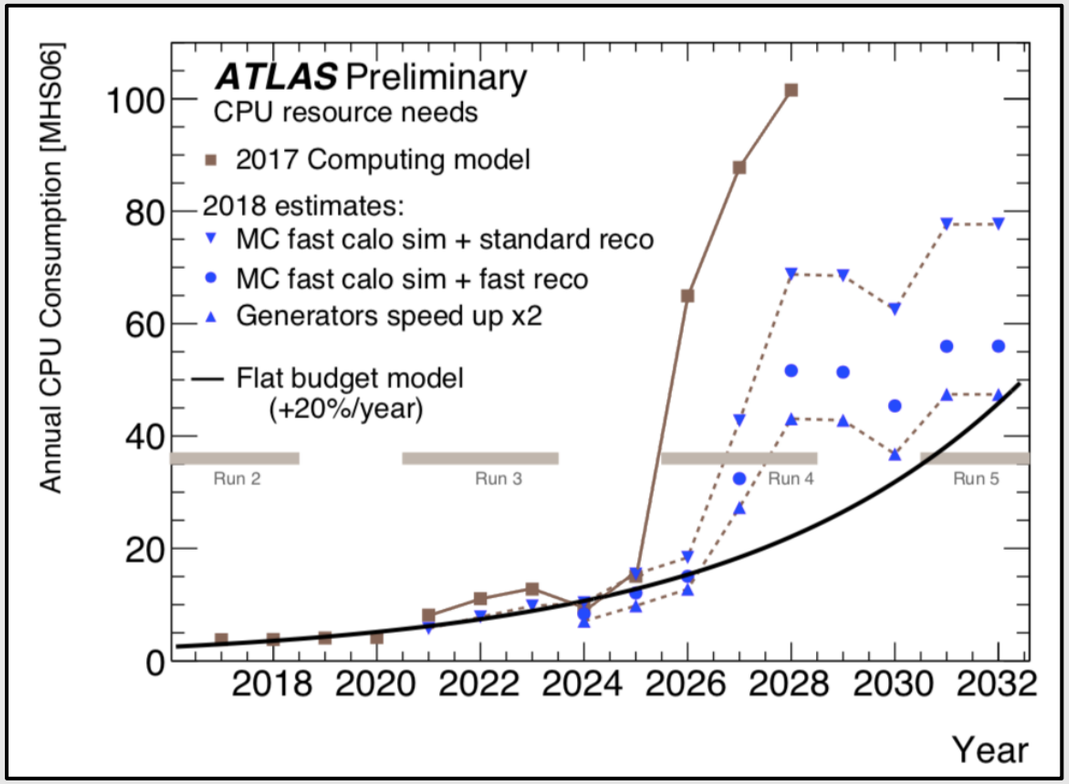
Porting Algorithms to Accelerators

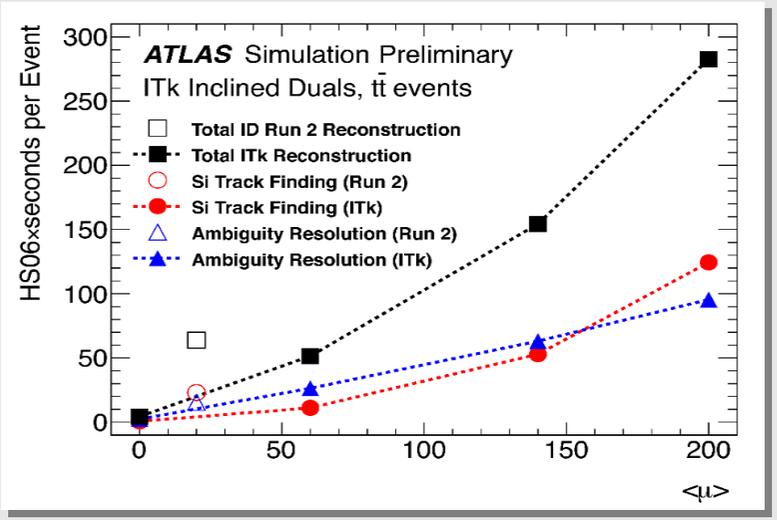
Charles Leggett

US ATLAS HPC Meeting
September 26 2019

The Challenge (Part I)

- We need enormously more compute power for HL-LHC than we did for Run I and II
  - We can survive Run III with the current environment
  - We can't survive in Run IV
- Lots of simulation, larger events due to much higher multiplicity





- Tracking combinatorics begin to dominate at high μ

The Challenge (Part II)

- In the next generation of supercomputers we see extensive use of accelerator technologies
  - Oak Ridge: **Summit** (2018)
    - 4608 IBM AC922 nodes *w/* 2x Power9 CPU
    - 3x NVIDIA Volta V100 + NVLink / CPU
  - LBL: NERSC-9 "**Perlmutter**" (2020)
    - AMD EPYC "Milan" x86 only nodes + mixed CPU / "next gen" NVidia GPU
  - Oak Ridge: **Frontier** (2021)
    - 1.5 exaflop
    - AMD EPYC CPU + 4x AMD "Instinct" GPU
  - Commercial clouds:
    - Brainwave / Azure FPGA
    - Google Cloud TPU
  - LLNL: **Sierra** (2018)
    - 4320 IBM AC922 nodes *w/* 2x Power9 CPU
    - 2x NVIDIA Volta V100 + NVLink / CPU
  - Argonne: **Aurora A21** (2021)
    - Intel Xeon CPU + Intel X^{e}/gen12 GPU + Optane
  - Tsukuba: **Cygnus** (2020)
    - 2x Intel Xeon 6162+ 4x NVidia V100 GPU
    - 2x CPU + 4x GPU + 2x Intel Stratix FPGA
  - Japan: **Fugaku** (2021)
    - manycore ARM A64fx (48+2)
    - integrated "SVE" 512 bit GPU-like accelerator
- In order to meet the HL-LHC computing requirements, we need to use all available computing resources, or cut back physics projections
  - **US funding agencies have indicated that we will not be able to get allocations if our code does not make use of accelerator hardware**

The Challenge (Part III)

- HEP algorithms are not easily parallelized or offloadable to an accelerator
  - very branchy
  - not very many wide loops or opportunities for SIMD style decomposition
  - EDM is C++ish. GPUs want arrays.
    - don't want to transform data back and forth every data is offloaded
  - lots and lots of Algorithms / Modules
    - death of a thousand cuts

- Most direct ports of HEP code to accelerators have been for the Online environment, where the hardware requirements are known in advance, and the system parameters are very well defined
  - you pick and choose your hardware, and code to match
  - you don't expect an upgrade cycle every 2 years

The Challenge (Part IV)

- We see all three flavours of ice cream in the next the DOE HPCs
  - **A21**: Intel, **Perlmutter**: NVidia , **Frontier**: AMD
- Each accelerator vendor has its own flavour of programming tools to target their GPU
  - NVidia: CUDA
  - Intel: SYCL / OneAPI / dpcpp
  - AMD: hip
- Our software needs to run for the next 10+ years
  - we cannot afford to re-write for each hardware platform
- And what comes out in 5 years?
  - we may see more “exotic” architectures (TPU, FPGA, ASIC)
- What happens if a vendor significantly modifies their programming environment?
  - eg AMD recently dropped SPIR support for hip/ROCm

Framework Integration

- Significant impedance mismatch:
  - V100 has 160,000 threads
  - Most of our loops are much less wide than that
    - gang data between events to increase GPU workload?
- It is likely that we will need to be able to schedule and execute concurrent kernels on the GPU
  - some support (eg CUDA streams), but not extensive and has significant performance drawbacks
  - currently significantly limits portability solutions
    - this will (is promised) to change in the coming year
- Depending on how vendors implement synchronous *vs* asynchronous kernel launching, may need to redesign parts of Athena / Gaudi framework
  - currently we've implemented a synchronous mechanism, on the assumption that we can re-task the CPU to do other work.
    - this has been poorly implemented by the hardware (NVidia, Intel)
  - asynchronous requires a callback (*á la* cmssw): incompatible with current Gaudi

Working Towards A Solution

- We need to find a portable solution that works for all accelerator platforms
  - portability is more important than performance

- Projects worth exploring
  - Kokkos
  - Raja
  - SYCL / OpenCL
  - OpenMP
  - OpenACC
  - ???

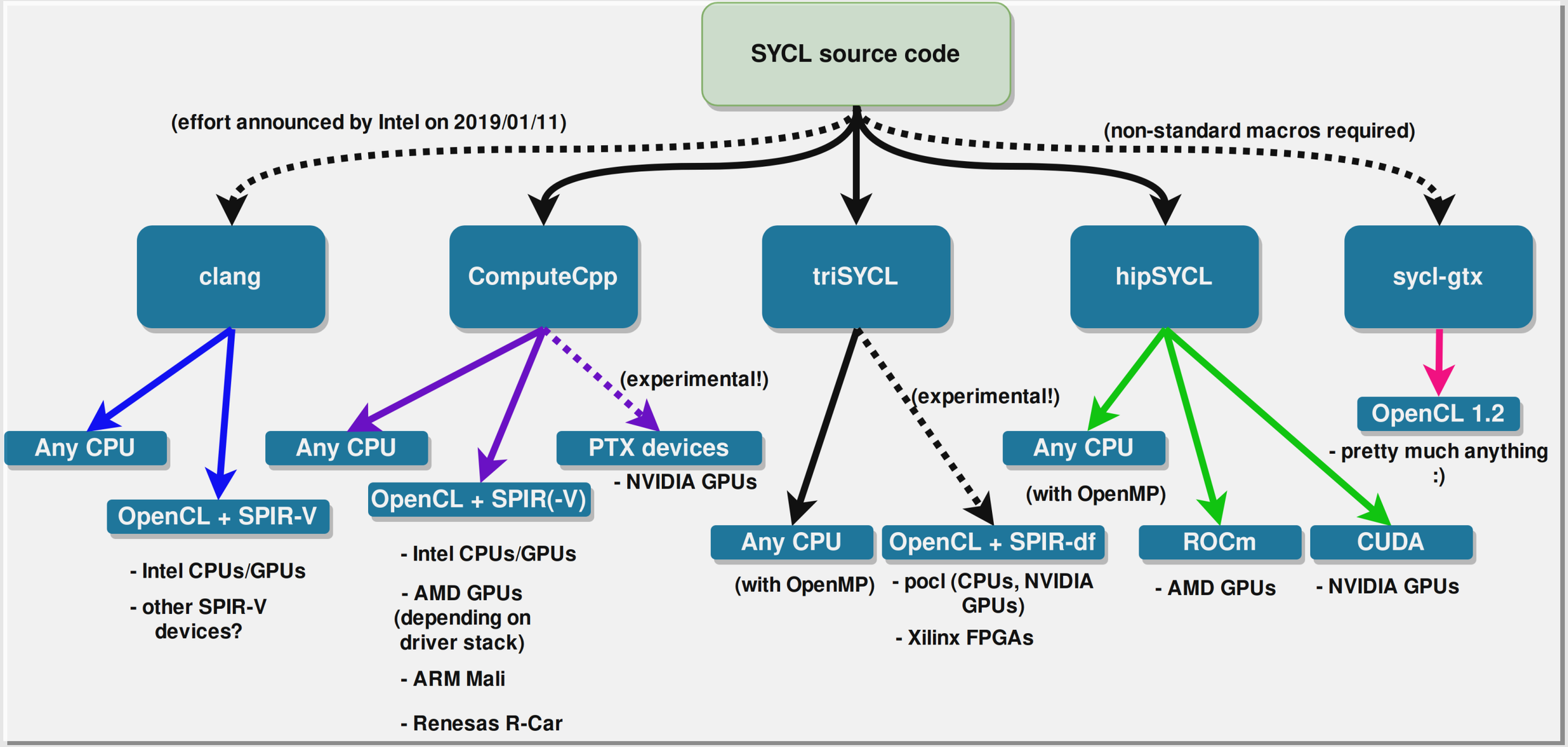
- Machine learning tools
  - pytorch, tensorflow, etc
  - requires major paradigm shift to use pervasively
  - does map much better onto accelerators
  - back-ends already there

- While some of these look good on paper, it is very important to understand how they map onto our workflows and framework
  - something which works well for the online may not map onto the offline
- We need to encourage ($$$) vendors to provide portability solutions
- We may need to develop these solutions ourselves if the vendors can't deliver

fin

(time for discussion)

SYCL Ecosystem



Kokkos

- Somewhat less flexible than SYCL
  - hard to explicitly dispatch kernels without use of a parallel_for-like construct
  - need to identify back-end at compilation time
  - no concurrent kernel execution at this time
    - beta version that explicitly uses cuda streams
- Has important features that aren't in SYCL
  - reduction construct
  - child tasks
  - more performant
- Support for Intel GPU and AMD not ready yet
  - sometime next year