High Energy Physics
Center for Computational Excellence
Portable Parallelization Strategies

Charles Leggett, Meifeng Lin
for HEP-CCE

HEP-CCE Closeout All
Dec 18 2023

Software and Hardware Support Matrix in 2019

	CUDA	Kokkos	SYCL	HIP	OpenMP	alpaka	std::par
NVIDIA GPU			*codeplay*	*hipcc*			*nvc++*
AMD GPU			*hipSYCL*	*hipcc*			
Intel GPU			*oneAPI*				*oneAPI:dpl*
x86 CPU			*oneAPI*				*gcc*
FPGA							

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

Software and Hardware Support Matrix in 2023

	CUDA	Kokkos	SYCL	HIP	OpenMP	alpaka	std::par
NVIDIA GPU				*hipcc*	*nvc++ LLVM, Cray GCC, XL*		*nvc++*
AMD GPU			*openSYCL intel/llvm*	*hipcc*	*AOMP LLVM Cray*		
Intel GPU			*oneAPI intel/llvm*	CHIP-SPV: *early prototype*	*Intel OneAPI compiler*	*prototype*	*oneapi::dpl*
x86 CPU			*oneAPI intel/llvm openSYCL*	*via HIP-CPU Runtime*	*nvc++ LLVM, CCE, GCC, XL*		
FPGA				*via Xilinx Runtime*	*prototype compilers (OpenArc, Intel, etc.)*	*protytype via SYCL*	

HEP Testbeds

FastCaloSim

- ATLAS parameterized LAr calorimeter simulation
- 3 simple kernels (large workspace reset, main simulation, stream compaction)
- 1-D and 2-D jagged arrays
- small data transfer d->h at end of each event

Patatrack

- CMS pixel detector reconstruction
- 40 kernels of varying complexity and lengths (many are short)
  - good test for latency, concurrency, asynchronous execution, memory pools

Wirecell Toolkit

- LArTPC signal simulation
- 3 kernels: rasterization, scatter-add, FFT convolution

p2r

- CMS "propagate-to-R" track reconstruction in a single kernel

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

Testbed Completion Status

Ported representative testbeds from ATLAS, CMS and DUNE to each portability layer.

	Kokkos	SYCL	OpenMP	Alpaka	std::par
Patatrack	Done	Done*	WIP	Done*	Done compiler bugs
Wirecell	Done	Done	Done	no	Done
FastCaloSim	Done	Done	Done	Done	Done
P2R	done	Done	OpenACC	Done	Done

Evaluated each porting experience according to a number of different objective and subjective metrics.

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

Metrics

- Ease of Learning
- Code conversion
  - From CPU to GPU and between different APIs
- Extent of modifications to existing code
  - Control of main, threading/execution model
- Extent of modifications to the Data Model
- Extent of modifications to the build system
- Hardware Mapping
  - Current and promised future support of hardware
- Feature Availability
- Interoperability
  - Interaction with external libraries, thread pools, C++ standards
- Address needs of large and small workflows
- Long term sustainability and code stability
  - Backward/forward compatibility of API and e.g. CUDA
- Compilation time
- Run time/Performance
- Ease of Debugging
- Aesthetics

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

Metrics Evaluations

Metric	Kokkos	alpaka	SYCL	std::par/nvc++	OpenMP
Ease of Learning	Similar to C++ and CUDA, optimization more challenging	Very verbose API and sparse documentation make for steep learning curve	Similar to C++ and CUDA. Lots of documentation.	is C++	C++ with extra pragmas. Sparse documentation/examples for offload.
Serial Code Conversion	Similar to CUDA, though different syntax. Specialized optimizations not straightforward	Work needed to wrap kernels in callable objects. Many typedefs in API benefit from layer of template functions.	Similar to CUDA, though different syntax	very simple	Easy incremental porting mechanism from serial code (add pragmas, get offload working, performance tuning). More work to port from existing CUDA code.
Code Modification	Can be used in an existing complicated application without changes elsewhere. Kokkos runtime needs explicit init/fini. Can take and interpret command line arguments	similar to CUDA	Parallelize loops with command group handlers and parallel_for. If using USM, necessary to orchestrate kernel calls explicitly via waiting on events	Memory accessible on device must be allocated/freed in a file compiled by nvc++, and on the heap. This may require some copying of data.	Special memory allocation and transfer APIs. Can operate device and host parallel simultaneously.
Data Model Modification	Views can be used as a smart pointer to 1D data Crafting a SoA with Views tedious. Jagged arrays (Views of Views) not gracefully supported	Buffers used to wrap existing objects tedious to use. Alpaka managed memory buffers can lead to unexpected behavior	Buffers can be instantiated only from device copyable types. USM is most compatible with current EDM and custom data types.	May need to copy data to make it visible to USM	In general simpler than CUDA
Build System Integration	Can choose at most one backend for each execution space type. Choice must be done at the time of configuring the Kokkos runtime build.	Extensive configurability via CMake	Mostly seamless integration with CMake and make. Depending on the target platform/backend, additional Clang command line arguments are needed.	Need compiler wrapper to filter out options that CMake adds that break compiler. some bugs with gcc lib compatibility	Good integration with CMake and make.
Hardware Mapping	Supports multiple host-parallel backends, and NVIDIA, AMD, and Intel GPUs. Kokkos developers have been pro-active in supporting new hardware architectures as they emerge	Supports multiple host parallel backends, and NVIDIA and AMD GPUs	Can build and run now on any major vendor CPU and GPU. Third-party libraries can be called through interoperability. Backed by Intel, Codeplay, academic institutions and labs.	nvc++ supports CPU serial, CPU multicore, NVIDIA GPU. CPU multicore doesn't work reliably.	Supports multiple host-parallel backends, and NVIDIA, AMD, and Intel GPUs.

Metric	Kokkos	alpaka	SYCL	std::par/nvc++	OpenMP
Feature Availability	Concurrent kernel only with CUDA backend, and then requires CUDA specific features. Concurrent calls to Serial backend safe, but not efficient. Unsupported: sort function called from device code; common API to vendor-optimized FFT libraries; RNGs not following Gaussian or uniform distributions, such as binomial	Similar to CUDA	Support for reductions, kernel chaining, callbacks. Concurrent kernels not supported in practice. No native support of host-parallel	No low level controls of hardware or kernel launch parameters	Scan and memset ops not yet supported on GPUs by most compilers. Some CUDA atomic ops also not yet supported.
Long Term Sustainability	Good prospects now, but what happens if CUDA/SYCL get integrated into C++ standards? Is there still a need for Kokkos?	Possibly problematic - very small user and developer community	Don't foresee SYCL disappearing; essentially an OpenCL successor. Long term support for technologies by hardware vendors.	Part C++ standard	Very well supported by industry
Compilation Time	Each back-end/configuration needs its own binary. Does not take long to compile Kokkos libs, but kernel compilation slower than CUDA.	Similar to native CUDA/HIP	Similar to CUDA for small kernels. Can be prohibitively slow for very large ones	Much slower than gcc	Similar to CUDA
Ease of Debugging	Compiler error messages are difficult to read due to long template names	Compiler error messages are very long, and not easy to decipher	Intel VTune and Advisor are useful debugging and profiling tools. For third-party library calls, depends on the vendor (Nvidia is great, AMD is not).	Runtime errors mostly have same name, making identification difficult. Can do some debugging with cuda-gdb.	Clang provides runtime environment variables that help with debugging.
Interoperability	Kokkos and native (eg CUDA/SYCL/HIP) code can be mixed within one application. Can have issues with external libraries like TBB	Can be mixed with CUDA/HIP and other libraries in the same application	Good support for TBB and OpenMP	Good support for external libraries	Good support with external libraries and other portability layers

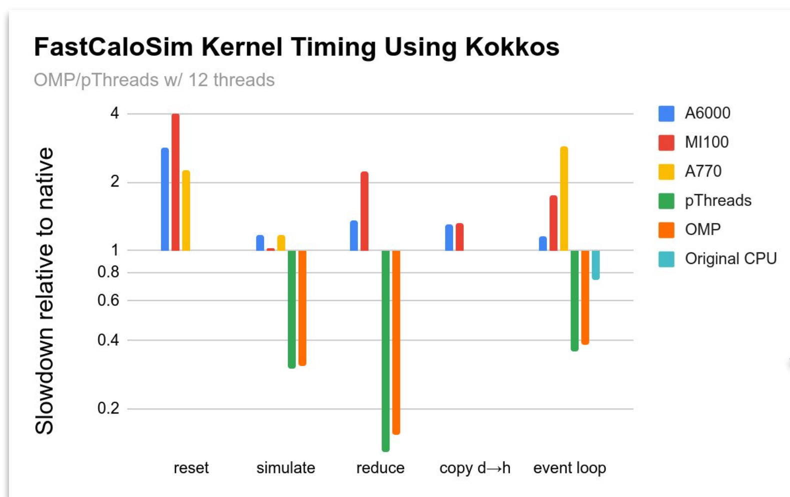
FastCaloSim Summary

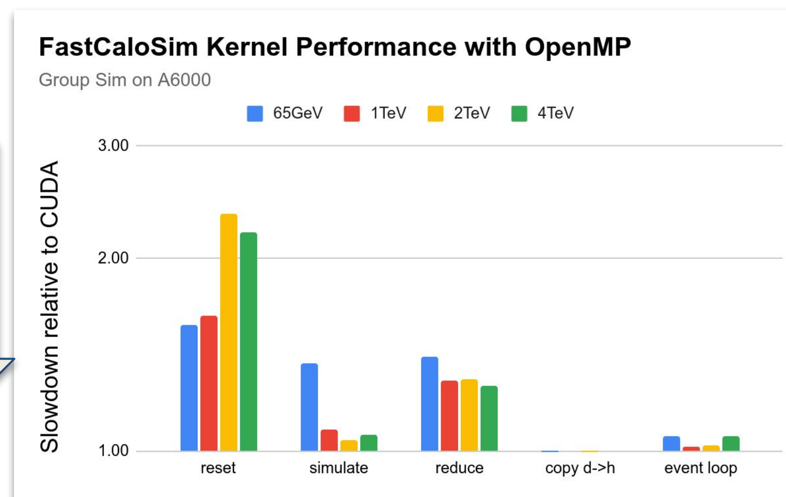
Moderate complexity enabled rapid exploration of all portability layers

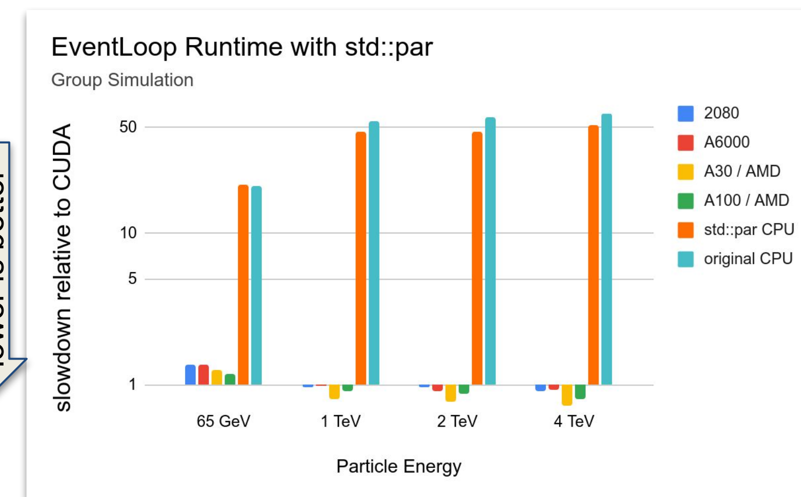
- hides weaknesses from more complex workflows

Good performance with all APIs (after tuning) for longer job configurations

- overheads from Kokkos initialization / kernel launch
- challenges from jagged arrays
- remarkable performance from std::par
  - very beneficial interactions with NVIDIA
- external library dependencies exposed issues with some API and build systems
- serial runtime with portability layers faster than original CPU implementation
- very good support from Kokkos and SYCL developers







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

Wirecell Toolkit

Three major steps of LArTPC simulation

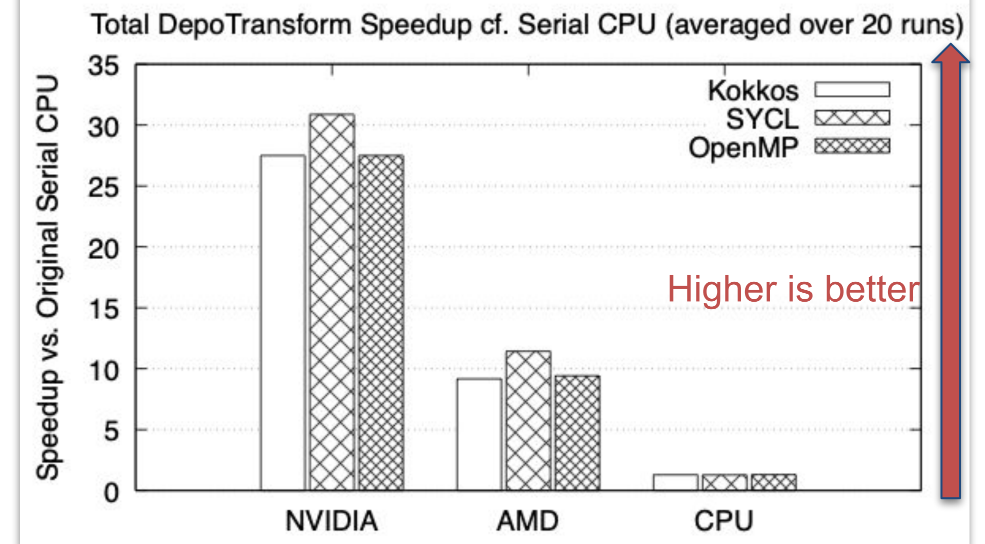
- **Rasterization:** depositions → patches (small 2D array, ~20×20)
  - # depo ~100k for cosmic ray event
- **Scatter adding:** patches → grid (large 2D array, ~10k×10k)
- **FFT:** convolution with detector response

Summary

- Restructured the code to expose more parallelism
- Wrappers to use optimized vendor libraries
- Ported to **CUDA (partial), Kokkos, SYCL, OpenMP** and **std::par** implementations
- Developed a stand-alone testing framework (without LArSoft dependence)
- Validated and benchmarked Kokkos, SYCL and OpenMP implementations; Achieved similar performance with different portability layers.

Experience and Lessons Learned:

- Finding the parallelizable kernels is key to achieve acceleration
- Once the first programming model was implemented (which helped solve many issues related to the dependencies, data structures, data layout etc.),. implementations with other programming models became quite straightforward.
- Different programming models have different pain points and require different tuning to achieve the best performance. In this test case, we were able to achieve similar performance with Kokkos, SYCL and OpenMP. More issues with stdpar.



Speedup in DepoTransform compared to original CPU on NVIDIA V100, AMD Radeon Pro VII, and AMD Ryzen 24-core CPU with Kokkos, SYCL and OpenMP

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

p2r Summary

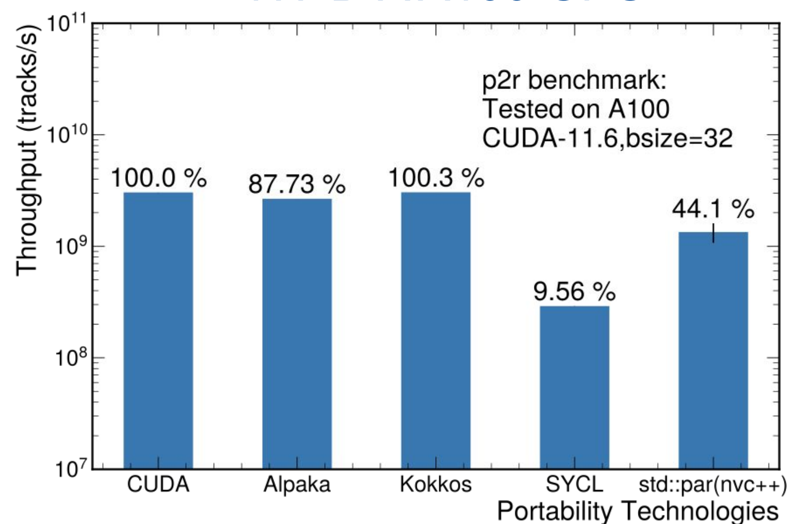
p2r is a relatively lightweight benchmark

- Performs core math of track reconstruction (track propagation and Kalman updates)
- Easy to port
- Easy to experiment with features/data-layout

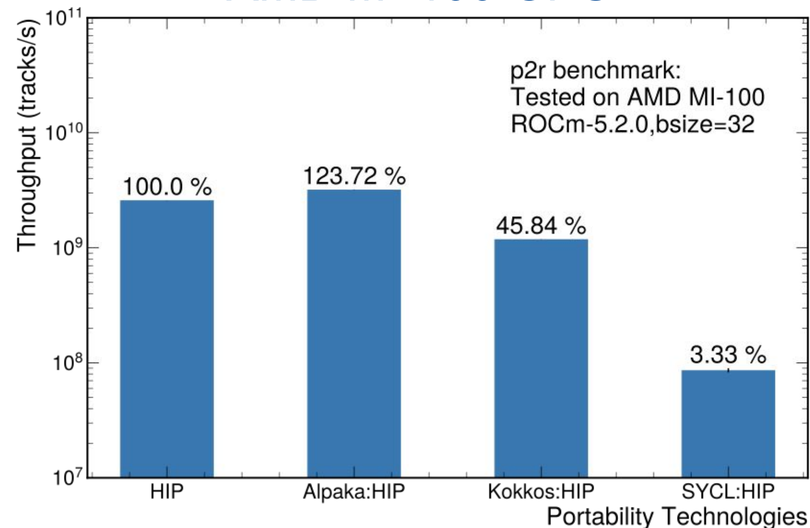
Lessons learned:

- Alpaka/Kokkos give close-to-native performance in NVIDIA/AMD GPUs and CPU
- SYCL/std::par performance are significantly behind despite sharing very similar implementation

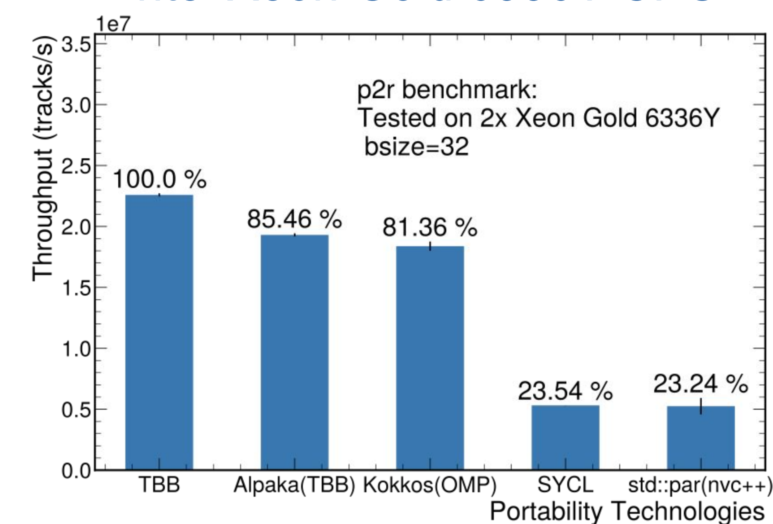
NVIDIA A100 GPU



AMD MI-100 GPU



Intel Xeon Gold 6336Y CPU



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

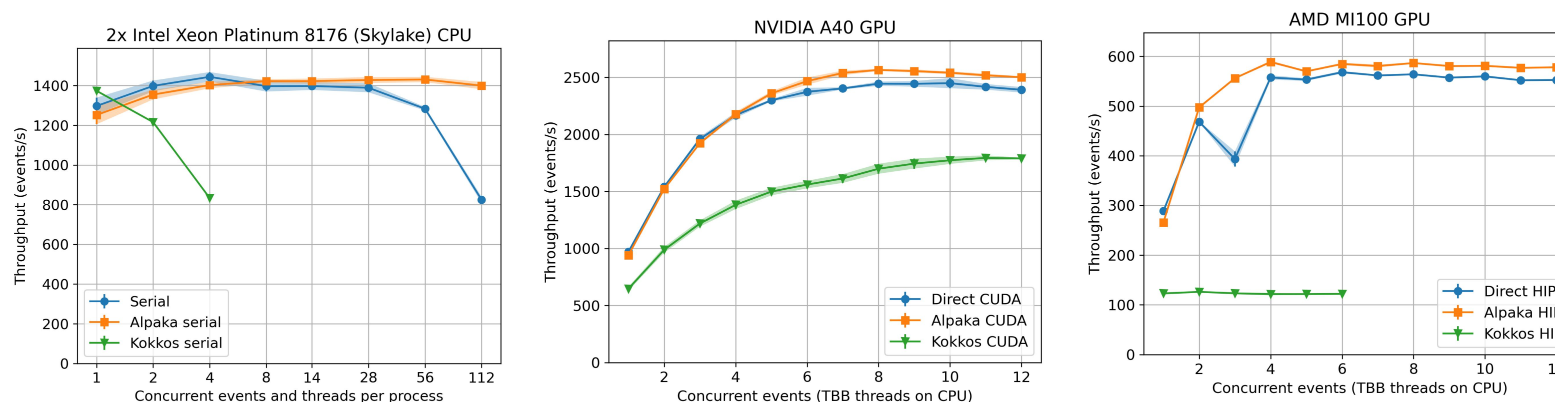
Patatrack Summary

Most complex use case: CMS pixel detector reconstruction from raw data to pixel tracks and vertices, multithreaded mock framework and build system

- closest approximation of integration in an experiment framework without actually doing it
- 40 kernels divided in 5 “framework modules” using rich set of CUDA features

Lessons learned

- Best performance on CPU, and NVIDIA and AMD GPUs with Alpaka
- Kokkos currently difficult to work with in a concurrent application, overheads
- SYCL (Intel oneAPI implementation): compilation problems, overheads
- std::par: compilation problems, crashes, leads to many more kernels (expect poor performance)
- OpenMP Target offload: compilation problems, data movement is concern



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

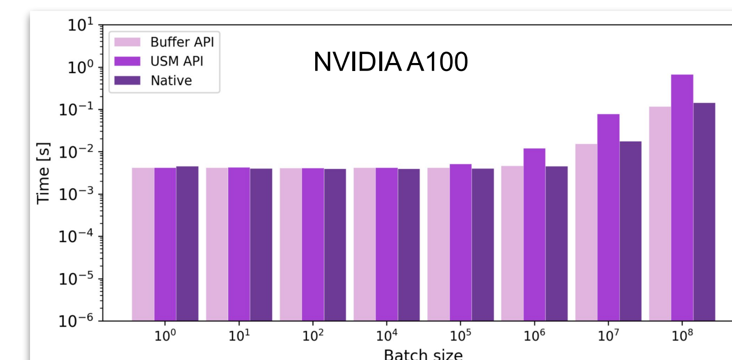
Random Number Generators

HEP-CCE

High quality random number generators are integral to HEP, especially essential to simulation.

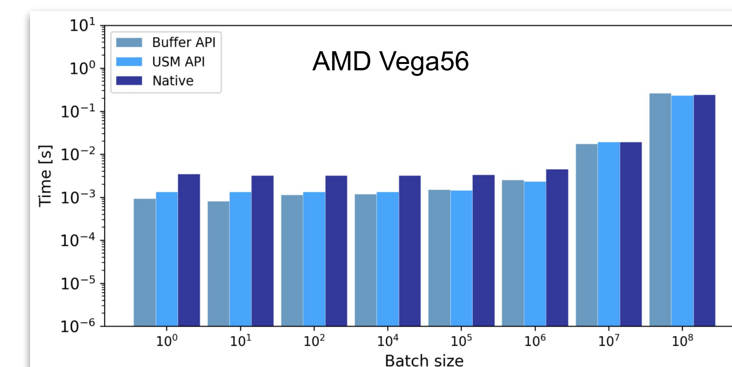
Developed extension to oneMKL to target AMD, NVIDIA and Intel GPU device specific RNG via SYCL using a uniform interface

- demonstrated near native performance
- integrated into Intel oneAPI libraries



Developed header-only wrapper that supports multiple random number distribution engines

- uniform, normal, Gaussian distributions
- cuRand, rocRand, random123
- NVIDIA, AMD GPUs, Intel GPUs WIP
- single and multithreaded CPU execution w/ OpenMP



U.S. DEPARTMENT OF ENERGY Office of Science · Argonne NATIONAL LABORATORY · Brookhaven National Laboratory · Fermilab · BERKELEY LAB Bringing Science Solutions to the World

Kokkos Evaluation

- Higher level of abstraction than CUDA
- Requires explicit initialization and finalization of runtime
- Separate compilation of library for different backends and features
  - implications for code distribution
- Can mix and match native and Kokkos kernels in same application

- Good performance for simple and long running kernels
  - hides overheads from initialization of data structures and kernel launches
- No native support for jagged arrays
- Concurrent kernels only with CUDA backend and CUDA specific features
  - concurrent calls to serial backend safe, but not efficient
- Poor interoperability with external concurrency mechanisms and thread pools (eg TBB)
- Built-in FFTs and RNGs, but no common API to vendor optimized libraries
- Most "portable" of all technologies - easiest to get app running on new platform
- Very good developer support

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

Alpaka Evaluation

- Verbose API, sparse documentation, steep learning curve
  - responsive and helpful developers
- Very small user community
- Need either to wrap kernels in callable objects or to use lambdas, heavy use of typedefs
  - compiler error messages hard to decipher
- For memory transfers between host and device one can use either alpaka buffers (takes ownership of the allocation) or alpaka views (to copy already allocated memory)
  - Our attempts to use alpaka views inside FastCaloSim led to random crashes
  - Patatrack didn't have these issues
- Extensive configurability with CMake
- Can mix in native kernels and libraries in same application
- Performance comparable to native

SYCL Evaluation

- Requires different compilers for different backends
  - oneAPI (Intel), llvm/sycl (Intel,NVIDIA,AMD), openSYCL (Intel,NVIDIA,AMD)
- Simplified code design as associates data dependencies with kernels for automatic data migration
  - can get better performance with explicit transfers
- No current support for concurrent kernels with any backend
  - though in theory supported by the standard
- Significant changes in SYCL language specification over time
- Good interoperability with external concurrency layers like TBB, OMP, MPI
- Mostly seamless integration with build systems
- Near native performance (*cf* CUDA / HIP)
  - except for p2r
- Strongly supported by Intel

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

OpenMP Target Offload Evaluation

- Widest difference of opinion in people's porting experience
  - some found it very easy to change serial code
  - some experienced large challenges
- Significant variation between performance characteristics of different compilers
  - compiler options
  - tuning of threads/teams
- Specialized operations unsupported (scan, memset) or much less performant (atomics) than CUDA
- Documentation is sparse, especially for more advanced features
- Debugging and profiling very challenging due to extra OpenMP code infrastructure and architecture-specific plugins loaded at initialization

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

std::par / nvc++ Evaluation

HEP-CCE

- Pure C++ - no learning curve (beyond using STL algorithms)
  - slightly convoluted indexed container access with C++17
  - automatic data migration triggered by page faults
- Not equivalent to CUDA/SYCL/Kokkos
  - no low level controls
  - not intended to be a CUDA replacement - rather a stepping stone to parallelism
- Support for NVIDIA GPUs (nvc++), Intel GPUs (oneAPI::dpl), and multicore CPUs (nvc++, g++)
- nvc++ is still immature
  - bugs (can't compile ROOT yet)
  - need workarounds to compile parts of projects w/ g++, parts with nvc++
  - any offloaded data must be allocated by code compiled with nvc++
  - compilation much slower than g++
  - unusual memory transfer speeds from AMD CPUs
- Very good performance for longer kernels that translate well into Thrust
- Path towards C++ standards based implementation of GPU offloading
  - C++26 std::exec (schedulers, senders, receivers), etc

Impacts on HEP

Final stage of CCE/PPS is reporting back to experiments

- Not yet there, but have made multiple interim reports
- ACAT, CHEP, HSF, IRIS-HEP

Already affected what HEP experiments are doing in both short and medium term

- CMS's choice of Alpaka
- ATLAS Run4 milestones
- DUNE signal processing kernels on GPUs
- Contributions to Snowmass21 process

Continue to interact with HEP experiments as we enter phase II of CCE

- Focussed meetings and workshops with stakeholder experiments
- Continued engagement with broader HEP community

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

Conclusions

There is no "one size fits all" solution. Different applications will have different optimal solutions.

- We have identified pain points with each portability layer
- Allows us to offer useful guidance to experiments depending on their use cases

Both software and hardware continue to rapidly evolve.

- Need to monitor GPU ecosystems and update recommendations as needed
- Emergence of C++ language level standards in the next 5 years may be a game changer

See increasing use of ML solutions for previously purely "algorithmic" tasks and vendor shifts to ML optimized hardware.

- Will expose a whole new set of issues

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

Last Steps

Need to complete paper that summarizes our work!

- https://www.overleaf.com/read/zjkvrgwxkfwp#b98623
- almost there - still a little left to do

Much of our work will continue in Phase II "PAW"

- But we will shift focus from portable applications to portable workflows
- Disseminate knowledge gained in Phase I to rest of the HEP community
- Use our testbeds, now "mini-apps" to test facilities

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

Cast of Characters

Current Members

- Mohammad Atif (BNL)
- Meghna Bhattacharya (FNAL)
- Mark Dewing (ANL)
- Zhihua Dong (BNL)
- Julien Esseiva (LBNL)
- Matti Kortelainen (FNAL)
- Martin Kwok (FNAL)
- Alexei Strelchenko (FNAL)
- Vakhtang Tsulaia (LBNL)
- Brett Viren (BNL)
- Tianle Wang (BNL)
- Haiwang Yu (BNL)

Past Members

- Vincent Pascuzzi (IBM)
- Kwangmin Yu (BNL)

Technical Leads

- Oliver Gutsche (FNAL), Charles Leggett (LBLN), Meifeng Lin (BNL)

fin

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