current simulations are too complex to achieve the precision goals set for the 2030s. We improved performance and scalability.



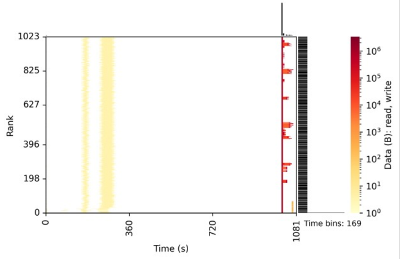
Computing time projections, ATLAS experiment

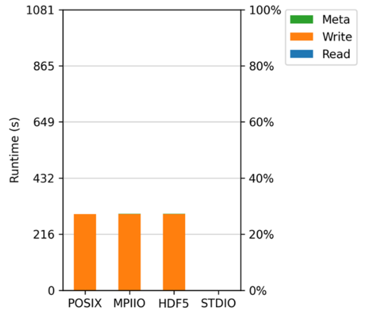
Significance and Impact

An I/O bottleneck limited the number of particle collisions we could simulate. It also broke strong scaling in parallel computations. By dramatically reducing the I/O overhead, we can now carry out the Monte-Carlo integration with a greater number of events. This reduces the statistical uncertainty of the simulations, and leads to theoretical predictions with higher confidence level.

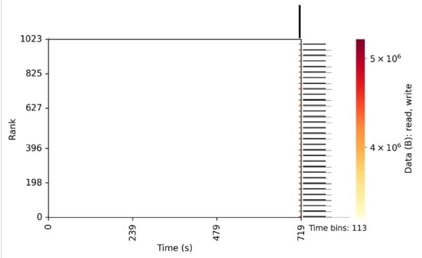
Technical Approach

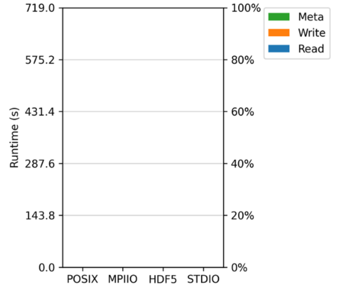
- Modified support libraries to expose collective HDF5 operations
- Adjusted use of HDF5 to allow further collective I/O optimizations
- Off-line converter to work around one HDF5 performance bug
- No machine-specific tuning required: can use this approach on any parallel file system





Initial I/O approach





After optimization: I/O time reduced to barely measurable; overall runtime reduced by 34%

I/O cost as the simulation scaled up alarmed scientists and prevented running higher resolution experiments. By enabling HDF5 collective data and metadata optimizations, and by making a minor adjustment to the arrangement of variables in the HDF5 file, the simulation could create HDF5 files significantly faster and simulate more events.

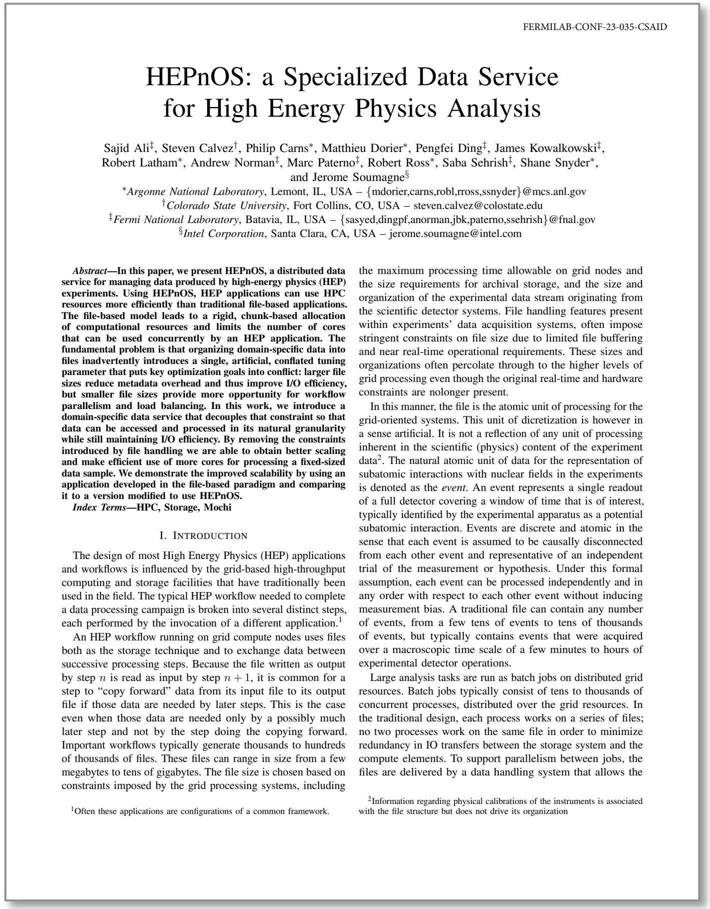
U.S. DEPARTMENT OF ENERGY | Office of Science

Argonne NATIONAL LABORATORY   Fermilab

BONUS #2: AN EXPERIMENT WITH DATA SERVICES IN HPC

ACCELERATING ANALYSIS ON AN HPC PLATFORM

- **What if we custom-built a data service for an experimental data analysis workflow?**
  - NOvA in this case
- Bulk ingest the entire dataset (to be analyzed) upfront into a scalable service
  - 17,878,347 candidate interactions (slices)
  - 1.1% of 2018 analysis size
  - Loaded four copies for experiment shown
- Run analysis across many compute nodes
  - 8:1 ratio of computing to service nodes
- Allow intermediate data products to be dropped into this service also



FERMILAB-CONF-23-035-CSAID

HEPnOS: a Specialized Data Service for High Energy Physics Analysis

Sajid Ali‡, Steven Calvez†, Philip Carns*, Matthieu Dorier*, Pengfei Ding‡, James Kowalkowski‡, Robert Latham*, Andrew Norman‡, Marc Paterno‡, Robert Ross*, Saba Sehrish‡, Shane Snyder*, and Jerome Soumagne§
*Argonne National Laboratory, Lemont, IL, USA – {mdorier,carns,robl,rross,ssnyder}@mcs.anl.gov
†Colorado State University, Fort Collins, CO, USA – steven.calvez@colostate.edu
‡Fermi National Laboratory, Batavia, IL, USA – {sasyed,dingpf,anorman,jbk,paterno,ssehrish}@fnal.gov
§Intel Corporation, Santa Clara, CA, USA – jerome.soumagne@intel.com

Abstract*—In this paper, we present HEPnOS, a distributed data service for managing data produced by high-energy physics (HEP) experiments. Using HEPnOS, HEP applications can use HPC resources more efficiently than traditional file-based applications. The file-based model leads to a rigid, chunk-based allocation of computational resources and limits the number of cores that can be used concurrently by an HEP application. The fundamental problem is that organizing domain-specific data into files inadvertently introduces a single, artificial, conflated tuning parameter that puts key optimization goals into conflict: larger file sizes reduce metadata overhead and thus improve I/O efficiency, but smaller file sizes provide more opportunity for workflow parallelism and load balancing. In this work, we introduce a domain-specific data service that decouples that constraint so that data can be accessed and processed in its natural granularity while still maintaining I/O efficiency. By removing the constraints introduced by file handling we are able to obtain better scaling and make efficient use of more cores for processing a fixed-sized data sample. We demonstrate the improved scalability by using an application developed in the file-based paradigm and comparing it to a version modified to use HEPnOS.

***Index Terms*—HPC, Storage, Mochi**

I. Introduction

The design of most High Energy Physics (HEP) applications and workflows is influenced by the grid-based high-throughput computing and storage facilities that have traditionally been used in the field. The typical HEP workflow needed to complete a data processing campaign is broken into several distinct steps, each performed by the invocation of a different application.[1]

An HEP workflow running on grid compute nodes uses files both as the storage technique and to exchange data between successive processing steps. Because the file written as output by step n is read as input by step $n+1$, it is common for a step to "copy forward" data from its input file to its output file if those data are needed by later steps. This is the case even when those data are needed only by a possibly much later step and not by the step doing the copying forward. Important workflows typically generate thousands to hundreds of thousands of files. These files can range in size from a few megabytes to tens of gigabytes. The file size is chosen based on constraints imposed by the grid processing systems, including the maximum processing time allowable on grid nodes and the size requirements for archival storage, and the size and organization of the experimental data stream originating from the scientific detector systems. File handling features present within experiments' data acquisition systems, often impose stringent constraints on file size due to limited file buffering and near real-time operational requirements. These sizes and organizations often percolate through to the higher levels of grid processing even though the original real-time and hardware constraints are nolonger present.

In this manner, the file is the atomic unit of processing for the grid-oriented systems. This unit of dicretization is however in a sense artificial. It is not a reflection of any unit of processing inherent in the scientific (physics) content of the experiment data[2]. The natural atomic unit of data for the representation of subatomic interactions with nuclear fields in the experiments is denoted as the *event*. An event represents a single readout of a full detector covering a window of time that is of interest, typically identified by the experimental apparatus as a potential subatomic interaction. Events are discrete and atomic in the sense that each event is assumed to be causally disconnected from each other event and representative of an independent trial of the measurement or hypothesis. Under this formal assumption, each event can be processed independently and in any order with respect to each other event without inducing measurement bias. A traditional file can contain any number of events, from a few tens of events to tens of thousands of events, but typically contains events that were acquired over a macroscopic time scale of a few minutes to hours of experimental detector operations.

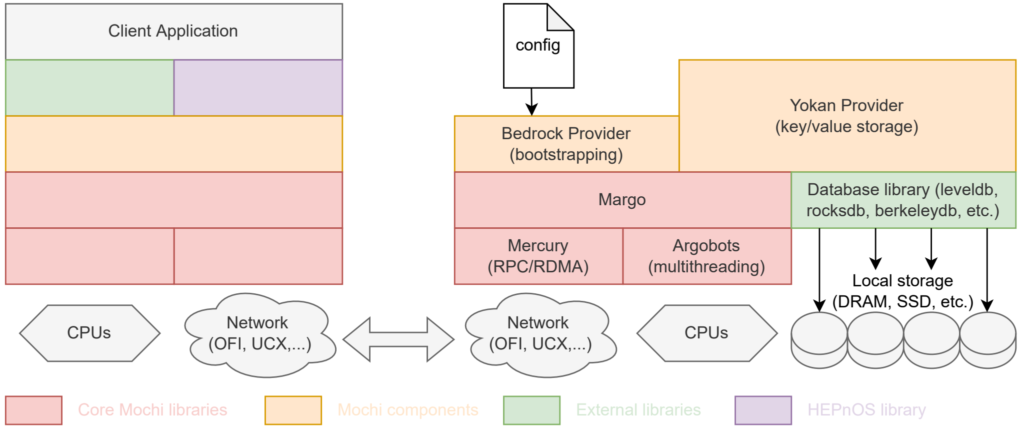
Large analysis tasks are run as batch jobs on distributed grid resources. Batch jobs typically consist of tens to thousands of concurrent processes, distributed over the grid resources. In the traditional design, each process works on a series of files; no two processes work on the same file in order to minimize redundancy in IO transfers between the storage system and the compute elements. To support parallelism between jobs, the files are delivered by a data handling system that allows the

[1]Often these applications are configurations of a common framework.

[2]Information regarding physical calibrations of the instruments is associated with the file structure but does not drive its organization

https://lss.fnal.gov/archive/2023/conf/fermilab-conf-23-035-csaid.pdf

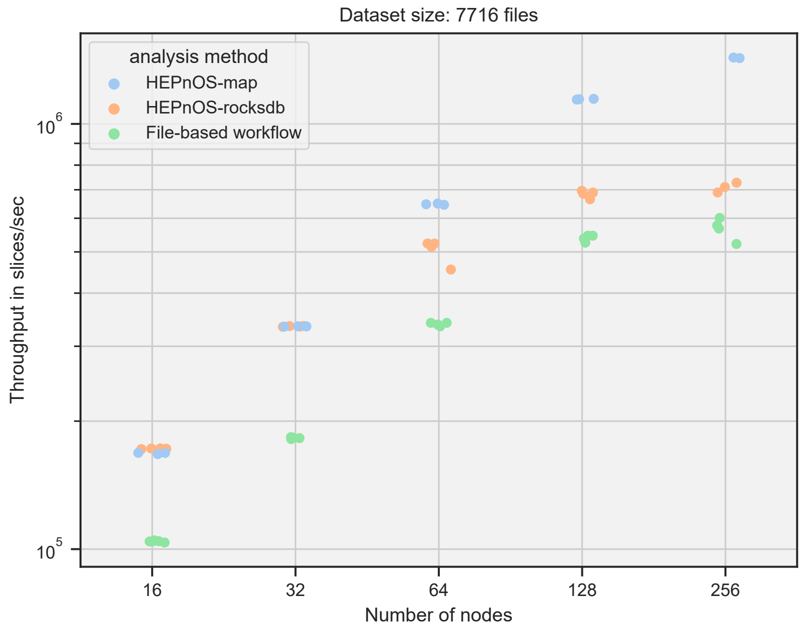
HEPNOS


Client Application
config
Yokan Provider (key/value storage)
Bedrock Provider (bootstrapping)
Margo
Database library (leveldb, rocksdb, berkeleydb, etc.)
Mercury (RPC/RDMA)
Argobots (multithreading)
Local storage (DRAM, SSD, etc.)
CPUs
Network (OFI, UCX,...)
Network (OFI, UCX,...)
CPUs
Core Mochi libraries
Mochi components
External libraries
HEPnOS library

- Treat data analysis as a large HPC job
- Leverage HPC network
- Use all on-node resources for analysis (64 cores, KNL)
- Data loaded from HDF files in parallel (i.e., MPI program), written in batches from MPI ranks
- Analysis (client application) is an MPI code (for coordination) calling CAFAna for analysis
  - Mapping of events to MPI ranks done dynamically (work queues) in HEPnOS
  - Data loaded in 16K event batches (i.e., not per-event iteration!) from service, kept in C++ data structures to reduce serialization/deserialization
  - Handed out on the node in 64-event batches for processing

RESULTS SNAPSHOT

- With the in-memory backend the HEPnOS based workflow achieves 85% strong scaling efficiency at 128 nodes (8K cores) comparing to 1 node.
- Had to load in large batches to get these results.
- Ignore the “File-based” results for today
- Three configurations
  - Map tests use memory
  - RocksDB tests use local SSD
  - File tests read from HDF files (probably ignore for this discussion)



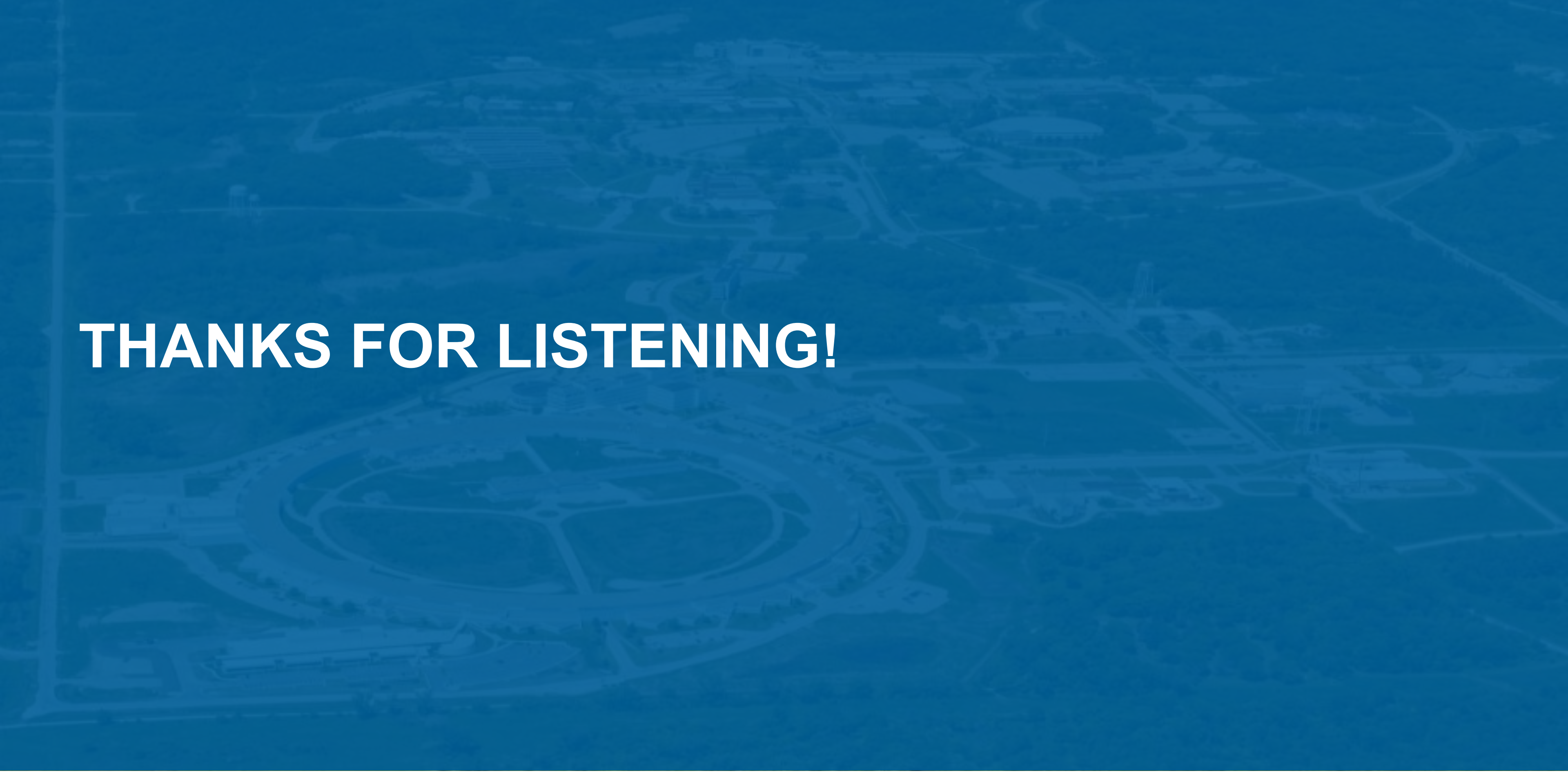
Plot illustrating the throughput (in slices processed per second) as a function of the total number of nodes used for processing the data using the existing traditional workflow and the HEPnOS based workflows. The performance of the HEPnOS based workflow is superior across all the different number of nodes used. The dots have been jittered to reduce over-plotting.

ONLY ONE MORE SLIDE!

Argonne
NATIONAL LABORATORY

OPPORTUNITIES?

- Benchmarking on DOE platforms
  - Talk with Florine on where to get these?
  - Also Analysis Description Language (ADL) benchmarks?
  - Coordinate on scaling, test file and DAOS (DFS and object) configurations
- Darshan and ROOT (and RNTuple) (and DAOS)
  - Capture, persist, analyze TTPerfStats and RNTuple performance data
  - Will need io_uring support for RNTuple file
  - May lead to performance optimization suggestions…
- RNTuple (internal) interface review
  - My team has built a number of I/O abstractions
  - May have some suggestions on the API between RNTuple and storage
    - RPageSource/RPageSink (if I understood Jakob right)?
- RNTuple HPC data service backend (i.e., like HEPnOS)?
  - Would be a research activity…



THANKS FOR LISTENING!

Argonne NATIONAL LABORATORY