The High Energy Physics Center for Computational Excellence (HEP-CCE) I/O and Storage Project: Plans and Progress

P. Van Gemmeren
for the HEP-CCE (IOS) project

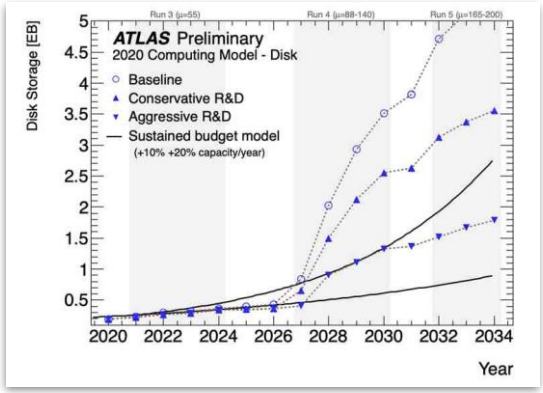
Software & Computing Round Table
Feb 9, 2021

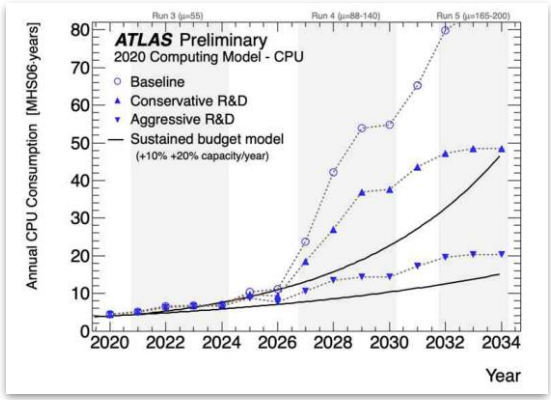
Why HEP-CCE?

HEP computing resource challenges

Notably 10x more data, 10x more complexity @ HL-LHC
→ need **PetaFlops *sustained*** per experiment

Project CPU **and** Storage resource **shortage**



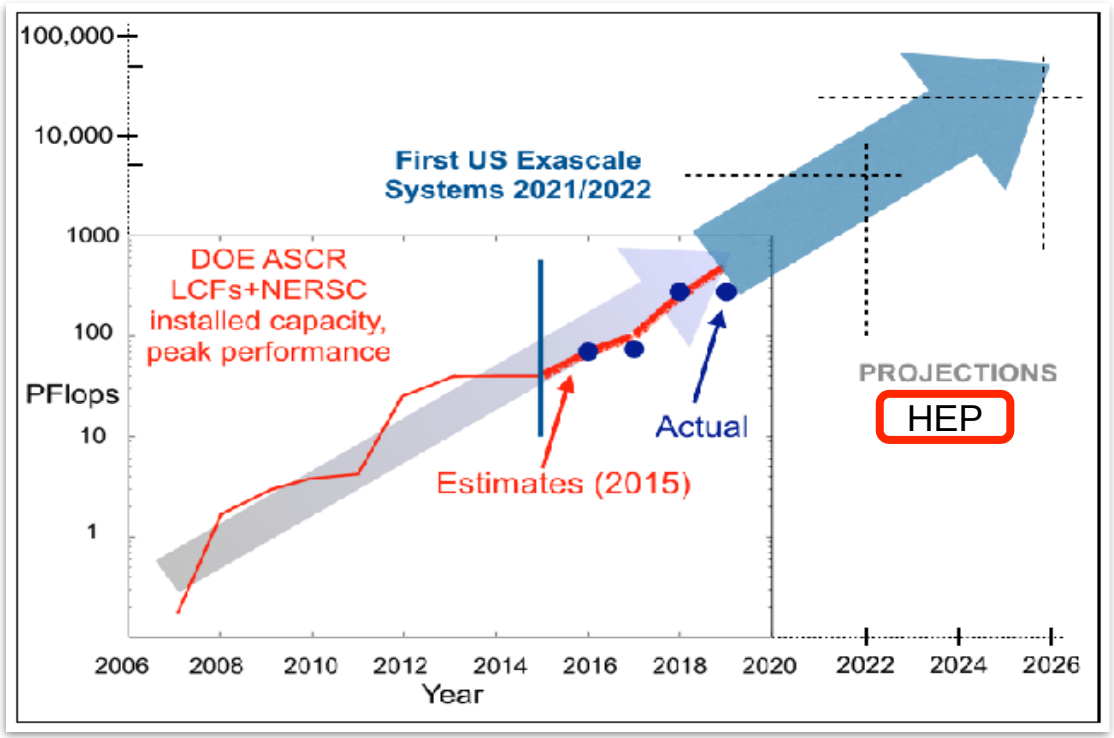


US long-term investment in HPCs

Platform of choice for simulations, and, more recently, data processing (light sources, LIGO, cosmology…)

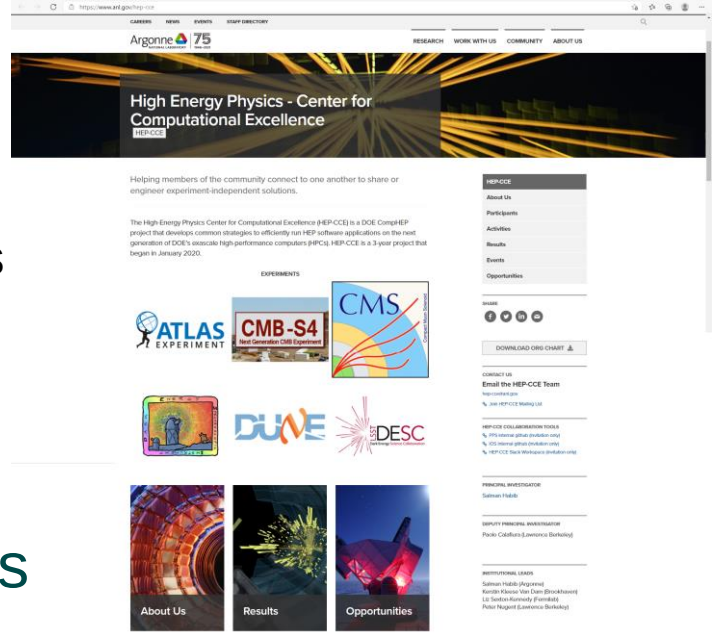
Expect O(10) **ExaFlops *peak*** performance by 2026
→ Challenging to run at 10% of peak

DOE would very much like HEP to partake
→ Most of the cycles provided by accelerators



What is HEP-CCE?

HEP-CCE



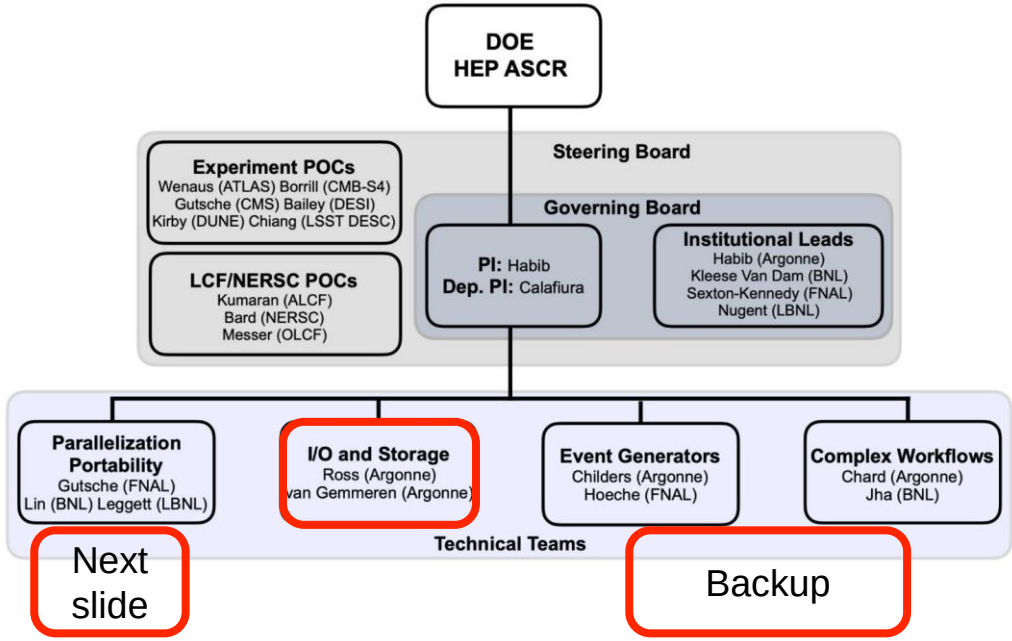
Three-year (2020-2023) **pilot** project

four US labs, six experiments, ~12 FTE, ~30 collaborators

1. Portable parallelization strategies
   - exploit massive concurrency
   - portability requirements
2. Fine-grained I/O and related storage issues
   - new data models (zero-copying, SOA,...)
   - event batching (XPU offloading)
3. Optimizing event generators
4. Running complex workflows on HPCs
   - main use case: cosmology surveys

Open collaboration

https://indico.fnal.gov/category/1053/
https://www.anl.gov/hep-cce



CCE/PPS: Goals and Year 1 priorities

Investigate a range of software portability solutions:

- Kokkos / Raja
- SYCL / dpc++
- Alpaka
- OpenMP / OpenACC

- products are rapidly evolving
- we have some hope of seeing the emergence of industry standards at the C++ language level

Port a small number of HEP testbeds to each language

- Patatrack Pixel Tracking (CMS)
- WireCell Toolkit (DUNE)
- FastCaloSim (ATLAS)

Define a set of metrics to evaluate the ports, and apply them

- Ease of porting, performance, code impact, relevance, *etc*

Make recommendations to the experiments

- Must address needs of both LHC style workflows with many modules and many developers, and smaller/simpler workflows

CCE/IOS: Goals

Parallel serialization/de-serialization of HEP data models

- Both single node and multi-node access patterns

Persistable data representations tuned for HPC storage systems

- Connection to PPS exploration of portable parallelization libraries
- Can benefit from Write-Once/Read-Many HEP access models

Accessing partial, partitioned or sub-event data blocks

- Matched to specific algorithm consumption requirement

Runtime memory mapping of data

- Exploit batched, vectorized, and data parallel operations and transforms on columnar data.
- Taking into account CPU-XPU communication

Open collaboration, meets bi-weekly Wednesday 2-3pm CST

https://indico.fnal.gov/category/1080/

CCE/IOS: Year 1 Activities

Darshan for ROOT I/O in HEP workflows on HPC.

- ROOT I/O is central to all HEP experiments. Measurements of its performance on HPC using tools like Darshan, could give valuable insights for possible improvements.

Investigate HDF5 as intermediate event storage for HPC processing

- In some workflows, such as the ATLAS EventService, temporary data is written to ROOT files. Moving this data to a parallel file format such as HDF5 could be beneficial.

Testing framework for understanding scalability and performance of HEP output methods

- An ability to simulate HEP output of specific data products (e.g., RECO, AOD, miniAOD) in different scenarios prepares us for deeper analysis of intermediate data storage options.

CCE/IOS: Year 1 Achievements

Darshan for ROOT I/O in HEP workflows on HPC.

- Darshan is a lightweight I/O characterization tool that can capture either condensed views or entire traces (DXT) of application I/O behavior
- Darshan is deployed at a number of DOE computing facilities (including ALCF, ANL LCRC, and NERSC) and has become a popular tool for HPC users to better understand their I/O workloads
  - While originally designed specifically for MPI applications, in the past year we have modified Darshan to also work in non-MPI contexts
  - Extends I/O instrumentation into exciting new HPC contexts, like HEP workflows that have traditionally not been based on MPI

CCE/IOS: Year 1 Achievements

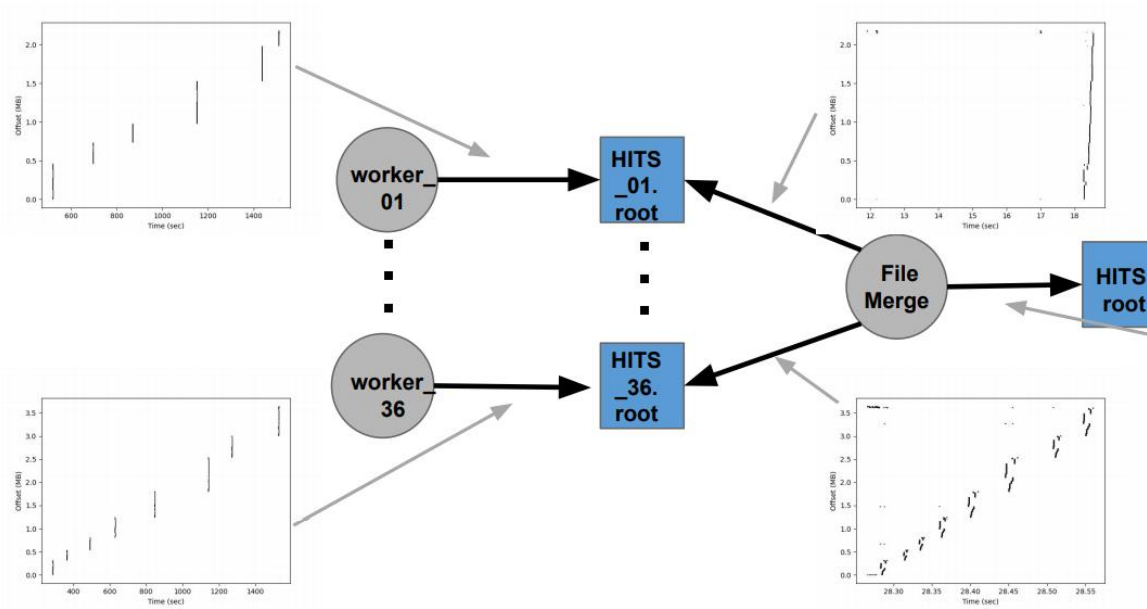
Darshan for ROOT I/O activity status.

- Integration of Darshan into HEP workflows like ATLAS, CMS, and DUNE can provide deeper understanding of their use of HPC storage resources. This understanding can be used to optimize workflow usage of ROOT as well as to make general improvements to the ROOT I/O library.

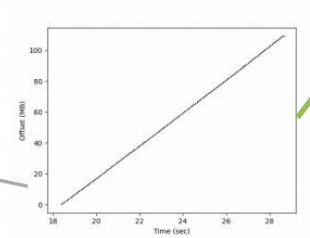
- Phase 1: Preparation
  - Introduction to Darshan, ROOT, and HEP workflow (ATLAS, CMS, DUNE) technologies and discussion of potential deployment strategies
- Phase 2: Prototyping
  - Deployment of Darshan for use in different HEP workflows (e.g., in containers)
  - Development of fork-safe Darshan instrumentation strategies (needed for ATLAS)
  - Development of Darshan analysis tools for understanding workflow I/O behavior

CCE/IOS: Year 1 Achievements

Darshan analysis of ATLAS, CMS and DUNE I/O.

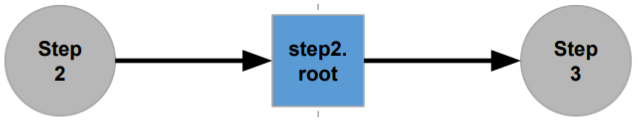


Darshan DXT traces for a portion of the ATLAS workflow



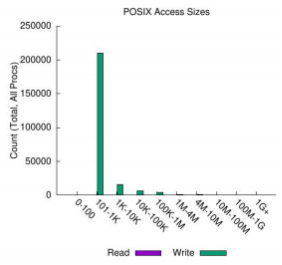
ATLAS: Multi-Process Simulation, creating many output files that are merged

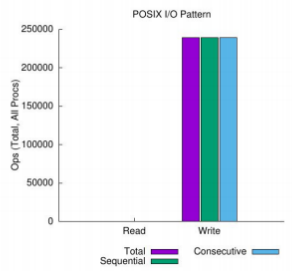
CMS: Overlay pile-up hits and digitization of hits

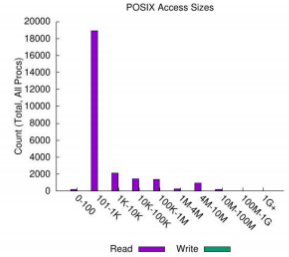


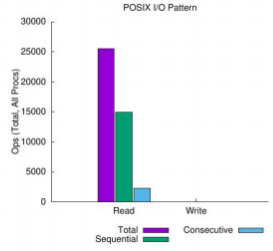
~13,500 MB written
~1,100 MB/sec
~20,000 IOPs/sec

~10,200 MB read
~2,400 MB/sec
~6,000 IOPs/sec









CCE/IOS: Year 1 Achievements

Darshan for ROOT I/O next steps.

- Investigate and develop workarounds for numerous obstacles to instrumenting ROOT I/O workloads
  - Unreliable I/O instrumentation for forked processes
  - Difficulty deploying Darshan in complex production environments (i.e., workflows employing containers)
  - Missing Darshan log data (e.g., due to instrumenting too many workflow files)
- With broader instrumentation coverage of ATLAS, CMS, DUNE workflows, we can turn focus to Darshan analysis tools to better understand flow of data, I/O access patterns, and achieved performance of different HEP data processing stages
  - Drive tuning decisions for ROOT usage and for ROOT implementation
  - New Python bindings for Darshan log utility library should ease development of new analysis tools

CCE/IOS: Year 1 Achievements

Investigate **HDF5** as intermediate event storage for HPC processing.

- HDF5 (Hierarchical Data Format) is a portable, self-describing file format designed to store large amounts of data
  - It is maintained by the HDF Group [https://www.hdfgroup.org]
  - It is widely available at HPC centers, and easily installable on laptops
  - It supports parallel IO using MPI, and has special drivers tuned for parallel file systems at HPC centers
- A few key abstractions are:
  - datasets, which are multidimensional arrays of homogeneous types,
  - groups, which are containers of datasets and other groups, and
  - attributes, which are small metadata objects to describe groups and datasets
- Allows efficient columnar data access for the “required” data products

CCE/IOS: Year 1 Achievements

Investigate HDF5 as intermediate event storage for HPC processing activity status.

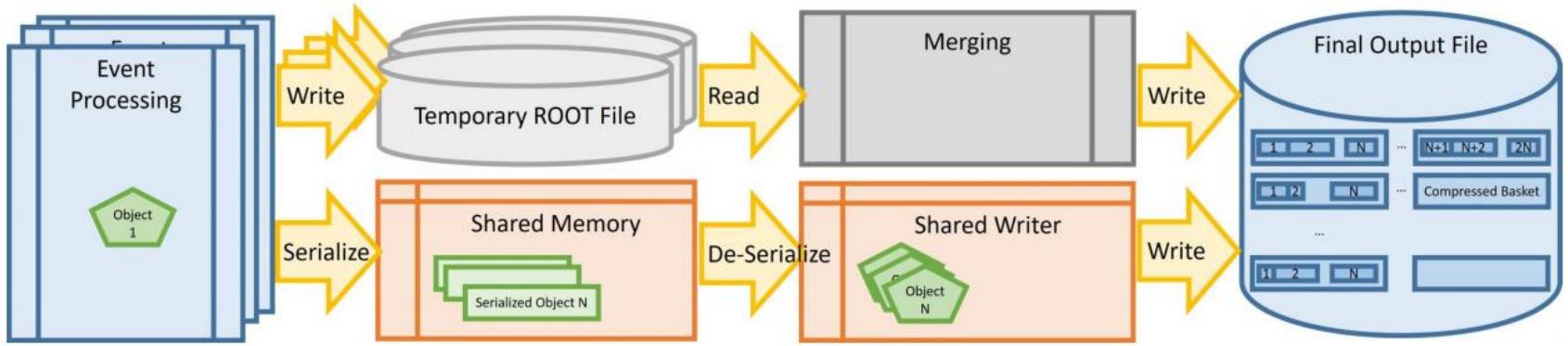
- The focus of this activity is to explore the use of HDF5 files for writing HEP data products that have already been serialized using ROOT serialization.
- We are interested in developing an experiment-independent approach.
- We are currently using a multi-threaded testing framework developed as part of the CCE project to work on the use of HDF5.
- This work is the first attempt to write intermediate output in HDF5 style.
- We have demonstrated the efficient and high performing data access and hence subsequent analysis by using HDF5 representation of the analysis-ready data in SciDAC (HEP on HPC) project.

CCE/IOS: Year 1 Achievements

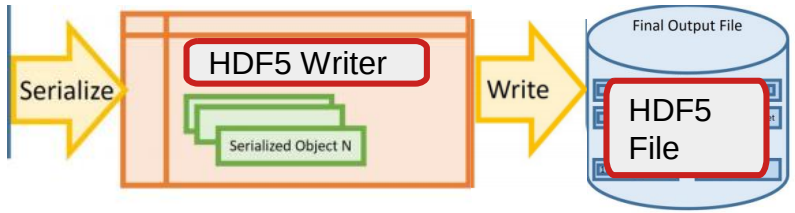
Investigate HDF5 as intermediate event storage for HPC processing: ATLAS showcase

ACAT 2017

The Shared Writer collects output data objects from all AthenaMP workers via shared memory and writes them to a single output file. This helps to avoid a separate merge step needed in regular AthenaMP processing.



Prototype HDF5 extension 2020



CCE/IOS: Year 1 Achievements

Investigate HDF5 as intermediate event storage for HPC processing next steps

- HDF5 tuning
  - Chunking: Looked at some prelim numbers, using 128 chunk size, which seems to work fine
  - Asynchronous I/O: Use a background thread to perform I/O
- Compression
  - In both serial and parallel mode, and combined with chunking
- Using node-local storage for storing intermediate HDF5 files
- Parallel I/O (using multi-process MPI-based writes)
- Multi-threaded HDF5
  - There is a feature branch available with simple read/write patterns, which we have in our use case, that we should look into
- Explore direct storage access from GPUs

CCE/IOS: Year 1 Achievements

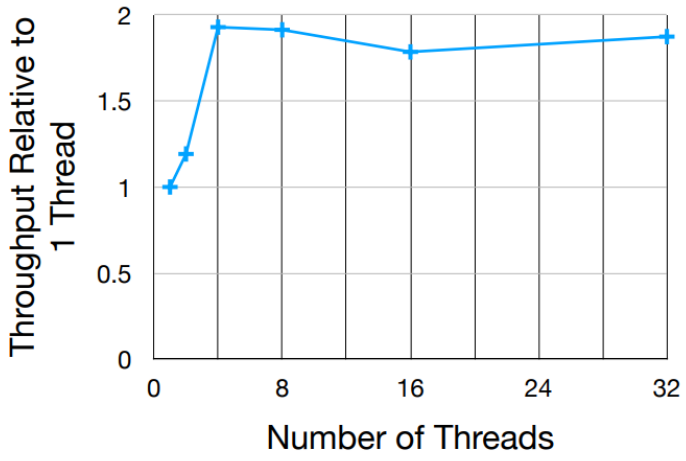
Testing framework for understanding scalability and performance of HEP output methods

- Mimic the characteristics of a HEP data processing framework
  - Similar multi-threaded behavior
  - Similar I/O behavior
  - Should reasonably behave like CMS, ATLAS and DUNE frameworks
- Easily try different I/O implementations
  - Choose what to use via command line arguments
- Experiment agnostic
  - With ability to read actual experiment ROOT files
  - ROOT will dynamically load serialization/deserialization plugins as needed
- Make it easier to perform performance measurements
  - I/O performance
  - threaded scaling performance

CCE/IOS: Year 1 Achievements

Testing framework for understanding scalability and performance of HEP output methods

- Have a flexible I/O testing framework
  - Can test input and output formats and approximate HEP job timings
- Has lead to thread scaling performance of ROOT serialization
  - On second round of improvements

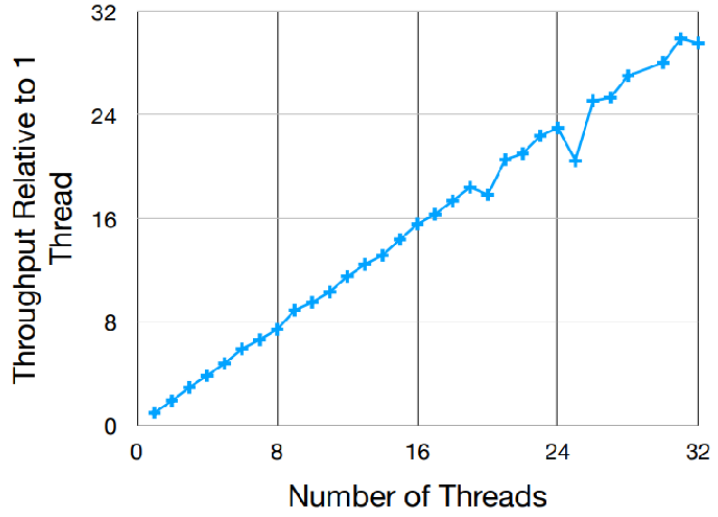


Frequent setting of an atomic value

```
Version_t GetClassVersion() const {
      fVersionUsed = kTRUE;
      return fClassVersion; }
```

Changed to

```
Version_t GetClassVersion() const {
      if (!fVersionUsed.load())
         fVersionUsed = kTRUE;
      return fClassVersion; }
```



Additional: Year 2 Priorities

PPS

New activity: Event Batching

- in collaboration with IOS

IOS

New activity: Data Model and storage for CPU/XPU communication

- Cross-educate on ASCR and HEP activities
- awkwardarray, dataframes, ATLAS/CMS columnar AODs

EG

- Madgraph port to CUDA/Kokkos in collaboration with Madgraph team.

Outlook

PPS and IOS have made significant progress in Year 1

Project presented in multiple venues and conferences with positive feedback

- very good interactions with experiments and tool developers

Ramping up effort (big challenge these days!)

- recruiting more developers from the experiments and DOE/ASCR experts
- several hires of postdocs and summer students

Very hopeful that there will be a significant impact on the experiments

- HEP-CCE will produce **strategies** tested on **prototypes**
  - Production-level implementations require *direct experiment involvement*.

Thanks!

https://www.anl.gov/hep-cce

gemmeren@anl.gov

Cast of Characters

PPS

- Taylor Childers (ANL)
- Mark Dewing (ANL)
- Zhihua Dong (BNL)
- Oli Gutsche (FNAL)
- Michael Kirby (FNAL)
- Matti Kortelainen (FNAL)
- Kyle Knopfel (FNAL)
- Charles Leggett (LBNL)
- Meifeng Lin (BNL)
- Vincent Pascuzzi (LBNL)
- Peter Nugent (LBNL)
- Liz Sexton-Kennedy (FNAL)
- Yunsong Wang (LBNL)
- Sam Williams (LBNL)
- Haiwang Yu (BNL)

not official list: taken from those who call into weekly meetings

IOS

- Peter van Gemmeren (ANL)
- Rob Ross (ANL)
- Doug Benjamin (ANL)
- Suren Byna (LBNL)
- Philippe Canal (FNAL)
- Matthieu Dorier (FNAL)
- Chris Jones (FNAL)
- Kenneth Herner (FNAL),
- Patrick Gartung (FNAL).
- Rob Latham (ANL)
- Liz Sexton-Kennedy (FNAL)
- Saba Sherish (FNAL)
- Shane Snyder (ANL)
- Torre Wenaus (BNL)
- Jakob Blomer (CERN)

CCE/EG: Event Generators

Why improve event generators?

- Event generation might consume more resources at HL-LHC that we currently extrapolate
- Generators code generally not well written ➜ scaling issues even on HTC resources, never mind HPC

Building on previous improvements

- Parallelized Pythia particle-level event generation
- Improved performance of Sherpa, particularly I/O (HDF5)
- Novel integrator using Neural Networks and Normalizing Flows

CCE plans

- Complete rewrite of matrix-element generator for CPUs & GPUs
- Paradigm shift from “best of scaling” to “best for computation”
- Exploration of different frameworks (low-level, Kokkos, ...)
- Alternatively try to revive existing implementations (HELAS/MG)
- Coordinate with HSF and worldwide effort

CCE/CW: Complex Workflows

Why complex workflows?

- Workflows enable representation and execution of analyses composed of heterogeneous components
  - Single node codes, multi-node MPI applications, scripts, binaries, glue code, etc.
- HPC systems and environments are not designed for such workloads
  - Millions of tasks, diverse and varying requirements (cores, duration, CPU/GPU), co-scheduling and dynamic sizing

CCE plans

- Support development and use of workflows within target domains
- Apply a modular, interoperable, and inclusive approach
  - Leverage/combine components from existing workflow systems
- Democratize access to advanced workflow capabilities and reduce overheads on small collaborations
  - Enable use of accelerators, heterogeneous hardware, without knowledge of low-level programming