High Energy Physics
Center for Computational Excellence

C. Leggett

ATLAS IBMeeting
July 15 2020

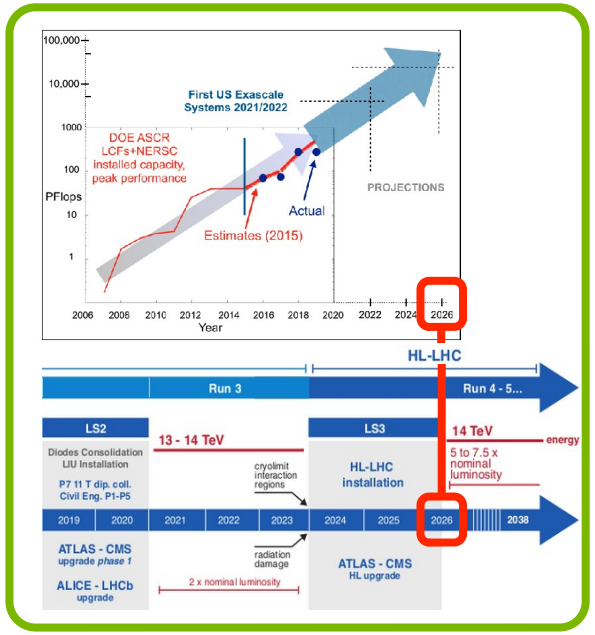
Origins of HEP-CCE

A 2013 study by a Topical Panel charged to asses the then-current status of essential enabling technologies in computing and simulation for HEP

- suggest paths forward that would improve the effectiveness of the field
  - one of which was a proposal for a distributed “Center of Excellence”

“Center would provide opportunities for a cross-cutting R&D program that includes compute and data intensive elements addressing the challenges of next-generation architectures, innovative approaches to the use of high performance computing for HEP science, and data-intensive collaborations with the ASCR community, including networking and data transport, as well as partnerships with industry when appropriate.”

Next Generation Computing and HEP Landscapes



Expecting an order of magnitude increase in data rates and exponentially increasing data complexity starting in mid 2020s

- a few PFlops needed by mid decade
- require new methods and capabilities for data processing, simulation and management

Traditional grid resources are will struggle to keep pace

Computational resources at ASCR DOE centers are expected to be ~exascale in early 2020s and 10x more by mid decade: 10^4 over HEP needs

- Friend or foe? Non traditional, heterogeneous architectures
- 60 – 90% of computing power in GPU (and possibly other) accelerators

BUT HEP community is in general not prepared to use these types of HPCs

Asses whether, and in what way HPC systems can met the needs of experiments, and to find portable solutions that can be deployed across multiple platforms

Current and Near Future Heterogeneous HPCs

Complex ecosystem

		Accelerators				
		Intel	**NVidia**	**AMD**	**FPGA**	**Other**
CPU	**Intel**	Aurora	Cori Piz Daint Tsukuba MareNostrum		Tsukuba	
	AMD		Perlmutter	Frontier El Capitan		
	IBM		Summit Sierra MareNostrum			
	Arm		Wombat			Astra*
	Fujitsu					Fugaku

- *Amazon EC2 P3*
- *Google Cloud TPU*
- *Microsoft Azure*
- *Intel DevCloud*

- Exploit massive concurrency of these HPCs
- Be portable: run with little modification on all platforms (and get the same answer)

Current HEP-CCE Mandate

Connect the DOE ASCR and R&D programs to HEP computing across all three HEP science efforts:

- Cosmic, Energy and Intensity Frontiers
- ATLAS, CMS, DUNE, LSST and others

Promote new capabilities to aid HEP theorists and experimentalists in overcoming computational challenges

Aid the HEP office in organizing and carrying out a number of cross-experiment topical investigations, studies and evaluations

- edge services for accessing HPC systems and data portals
- porting HEP applications to HPCs
- use of containers for HEP software management and I/O performance enhancement on HPCs
- petascale data transfer project with ESnet to move data across ASCR and NSF facilities
- topical workshops and hackathons for outreach and education
- promotion of software “best practices” that aid in HPC adoption
- assessments of HEP software products regarding their HPC suitability
- summer student programmes

Four HEP-CCE Sub-Groups

i Portable parallelization strategies
- exploit massive concurrency
- portability requirements

ii Fine-grained I/O and related storage issues
- new data models
- event subdivision

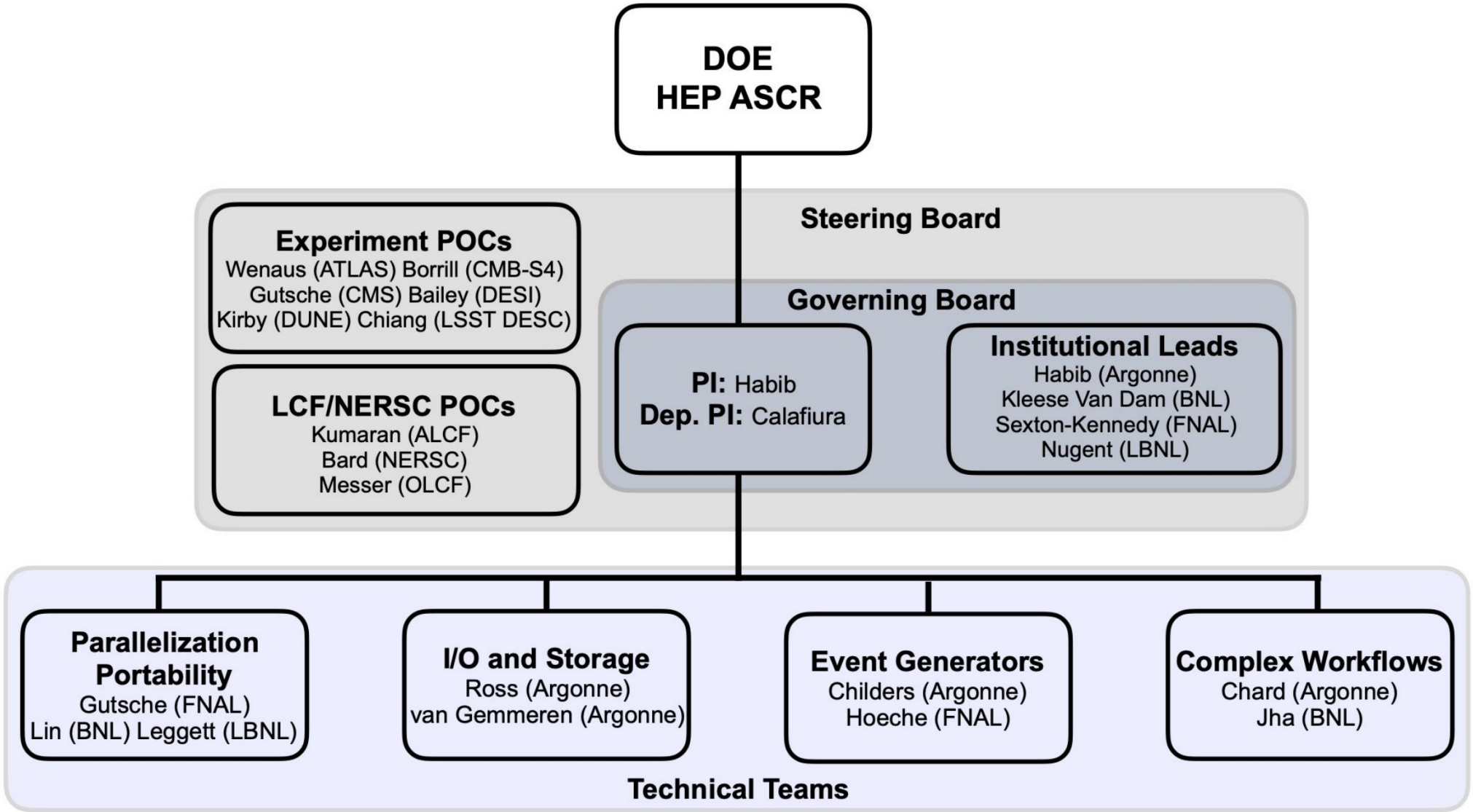
iii Optimizing event generators

iv Running complex workflows on HPC systems
- dynamic workflows with GPU exploits

3 year pilot project (Oct 2020 – Oct 2022)

Involves close collaboration between the domain scientists, the ASCR experts and the wider set of users who work with the experimental data, to ensure that the proposed solutions do in fact properly address the needs of the experiments

Organizational Chart



CCE/PPS: Goals

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/PPS: Software Support Overview

	OpenMP Offload	Kokkos	dpc++ / SYCL	HIP	CUDA	Alpaka
NVidia GPU			*Intel/codeplay*			
AMD GPU		*prototype*	*via hipSYCL*			
Intel GPU						*very early development*
CPU						
Fortran						
FPGA						*possibly via SYCL*

Supported

Under Development

3rd Party

Not Supported

We are seeing rapid and ongoing development

CCE/PPS: CMS Patatrack Pixel Tracking

Goal is demonstrate that part of the CMS HLT Pixel local reconstruction can be efficiently offloaded to a GPU

- reconstruct pixel-based tracks and vertices on the GPU
- leverage existing support in CMSSW for threads and on-demand reconstruction
- minimize data transfer

Copy the raw data to the GPU

Run multiple kernels to perform the various steps

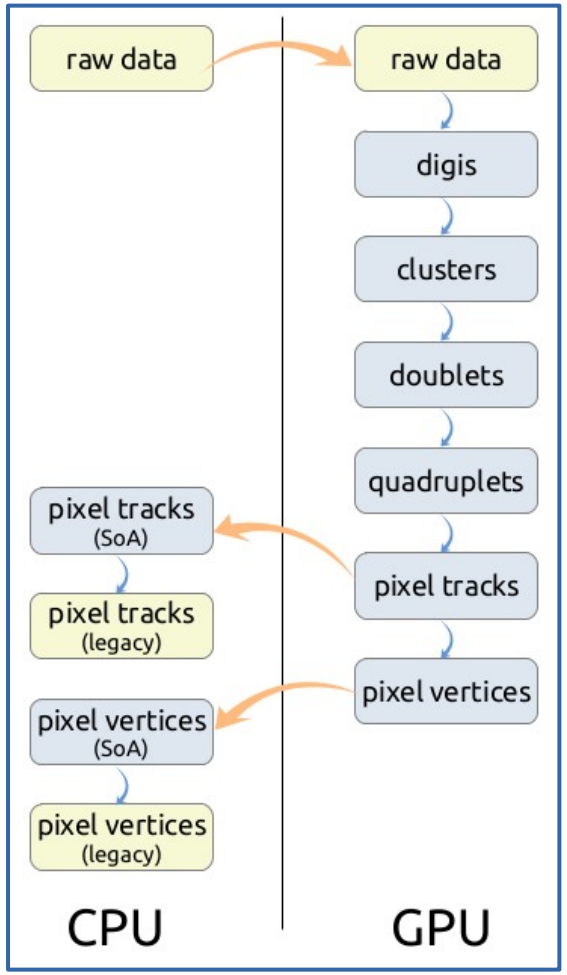
- decode the pixel raw data
- cluster the pixel hits (SoA)
- form hit doublets
- form hit quadruplets (or ntuplets) with a Cellular automaton algorithm – clean up duplicates

Take advantage of the GPU computing power to improve physics

- fit the track parameters (Riemann fit, broken line fit) and apply quality cuts
- reconstruct vertices

Copy only the final results back to the host (optimised SoA)

- convert to legacy format if requested



CCE/PPS: Wire-Cell Toolkit

Much of DUNE software is based on LArSoft, which is single-threaded and has high memory usage.

Wire-Cell Toolkit (WCT) is a new standalone C++ software package for Liquid Argon Time Projection Chamber (TPC) simulation, signal processing, reconstruction and visualization.

- Written in modern C++ (c++17 standard)
- Follows data flow programming paradigm
- Supports both single-threaded and multi-threaded execution with the choice determined by user configuration.
  - MT graph execution supports pipelining, more than one "event" may be in flight through the flow graph.
- Runs from stand-alone command line program or from a LArSoft module.

WCT includes central elements for DUNE data analysis, such as signal and noise simulation, noise filtering and signal processing

- CPU intensive; currently deployed in production jobs for MicroBooNE and ProtoDUNE
- Some algorithms may be suited for GPU acceleration

Preliminary CUDA port of the signal processing and simulation modules show promising speedups

CCE/PPS: ATLAS FastCaloSim

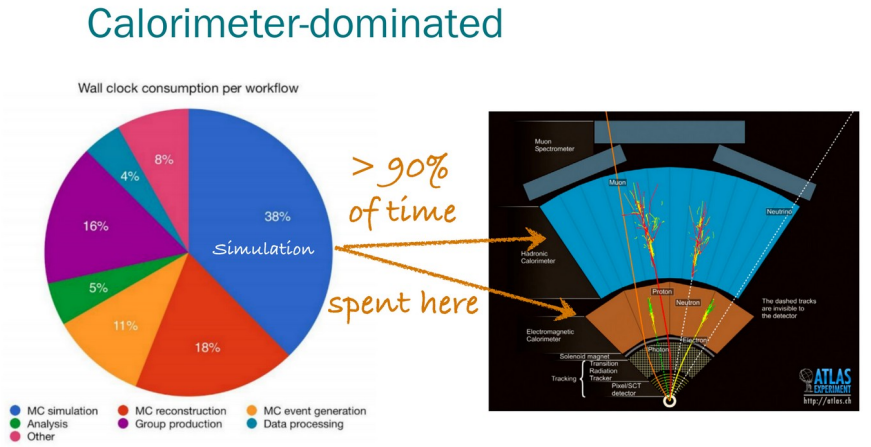
ATLAS Calorimeter simulation measures the energy deposition of O(1000) particles after each collision

Full detailed simulation uses Geant4

- very slow due to complex LAr Geometry

Fast calorimeter simulation uses parametrization

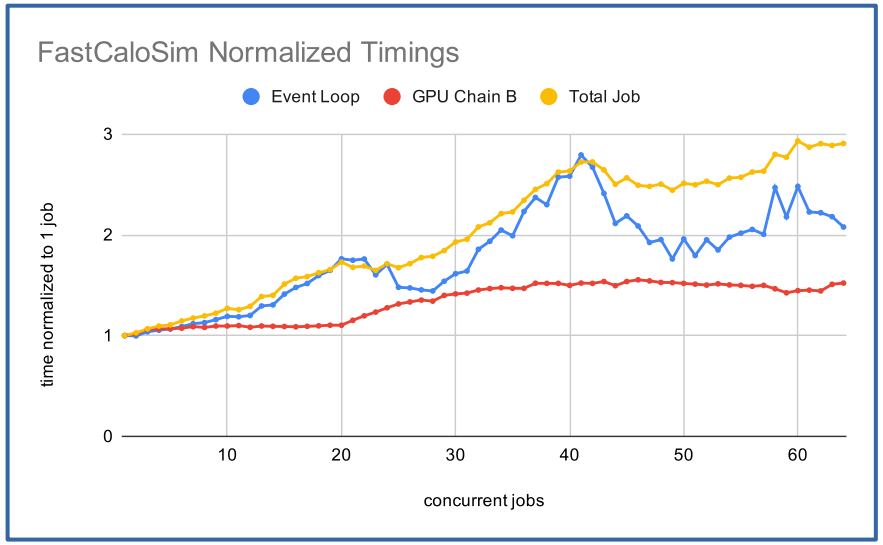
- less accurate, but much faster than Geant4



FastCaloSimV2: a relatively self-contained code base for fast ATLAS parametrized calorimeter simulation

BNL group has already created a CUDA port

- modify/flatten data structures (eg Geometry) to offload to GPU
- multi-stage CUDA kernels to generate histograms
- current efficiency hampered by small work sizes
- need to use more particles or gang events



U.S. DEPARTMENT OF ENERGY | Office of Science
Argonne NATIONAL LABORATORY
BROOKHAVEN NATIONAL LABORATORY
Fermilab
BERKELEY LAB
Bringing Science Solutions to the World

CCE/PPS: Metrics: Evaluation of PPS Platform

Ease of learning (experts and novices) and extent of code modification

Code conversion

- CPU → PPL / CUDA → PPL / PPL → PPL

Impact on other existing code

- Event Data Model
- does it take over main(), does it affect the threading or execution model, *etc*

Impact on existing toolchain and build infrastructure

- do we need to recompile entire software stack?
- cmake / make transparencies

Hardware mapping

- current and future support
- new architectures

Feature availability

- reductions, kernel chaining, callbacks, *etc*
- concurrent kernel execution

Ease of Debugging

Address needs of all types of workflows

- scaling with # kernels / application
- scaling with # developers
- compute vs memory bound

Long-term sustainability and code stability

- Support model of technologies ➜ stability of implementation if underlying libraries (CUDA) change
- CUDA is going to be around for a long time, what about the portability solutions?
- Long term support for technologies by vendors

Compilation time

- separate builds for different architectures?

Performance: CPU and GPU

- degradation of CPU code?

Validation

Aesthetics

- compatibility with C++ standards

U.S. DEPARTMENT OF ENERGY | Office of Science
Argonne NATIONAL LABORATORY
BROOKHAVEN NATIONAL LABORATORY
Fermilab
BERKELEY LAB Bringing Science Solutions to the World

CCE/PPS: Current Status

Currently porting Patatrack Pixel Tracking, FastCaloSim and WireCell Toolkit to Kokkos v3.1

- build system integration
  - integrating with existing make / cmake build systems can be challenging
- kernel and data structure identification
  - replacing CUDA malloc/memcpy with **`Kokkos::Views`** is intrusive
- preliminary porting of kernels

CCE/PPS: Current Status

Currently porting Patatrack Pixel Tracking, FastCaloSim and WireCell Toolkit to Kokkos v3.1

- build system integration
  - integrating with existing make / cmake build systems can be challenging
- kernel and data structure identification
  - replacing CUDA malloc/memcpy with **`Kokkos::Views`** is intrusive
- preliminary porting of kernels

Recently finished porting FastCaloSim

- kernels are *heavily* launch latency limited
- cost of launch is similar to kernel execution time
- actual kernel run times are comparable in CUDA and Kokkos/CUDA

FCS: total time, wrapping kernel calls on host with std::chrono

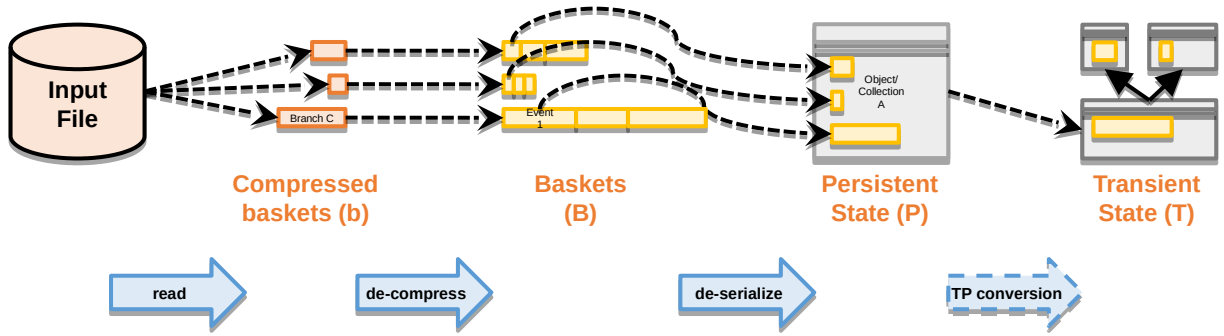
sum over 10000 events	time / s	factor **slower** than CUDA				
	CUDA	**kk/CUDA**	**kk/HIP**	**kk/Serial**	**kk/OMP**	**kk/pth 15**
clean	0.140	2.8	20.1	7.7	2.3	1.6
sim_A	0.490	1.2	37.9	36.9	11.5	11.2
sim_ct	0.150	1.4	1.4	12.9	2.0	1.7
copy d->h	0.250	1.3	1.0	0.0	0.0	0.0
event loop	3.300	1.0	11.9	7.3	2.7	2.6

NVidia RTX 2080 Super, AMD Vega 56, 72 Core Xeon 5220

CCE/IOS: Goals

The project will focus and concentrate effort on:

- Efficient serialization/de-serialization of data representations under parallel I/O models
  - both single node and multi-node access patterns
- Development of persistable data representations that can be tuned for access on HPC storage systems and optimized for HEP I/O patterns.
  - may involve accelerator technologies
  - can benefit from Write-Once/Read-Many access models
- Optimization of reads of partial, partitioned or sub-event data blocks from storage which are matched to specific algorithm consumption requirement
- Optimization of runtime memory mapping of data to exploit batched, vectorized, and data parallel operations and transforms on columnar data.



CCE/IOS: Year 1 Plans

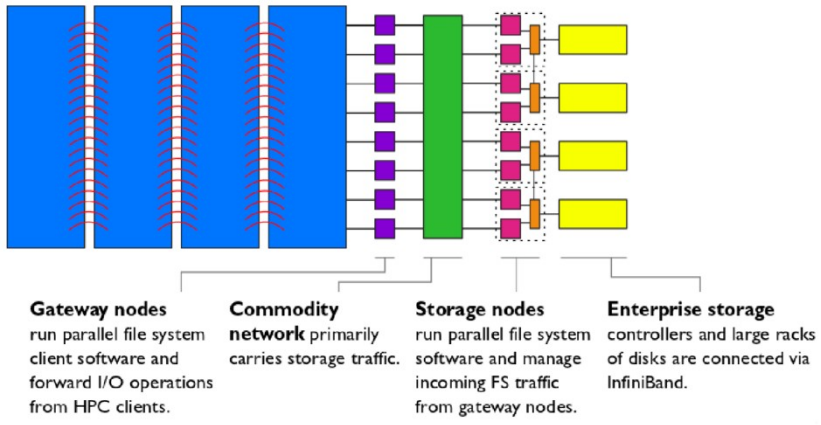
1st quarter: Document i/o patterns and EDM

- Give short presentations on background topics with Q&A

2nd quarter: Performance of HEP experiment benchmarks on Grid resources

- ROOT: Optimizable for HPC, xCache, Instrument ROOT I/O patterns.
- ATLAS: EventService Simulation (fined-grained (event-wise) processing).

3rd quarter: produce benchmarks either by packaging experiments workflows or by building synthetic benchmarks



CCE/IOS: Current 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.

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
  - R&D related to portability, scalability, performance, and reproducibility

Outlook

Several projects are well underway and have made significant progress

- CW and EG have not yet fully ramped up, as they are much smaller in scope and will not receive full funding until FY21
- we are considering expanding the scope of some of the PPS projects

Project introduction and status reports have been presented in multiple venues and conferences with positive feedback

- very good interactions with the various experiments

Increasing levels of effort

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

Funding levels currently look good (~ $4M per year)

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

fin

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)
- Rob Latham (ANL)
- Liz Sexton-Kennedy (FNAL)
- Saba Sherish (FNAL)
- Shane Snyder (ANL)
- Torre Wenaus (BNL)
- Jakob Blomer (CERN)

Kokkos

Kokkos provides portability via various backends: OpenMP, CUDA, (tbb), *etc*

Single source C++, standard dependent on backend (nvcc limits to C++14)

Abstractions usually provided via C++ header libraries

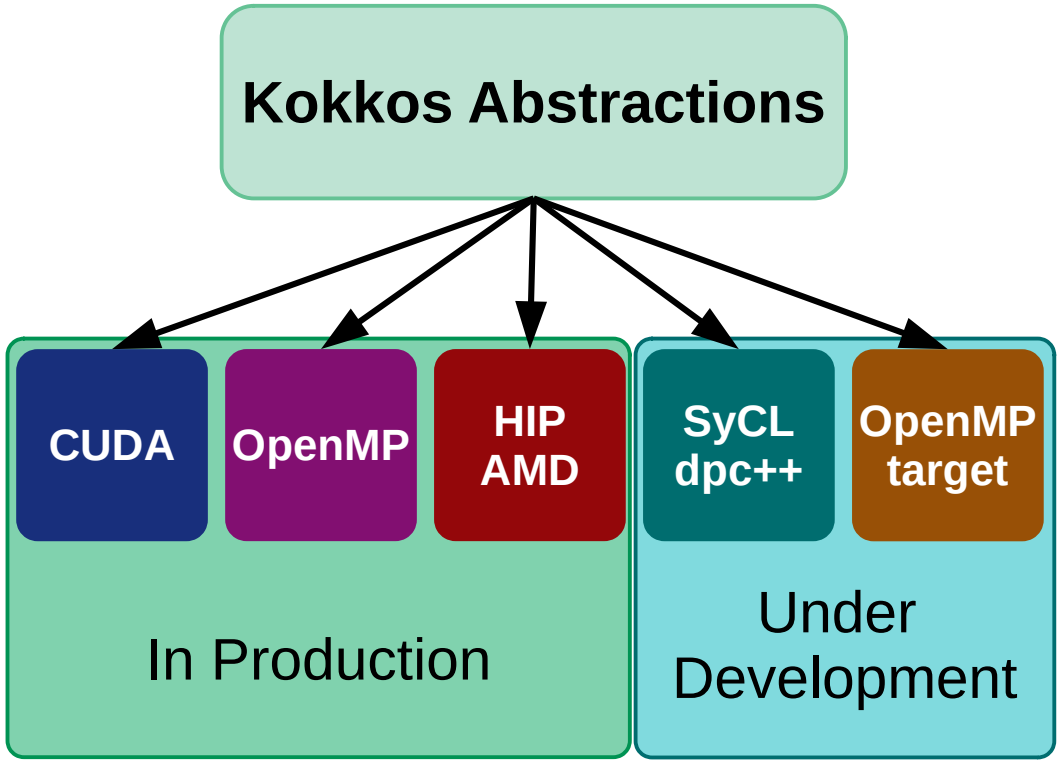
- parallel_for, reduction, scans

Data interaction *via* **`Kokkos::Views<...>`**

- explicit memory movement
- memory space selected via policy
- impact on C++ code

Execution is bound to an Execution Space

- device backend must be selected at compile time



SYCL / dpc++

SYCL 1.2.1 plus Intel extensions (some of which are from SYCL 2.X)

Single source C++ (understands C++17)

Explicit memory transfers not needed

- builds a DAG of kernel/data dependencies, transfers data when needed on device

Executes on all platforms (or at least will "soon")

- including CPU, GPU, FPGA
- selectable at runtime (mostly)
- complex ecosystem

Intel wants to push into llvm main branch

- OneAPI to become an open specification

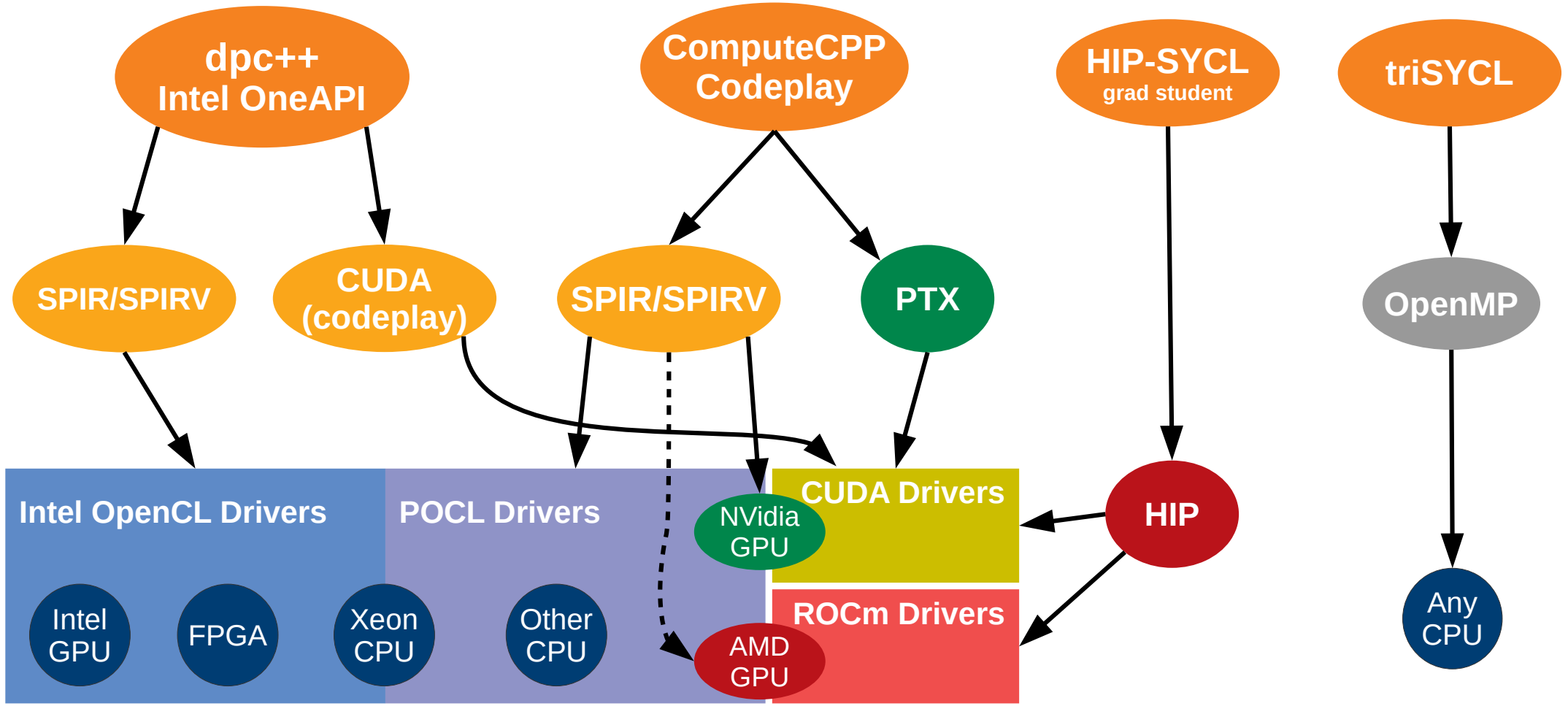
Long term OpenCL IR layer in question: Intel is replacing it with "LevelZero"

- OpenCL 1.2 standard is too limiting

Concurrent kernels don't work, for ANY backend (and won't for the foreseeable future)

- except CPU, where you can get 2 threads/core concurrently

SYCL Ecosystem



OpenCL 3.0

Newly released provisional specification by Khronos group

- Makes many of the "improvements" of the 2.X standard over 1.2 optional, enabling much broader range of devices to be 3.0 compliant, allowing use of extra features if they're available/applicable. "Less is More" philosophy.
- Allows more gradual adoption of extra features by vendors. Move away from "all or nothing" approach of 2.X
- OpenCL C++ abandoned in favor of clang based "C++ for OpenCL" (allows most C++17)

Strong-ish statements of support from various industry groups:

NVIDIA will ship a conformant OpenCL 3.0 when the specification is finalized

Compute product manager at NVIDIA

Codeplay is excited to enable hardware vendors to support OpenCL 3.0 and to take advantage of the flexibility provided

Founder and CEO of Codeplay Software

Intel strongly supports cross-architecture standards being driven across the compute ecosystem such as in OpenCL 3.0 and SYCL

Vice President, Intel Architecture, Graphics and Software

AMD?

Alpaka

Single source C++ kernels, standard limited by backend

Platform decided at compile time

CUDA like, multidimensional set of threads

- grid / block / thread / element

Maps abstraction model to desired acceleration back ends

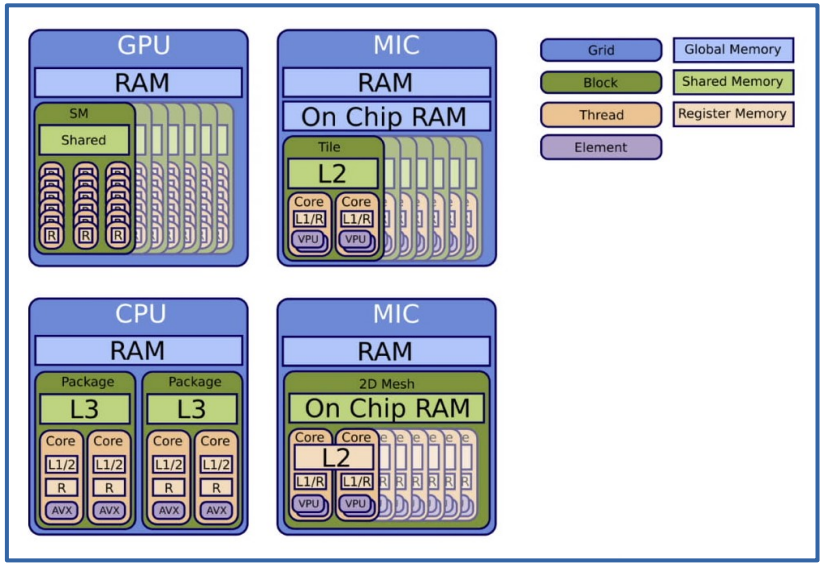
Data agnostic memory model:

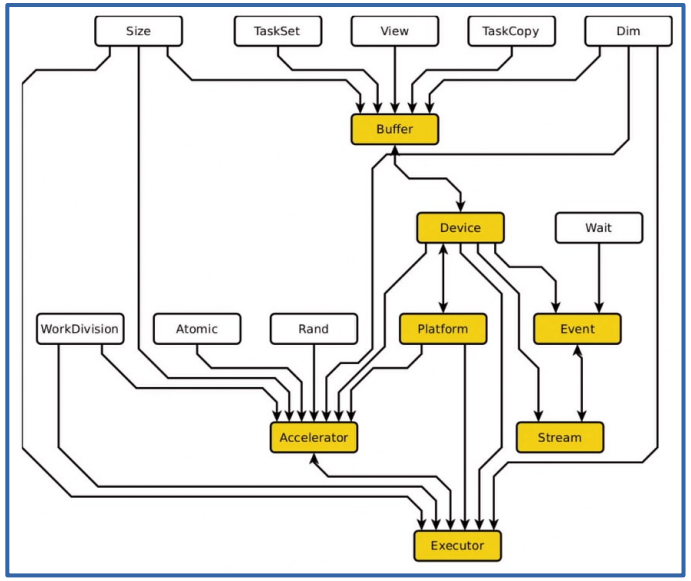
- allocates memory for you, but needs directives for hardware mapping
- same API for allocating on host and device

Uses templates for a "Zero Overhead Abstraction Layer"

Straightforward porting of CUDA kernels using cupla

- only include and kernel calls need to be changed





U.S. DEPARTMENT OF ENERGY | Office of Science
Argonne NATIONAL LABORATORY
BROOKHAVEN NATIONAL LABORATORY
Fermilab
BERKELEY LAB
Bringing Science Solutions to the World

OpenMP / OpenACC

Two similar mechanisms for annotating code to direct the compiler to offload bits of code to other devices

- uses `#pragmas`

OpenMP was really developed for MP on HPC

- very large and complex standard
- recently extended to target GPUs
- very prescriptive: need to tell compiler exactly how to unroll loops
  - have to modify pragmas when move to different GPU architecture

OpenACC developed explicitly for accelerators

- lets compiler make intelligent decision on how to decompose problems
- is a standard that describes what compilers ***should*** do, not *must*
  - different compilers interpret ***should*** very differently
- very strong support in Fortran community

Best supported on HPC