ICARUS Signal Processing With HEPnOS

Sajid Ali *for* SciDAC4 HEPonHPC & HEPReco project
Fermi National Accelerator Laboratory
CHEP 2023

In partnership with:

Background: File-based workflows for grid-computing

- Current workflow uses a **file-based workflow** suited for **grid-computing:**
  - Input is a set of files, each containing ~30 events, ~10-20 Gb (for ICARUS).
  - Run each task (set of analysis modules) in the analysis pipeline as a **process.**
  - Size of a task is limited by available time, memory, disk space for output files.
  - Use **files to pass the results** between tasks.

- ICARUS uses the *art* framework to analyze events. Each *art* instance can concurrently process multiple events (each harnessing multiple-threads).

- Typically, one *art* process runs on a node and analyzes all the events in a file, producing one output file containing the **data products produced by the analysis and ones produced by preceding analysis**.

- Thus, we **typically have fewer *art* processes than files!**

Harness HPC resources using a distributed data store

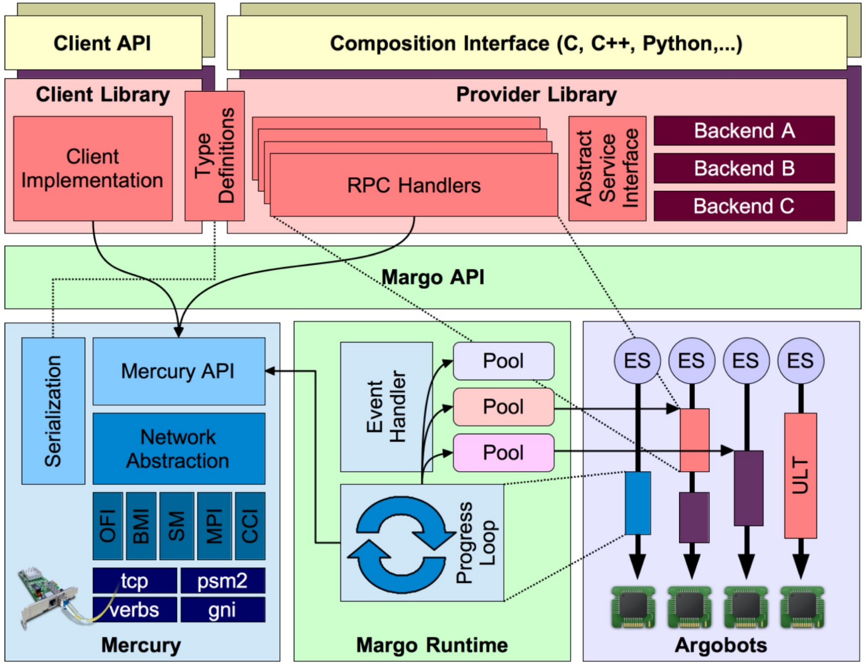
- Parallelism in analysis pipeline **limited by number of files.**
- **Remove this bottleneck** by **harnessing HPC resources** which have **low latency, high bandwidth interconnects** between nodes.

- We aim to achieve this by using a distributed data store which:
  - **Holds all data products for the duration of the analysis** on a set of (server) nodes
  - Allows *art* process to **store/retrieve data-products via the interconnect**

- By using it, we have now:
  - **Removed the constraint on parallelism** imposed by number of files.
  - **Reduced memory usage by each process** by fetching only the data-products required by a module and storing the results at the end of the module.
  - **Harnessed the interconnect** instead of the file-system to store/fetch data-products.

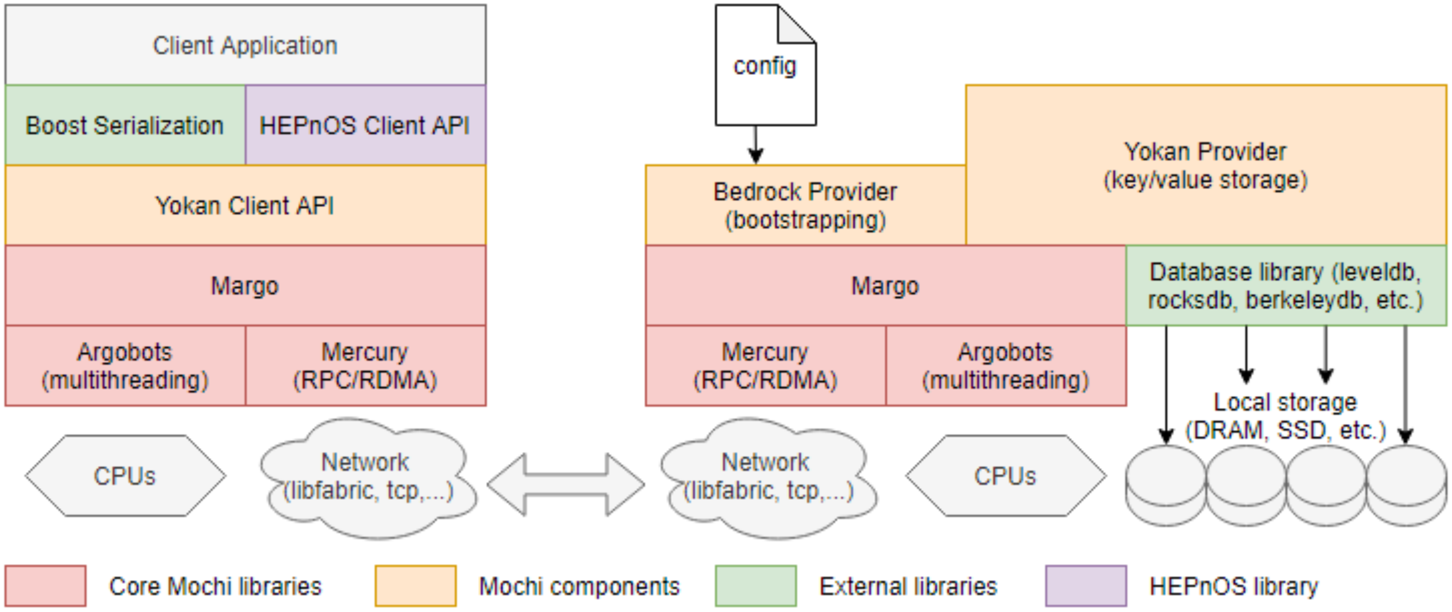
Background: Custom data services with Mochi

- Mochi microservices: a suite of re-usable components for building data services including:
- Mercury: RPC framework that can use a variety of transports, which supports bulk data transfers.
- Argobots: Lightweight user level threads to run tasks in execution streams.
- Margo: Utilities for argobots aware mercury requests.



Anatomy of a data service backed by mochi microservices. Illustration by Matthieu Dorier.

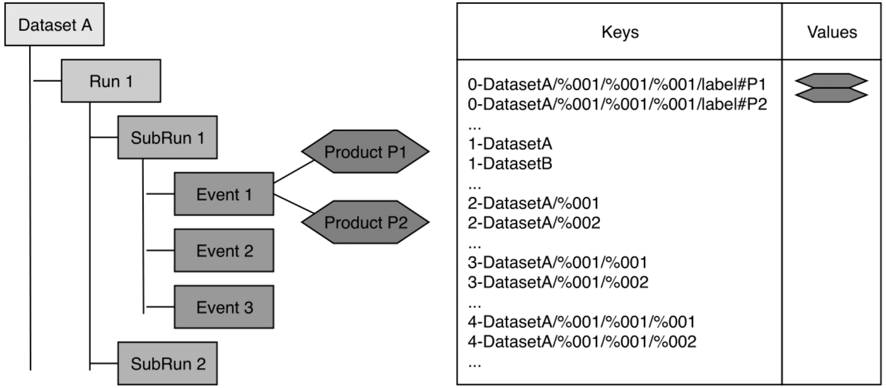
High-Energy Physics's new Object Store: Architecture



Architecture of HEPnOS: (Left) Client stack, (Right) Server stack. Illustration by Matthieu Dorier.

High-Energy Physics's new Object Store: Features

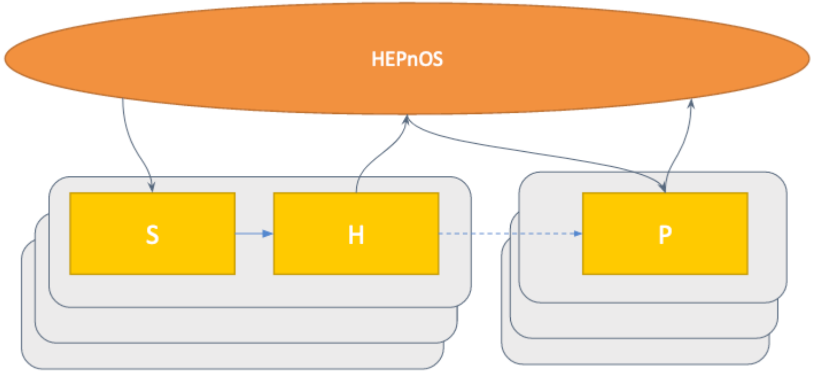
- Write-once, read-many access.
- Bulk ingest and iterative access.
- Eliminates software artifacts related to the filesystem and grid computing.
- Parallelism expressed at the event level instead of file level, allowing for better load balancing.



(Left) Hierarchical dataset organization. (Right) Representation in HEPnOS. Illustration by Matthieu Dorier.

Integrating HEPnOS into *art*

- HEPnOS allow for interaction via “queues”.
  - “Producer” mode allows one to put data products into the “queue”.
  - “Consumer” mode allows one to pop data products form the “queue”.
- Prototype I/O modules for *art* implemented corresponding to the two modes.
- This design allows for load-balancing at the event-level as **we can now have as many concurrent processes as events.**
- Compute resources can be partitioned based on the needs of tasks with this design, leading to efficient “pipelining”.



Data-flows for the first few steps of the ICARUS analysis pipeline. S: Signal Processing, H: Hit finding (JINST 17 (2022) 01, P01026), P: Pandora (Eur. Phys. J. C78, 1, 82 (2018)).

Preliminary evaluation

- Can we demonstrate event-level parallelism with HEPnOS by running as many processes as there are events in a file?

- We chose a typical ICARUS file with 30 events and ran the S and H steps using:
  - File based workflow: one *art* process with 4 threads analyzing all the events sequentially.
  - HEPnOS based workflow: varying number of *art* processes (each with 4 threads) fetching data products from HEPnOS, analyzing them and storing the resulting products back into HEPnOS.

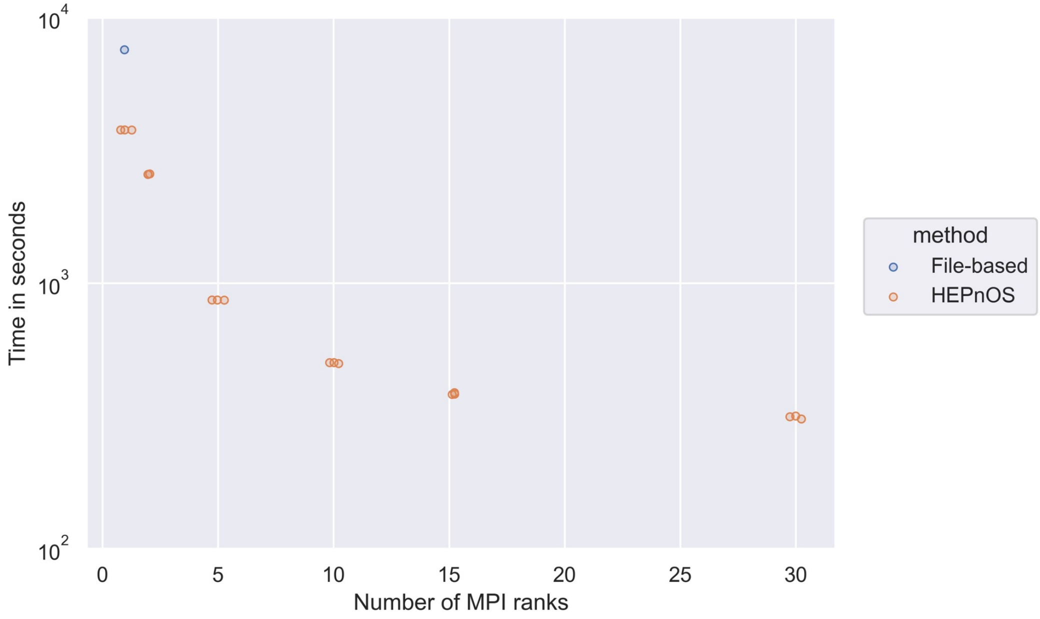
- We want to determine the scaling efficiency of the HEPnOS based workflow.

Preliminary evaluation

- Used 1 KNL node on Theta, with 64 (x86_64) cores for the analysis and 1 KNL node as the server for the *HEPnOS* workflow.

- Used MPI to launch multiple *art* processes.

- We observe a scaling efficiency of 75% at 10 MPI ranks and 66% at 15 MPI ranks.



Summary

- Demonstrated the use of a system-wide distributed event store suited to harness HPC interconnects.
- Demonstrated the ability to **harness workflow parallelism at the event-level** with *HEPnOS*.
- Work is currently underway on:
  - Demonstrating speedups using a larger dataset of 10k events (~4.5Tb).
  - Measurements of improvements from better pipelining.
  - Analyzing HEPnOS server configurations remove bottlenecks.

SciDAC team

- HEP and ASCR Collaboration
  - LHC and neutrino physics: N. Buchanan (CSU, NOvA/DUNE), P. Calafiura (LBNL, LHC-ATLAS), Z. Marshall (LBNL, LHC-ATLAS), S. Mrenna (FNAL, LHC-CMS), A. Norman (FNAL, NOvA/DUNE), A. Sousa (UC, NOvA/DUNE)
  - FASTMath Optimization: S. Leyffer (ANL), J. Mueller (LBNL)
  - RAPIDS Workflow, Data Modeling: T. Peterka (ANL), R. Ross (ANL)
  - Data science: M. Paterno (FNAL), H. Schulz (UC), S. Sehrish (FNAL)
  - J. Kowalkowski – PI (FNAL)
- Research Associates and Graduate students
  - Steven Calvez (CSU/PD), Pengfei Ding (FNAL), Matthieu Dorier (ANL/PD), Derek Doyle (CSU/GS), Xiangyang Ju (LBNL/PD), Mohan Krishnamoorthy (ANL/PD), Jacob Todd (UC/PD), Marianette Wospakrik (FNAL/PD), Orçun Yıldız (ANL/PD)
- http://computing.fnal.gov/hep-on-hpc/

Acknowledgement

- This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program, grant 1013935.

- DoE report number: FERMILAB-SLIDES-22-082-SCD.