Exascale Computing for Lattice QCD

C. DeTar

USQCD All Hands Meeting
Jefferson Lab
May 2, 2020

Hardware

- ALCF Aurora End of 2021
  - Intel/Cray
  - Node: 2 Intel Sapphire Rapids processors + 6 Xe (Ponte Vecchio) GPUs
  - Cray Slingshot Network
  - Programming model: Usual, Intel OneAPI, OpenMP, SyCL/DPC++
- ORNL Frontier End of 2021
  - AMD/Cray
  - Node: 1 AMD EPYC CPU + 4 AMD Radeon GPUs
  - Cray Slingshot Network
  - Programming model: Usual + AMD HIP

Performance goal

- Lattice QCD is one of 24 ECP applications
- FOM (figure of merit) Our benchmark suite must run 50 X faster on Aurora or Frontier than on Mira or Titan
- Benchmark suite has 3 x 2 = 6 components
  - (MILC + CPS + Chroma) X (configuration generation + analysis)
    - Measure time to complete a specific task. Average improvement factors
- Currently we are testing on Summit where our FOM stands at 7.5 X as of April 2019.
  - We will be measuring a new FOM in the next few weeks

LatticeQCD ECP

- Solver Task: Algorithms, multigrid (see Evan's talk)
- Critical Slowing Down: (see Chulwoo's talk)
- Contractions and Matrix Elements (see Robert's talk)
- Software (This overview. see Balint, Kate, and Peter's talks)

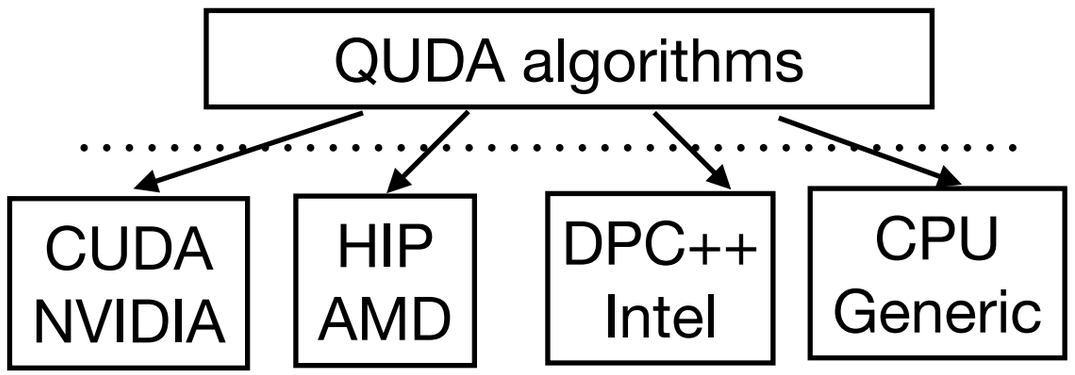
ECP Software Effort Organization

- Readiness for each of the three major code bases
  - Chroma (Jóo, Edwards, Winter)
  - CPS (Jung, Duo Guo, Yong Chull, Kelly)
  - MILC (CD, Gottlieb, Gelzer)
  - OpenMP, OpenACC, Kokkos Offloading (Meifeng Lin, Chapman, Kale)
  - Grid portability (Boyle, Filuci, Yamaguchi, CD, Vaquero)
  - QUDA portability (Kate, Howarth, Strelchenko, Osborn, Xiao-Yong Ji

Challenge

- Novel HPC architectures
  - Thus far, lattice QCD GPU experience has been almost exclusively with NVIDIA and highly optimized QUDA code.
  - Must port our codes to new hardware with new and still-developing programming tools:
    - OpenMP, SyCL, DPC++, HIP, etc
  - Performance unknown
- ECP Lattice QCD software strategy
  - Port QUDA and Grid for wide community use
  - Other effort
    - HotQCD (Steinbrecher -> Intel)
    - QDP++ (Jóo -> ORNL)
    - QEX (Osborn, Xiao-Yong Jin)
    - K-pi-pi code (Kelly)

QUDA strategy and status



- QUDA strategy (thanks Kate Clark et al!)
  - Currently depends on NVIDIA's CUDA
  - Define/Insert a back-end interface
  - Provide architecture-specific backend implementations
  - Interface consists of wrappers for CUDA calls plus kernel offloads plus reductions
- QUDA status
  - CUDA wrappers essentially done (Howarth)
  - Initial proposal for kernel offloading to various backends. Work on a CPU implementation (Osborn). (QUDA already has some built-in CPU support.)
  - Some experimentation with DPC++ port using Intel's conversion tool (Strelchenko)

Grid strategy and status

- Grid strategy
  - Grid was originally optimized for KNL — vector processor with SIMD lanes
  - Port to NVIDIA GPU via CUDA. Do this in a generic way for maximum flexibility.
  - Next step: port to DPC++/SyCL and HIP
- Grid status (see Peter's talk next)
  - NVIDIA GPU port completed last year
  - SyCL port waiting for more mature software stack (and some new features?)

Other effort

- QDP++ strategy and status (see Balint's talk)
  - Exploring JIT/LLVM and Kokkos
  - Kokkos will have both HIP and SyCL back ends
- HotQCD strategy and status (Steinbrecher)
  - Use OpenMP offload for Intel
  - Version works on Intel Gen9 — waiting for next generation Intel hardware