HEP-CCE Status

SCD Projects Meeting - R&D
15 September 2022

All Hands Meeting

- HEP-CCE is organizing an All Hands Meeting for October 11-13 at Berkeley lab
  - https://indico.fnal.gov/event/56044/
- Intention is to
  - Carry out a self-assessment of where we are at this point as a project (roughly two and a half years after the start of the planned activities) for PPS, IOS and EG
  - Plan activities until the end of the current funding cycle (FY23)
  - Discuss status of activities and proposals for beyond FY23

Fine-Grained I/O and Storage (IOS)

- Class Layout Study
  - Looked at size of objects stored in CMS' archival format (AOD)
  - Determined objects with largest compressed size on disk were best to optimize
    - Time spent in output was proportionally to compressed size
  - Found best to optimize objects for compression
    - Optimizing for easy serialization was not guaranteed to help compression
  - Biggest single win was removing *unnecessary* information
    - Dropping *unnecessary* particles from GenParticle list gave 9% reduction in file size
  - ROOT's lossy compression also helpful for compression
    - Reducing precision for P4 info can reduce file size by 10% across many data products

Fine-Grained I/O and Storage (IOS)

- Parallel HDF5 Study
  - Developed parallel HDF5 output modules for the MPI-based version of the root serialization test framework, two implementations based on different event distribution approach
  - Ran initial tests on Cori (Haswell nodes) to understand scaling behavior of only one implementation that uses the number of events to be processed to be known in advance. The settings e.g. what batch size, HDF5 chunk size, and number of threads to use are taken from our prior runs on Cori that resulted in the best performance of the serial HDF5 output module. For these tests, up to 8 nodes are used so far.
    - Observe almost perfect scaling for 1 MPI process/node
      - by keeping the number of events to be processed constant and by increasing the number of MPI processes
      - by increasing the number of events to be processed with the number of MPI processes
    - Investigating and analyzing results already obtained with multiple MPI processes per node
  - What is Next:
    - Run more tests for MPI startup and communication overhead cost and large scale studies followed by a technical paper/report.

Portable Parallelization Strategies (PPS)

- Patatrack
  - A frozen, standalone version of CMS heterogeneous pixel track and vertex reconstruction
    - "End-to-end", with mock framework and build system
  - Current status

	Implementations							
	CPU Serial	**CUDA** (original)	**HIP**	**Kokkos**	**Alpaka** (by CERN team)	**std::par**	**SYCL** (also by CERN team)	**OpenMP**
NVIDIA								
AMD				Crashes randomly				
Intel				Does not compile (Eigen)				
CPU				Serial, POSIX threads	Serial, TBB			

Completed
In progress
Not started

Portable Parallelization Strategies (PPS)

- Patatrack: recent updates
  - Work with OpenMP-Target, std::par, and SYCL progresses
- Plans
  - Test on AMD and Intel GPUs on JLSE machines at Argonne
  - Continue with direct SYCL, OpenMP-Target, and `std::par`

Portable Parallelization Strategies (PPS)

Completed
In progress
Not started

- Propagation-to-r (p2r)
  - Kernel for track propagation in radial direction extracted from mkFit
  - Current status

	Implementations								
	TBB	CUDA	HIP	Kokkos	Alpaka	std::par	SYCL	DPL	OpenACC
NVIDIA									
AMD									
Intel									
CPU									

Portable Parallelization Strategies (PPS)

- Propagation-to-r (p2r): recent updates
  - First SYCL/std::par/oneDPL implementation (by Alexei) uses a different memory layout than CUDA/Kokkos/Alpaka versions
    - Storing per-track information in local, per-thread variable (A) instead of using shared memory (B)
  - Implemented new version of Kokkos+Alpaka using the (A) layout
    - (A) layout is much faster on GPU than (B) layout (when omitting data transfers)
    - (A) layout is ~30% slower on CPU than (B) layout (does not vectorize as well)
  - Can (in principle) make fair comparison of Kokkos/Alpaka/SYCL versions on NVIDIA/AMD/Intel GPU
    - Need hip-SYCL for AMD GPU for SYCL version
- Plans
  - Measure each backends of all the implementations on JLSE hardware
  - Present results at ACAT 22