QUDA: QCD in CUDA for Multi-GPU Lattice Field Theory

Rich Brower (SciDAC software co-ordinator)

Stage 1: Basic Linear Solvers

QUDA (QCD in CUDA) library started in 2008 with NVIDIA's CUDA implementation by Kip Batson and Mike Clark at Boston University. It has expanded to a broad base of USQCD, SciDAC [1] software developers and is in wide use as the GPU backend for HEP and NP SciDAC application codes: Chroma, CPS, MILC, etc.

Provides:
- Various solvers for several discretizations, including multi-GPU support and domain-decomposed (Schwarz) preconditioners
- Additional performance-critical routines needed for physics applications

Maximize performance:
- Exploit physical symmetries
- Mixed-precision methods
- Averaging for high performance on all-CUDA-capable architectures
- Cache blocking

QCD on CUDA team – http://lattice.github.com/quda

- Ron Babich (NVIDIA)
- Rich Brower (Boston University)
- Michael Cheng (Boston University)
- Steve Gottlieb (Indiana University)
- Michael Clark (NVIDIA)
- Justin Foley (University of Utah)
- Guochun Shi (NCSA -> Google)
- Alexei Strelchenko (Cyprus Institute -> FNAL)
- Bálint Joó (Jlab)
- Claudio Rebbi (Boston University)
- Hyung-Jin Kim (BNL)
- Frank Winter (UoE -> Jlab)

Stage 2: Scaling to Multi-GPUs

ASDASD

- "QCD on CUDA" team – http://lattice.github.com/quda
- Ron Babich (NVIDIA)
- Kip Batson (LANL)
- Rich Brower (Boston University)
- Michael Cheng (Boston University)
- Mike Clark (NVIDIA)
- Justin Foley (University of Utah)
- Joel Gostick (Indiana University)
- Steve Gottlieb (Indiana University)
- Guochun Shi (NCSA -> Google)
- Alexei Strelchenko (Cyprus Institute -> FNAL)
- Bálint Joó (Jlab)
- Claudio Rebbi (Boston University)
- Hyung-Jin Kim (BNL)
- Frank Winter (UoE -> Jlab)

QUDA (QCD in CUDA) library started in 2008 with NVIDIA's CUDA implementation by Kip Barros and Mike Clark at Boston University. It has expanded to a broad base of USQCD, SciDAC [1] software developers and is in wide use as the GPU backend for HEP and NP SciDAC application codes: Chroma, CPS, MILC, etc.

Provides:
- Various solvers for several discretizations, including multi-GPU support and domain-decomposed (Schwarz) preconditioners
- Additional performance-critical routines needed for physics applications

Maximize performance:
- Exploit physical symmetries
- Mixed-precision methods
- Averaging for high performance on all-CUDA-capable architectures
- Cache blocking

Stage 3: Multi-scale Physics

Physics Aware Algorithm: Adaptive Multigrid

Block Jacobi Domain Decomposition [2]

References