BREAKING THE LATENCY BARRIER
STRONG SCALING LQCD ON GPUS
Kate Clark, Mathias Wagner, Evan Weinberg
QUDA

• “QCD on CUDA” - http://lattice.github.com/quda (open source, BSD license)
• Effort started at Boston University in 2008, now in wide use as the GPU backend for BQCD, Chroma, CPS, MILC, TIFR, etc.
• Provides:
  - Various solvers for all major fermionic discretizations, with multi-GPU support
  - Additional performance-critical routines needed for gauge-field generation
• Maximize performance
  • Exploit physical symmetries to minimize memory traffic
  • Mixed-precision methods
  • Autotuning for high performance on all CUDA-capable architectures
  • Domain-decomposed (Schwarz) preconditioners for strong scaling
  • Eigenvector and deflated solvers (Lanczos, EigCG, GMRES-DR)
  • Multi-source solvers
  • Multigrid solvers for optimal convergence
• A research tool for how to reach the exascale
QUDA CONTRIBUTORS
10+ years - lots of contributors

Ron Babich (NVIDIA)
Simone Bacchio (Cyprus)
Michael Baldhauf (Regensburg)
Kip Barros (LANL)
Rich Brower (Boston University)
Nuno Cardoso (NCSA)
Kate Clark (NVIDIA)
Michael Cheng (Boston University)
Carleton DeTar (Utah University)
Justin Foley (Utah -> NIH)
Joel Giedt (Rensselaer Polytechnic Institute)
Arjun Gambhir (William and Mary)
Steve Gottlieb (Indiana University)
Kyriakos Hadjiyiannakou (Cyprus)

Dean Howarth (BU)
Bálint Joó (Jlab)
Hyung-Jin Kim (BNL -> Samsung)
Bartek Kostrzewa (Bonn)
Claudio Rebbi (Boston University)
Hauke Sandmeyer (Bielefeld)
Guochun Shi (NCSA -> Google)
Mario Schröck (INFN)
Alexei Strelchenko (FNAL)
Jiqun Tu (Columbia)
Alejandro Vaquero (Utah University)
Mathias Wagner (NVIDIA)
Evan Weinberg (NVIDIA)
Frank Winter (Jlab)
MULTI GPU BUILDING BLOCKS

Halo packing Kernel

Interior Kernel

Halo communication

Halo update Kernel
SOME TERMINOLGY

P2P, CUDA aware, GPU Direct RDMA

P2P: direct exchange between GPUs
SOME TERMINOLGY

P2P, CUDA aware, GPU Direct RDMA

P2P: direct exchange between GPUs
**SOME TERMINOLGY**

**P2P**, CUDA aware, GPU Direct RDMA

**P2P**: direct exchange between GPUs

**CUDA aware MPI**: MPI can take GPU pointer
can do batched, overlapping transfers (high bandwidth)
SOME TERMINOLGY

P2P, CUDA aware, GPU Direct RDMA

P2P: direct exchange between GPUs

CUDA aware MPI: MPI can take GPU pointer
can do batched, overlapping transfers (high bandwidth)
SOME TERMINOLGY

P2P, CUDA aware, GPU Direct RDMA

**P2P:** direct exchange between GPUs

**CUDA aware MPI:** MPI can take GPU pointer
- can do batched, overlapping transfers
  (high bandwidth)
SOME TERMINOLGY
P2P, CUDA aware, GPU Direct RDMA

P2P: direct exchange between GPUs

CUDA aware MPI: MPI can take GPU pointer
can do batched, overlapping transfers (high bandwidth)

GPU Direct RDMA: (implies CUDA aware MPI)
RDMA transfer from/to GPU memory here NIC <-> GPU
SOME TERMINOLOGY
P2P, CUDA aware, GPU Direct RDMA

**P2P**: direct exchange between GPUs

**CUDA aware MPI**: MPI can take GPU pointer, can do batched, overlapping transfers (high bandwidth)

**GPU Direct RDMA**: (implies CUDA aware MPI) RDMA transfer from/to GPU memory here NIC <-> GPU
STRONG SCALING
STRONG SCALING

Multiple meanings
- Same problem size, more nodes, more GPUs
- Same problem, next generation GPUs
- Multigrid - strong scaling within the same run (not discussed here)

To tame strong scaling we have to understand the limiters
- Bandwidth limiters
- Latency limiters
SINGLE GPU PERFORMANCE

Wilson Dslash

Tesla V100
CUDA 10.1
GCC 7.3

GFLOPS

Lattice length

1312 GB/s
1291 GB/s
1180 GB/s

“strong scaling”
SINGLE GPU PERFORMANCE

Wilson Dslash

Look at scaling of Dslash on one DGX-1: half precision with $16^4$ local volume

Tesla V100
CUDA 10.1
GCC 7.3
WHAT IS LIMITING STRONG SCALING
classical host staging

DGX-1,16⁴ local volume, half precision, 1x2x2x2 partitioning
API OVERHEADS

CPU overheads and synchronization are expensive

DGX-1,16⁴ local volume, half precision, 1x2x2x2 partitioning
USING NVLINK AND FUSING KERNELS
fewer copies with higher bandwidth, fewer kernels, less API overhead

DGX-1,16\* local volume, half precision, 1x2x2x2 partitioning
REMOTE WRITE

Packing kernel writes to remote GPU using CUDA IPC

DGX-1, 16\(^4\) local volume, half precision, 1x2x2x2 partitioning
NVSHEMEM
NVSHMEM
GPU centric communication

Implementation of OpenSHMEM, a Partitioned Global Address Space (PGAS) library

NVSHMEM features
- Symmetric memory allocations in device memory
- Communication API calls on CPU (standard and stream-ordered)
- Allows kernel-side communication (API and LD/ST) between GPUs
- NVLink and PCIe support (intranode), InfiniBand support (internode)
- X86 and Power9 support
- Interoperability with MPI and OpenSHMEM libraries

Early access (EA2) available - please reach out to nvshmem@nvidia.com
Keep general structure of packing, interior and exterior Dslash

Use `nvshmem_ptr` for intra-node remote writes (fine-grained)
  Packing buffer is located on remote device
Use `nvshmem_putmem_nbi` to write to remote GPU over IB (1 RDMA transfer)

Need to make sure writes are visible: `nvshmem_barrier_all_on_stream`
  or barrier kernel that only waits for writes from neighbors

**Disclaimer:**
Results from an first implementation in QUDA with a pre-release version of NVSHMEM
NVSHMEM DSLASH

DGX-1,16⁴ local volume, half precision, 1x2x2x2 partitioning
**NVSHMEM + FUSING KERNELS**

no extra packing and barrier kernels needed

---

**DGX-1,16**

local volume, half precision, 1x2x2x2 partitioning
LATENCY OPTIMIZATIONS

Different strategies implemented

DGX-1,16⁴ local volume, Wilson, half precision, 1x2x2x2 partitioning
DGX-2: FULL NON-BLOCKING BANDWIDTH

2.4 TB/s bisection bandwidth
DGX-2 STRONG SCALING
Global Volume $32^4$, Wilson-Dslash
DGX-2 STRONG SCALING
Global Volume $32^4$, Wilson-Dslash

<table>
<thead>
<tr>
<th>#GPUs</th>
<th>MPI double</th>
<th>SHMEM double</th>
<th>MPI single</th>
<th>SHMEM single</th>
<th>MPI half</th>
<th>SHMEM half</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
DGX-2 STRONG SCALING
Global Volume $32^4$, Wilson-Dslash

- MPI double
- MPI single
- MPI half
- SHMEM double
- SHMEM single
- SHMEM half

GFlop/s vs. #GPUs
DGX-2 STRONG SCALING
Global Volume $32^4$, HISQ-Dslash

- MPI double
- SHMEM double
- MPI single
- SHMEM single
- MPI half
- SHMEM half

GFlop/s vs. #GPUs
DGX-2 STRONG SCALING
Global Volume $32^4$, HISQ-Dslash
DGX-2 STRONG SCALING
Global Volume $32^4$, HISQ-Dslash
NVSHMEM OUTLOOK

intra-kernel synchronization and communication
NVSHMEM OUTLOOK
intra-kernel synchronization and communication

One kernel to rule them all!
Communication is handled in the kernel and latencies are hidden.
MULTI-NODE SCALING
SCALING TESTBEDS

PROMETHEUS (NVIDIA)
- NVIDIA DGX-1
  - 8 V100 (16GB)
  - 4 EDR IB
  - 50/100 GB/s NVLink P2P*

DGX SuperPOD (NVIDIA)
- Top500 #22
- NVIDIA DGX-2H
  - 16 Tesla V100 (32GB)
  - 8 EDR IB
  - 300 GB/s NVLink P2P*

SUMMIT (OLCF)
- Top500 #1
- IBM AC 922
  - 6 Tesla V100 (16GB)
  - 2 EDR IB
  - 100 GB/s NVLink P2P*

*) bi-directional bandwidth between two GPUs
MULTI-NODE STRONG SCALING
Prometheus, Wilson, 64³×128 global volume
MULTI-NODE STRONG SCALING
Prometheus, Wilson, 64³x128 global volume

GFlop/s vs. #GPUs for different interconnects:
- single
- single GDR
- single NVSHMEM
MULTI-NODE STRONG SCALING
Prometheus, Wilson, $64^3 \times 128$ global volume

![Graph showing GFlop/s vs. #GPUs for single, single GDR, and single NVSHMEM modes.](image)
MULTI-NODE STRONG SCALING

Prometheus, Wilson, $64^3 \times 128$ global volume

GFlop/s vs. #GPUs for different memory types and precision levels:
- Single
- Half
- Double
- Single GDR
- Half GDR
- Double GDR
- Single NVSHMEM
- Half NVSHMEM
- Double NVSHMEM
MULTI-NODE STRONG SCALING
Prometheus, Wilson, 64³x128 global volume
MULTI-NODE STRONG SCALING
half precision, Wilson, $64^3 \times 128$ global volume
MULTI-NODE STRONG SCALING
half precision, Wilson, $64^3 \times 128$ global volume

![Graph showing GFlop/s vs. #GPUs for different systems: Prometheus MPI, Prometheus NVSHMEM, DGX SuperPOD MPI, DGX SuperPOD NVSHMEM, SUMMIT MPI, SUMMIT NVSHMEM. The graph plots performance metrics across increasing number of GPUs.]
MULTI-NODE STRONG SCALING
half precision, Wilson, $64^3 \times 128$ global volume

*) NOTE: SUMMIT results have been obtained using $64^3 \times 96$ global volume and scaled by $8/6$ (x and y-axis) to make up for 6 GPUs/node. At a given number of GPUs the local volume per GPU is identical for all systems.
STRONG SCALING LATTICE QCD …

Optimize for latency and bandwidth
Autotune kernel performance and comms policy

**GPU centric communication with NVSHMEM**
- reduce number of kernels through fusion
- reduce API overheads and synchronizations
- simplify code (fewer lines)

Significantly improved strong scaling …
... but still early results (stay tuned)

generalize for all Dirac operators
including Multigrid (coarse operators)
ANNOUNCING QUDA 1.0

The high-level, C++-based framework for solving lattices in QCD and beyond is finally here. QUDA 1.0 brings a lot of benefits:

- Old Dslash kernel code became increasingly limiting and bug fixes
- Various other new routines and algorithms, code cleanup
- Complete re-write of all operators in a supercomputers in the world

LQCD ideal for GPU deployment
LQCD simulations are typically memory-bandwidth bound, algorithms performantly and at scale with relative ease

Enables testing new fermion discretizations and new exascale. With its new high-level C++11 framework, including Chroma, MILC, CPS, BQCD, TIFR and tmLQCD

Rewrite brings a lot of benefits:
- Ability to run on CPU: Future framework for all architectures??
- Larger local volumes
- Extensibility, composability and maintainability:
  - Alternative compilation targets (e.g., C++17 pSTL)
  - C++ Interface
  - Continuous optimization of multi-grid algorithms
  - Eigenvector compression
- Improved scaling: NVSHMEM
- Alternative compilation targets (e.g., C++17 pSTL)
- C++ Interface
- Continuous optimization of multi-grid algorithms
- Eigenvector compression
- Improved scaling: NVSHMEM

Copyright © 2019 NVIDIA Corporation

Authors: Kate Clark, Mathias Wagner, Evan Weinberg, NVIDIA

QUDA: A LIBRARY FOR QCD AND BEYOND

https://devblogs.nvidia.com/parallelforall/inside-volta/
https://devblogs.nvidia.com/parallelforall/inside-tesla/

QUDA VOLTA V100

SUMMIT AND SIERRA ARE NOW LIVE
- Summit + Sierra 
  - Summit: 4,920 nodes, 126,000 V100s with 300 GB/s NVLink
  - Sierra: 4,320 nodes, 4,320 Telsa V100s with 240 GB/s NVLink

DEFLECTED ADAPTIVE MG AT THE PHYSICAL POINT
- Deflating CG steps to solve for operator eigenvalues

CHROMA HMC-MG ON SUMMIT
- Multi-GPU HISQ Dslash

QUDA NODE PERFORMANCE OVER TIME
- NVIDIA DGX-2
- NVIDIA DGX-1

RAFFLE

POSTER SESSION

Tuesday, 17:50
MULTI-NODE STRONG SCALING

double precision, Wilson, $64^3 \times 128$ global volume

*) NOTE: SUMMIT results have been obtained using $64^3 \times 96$ global volume and scaled by 8/6 (x and y-axis) to make up for 6 GPUs/node. At a given number of GPUs the local volume per GPU is identical for all systems.
### MULTI-NODE STRONG SCALING

**single precision, Wilson, 64^3x128 global volume**

<table>
<thead>
<tr>
<th>GFlop/s</th>
<th>#GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>50,000</td>
<td>16</td>
</tr>
<tr>
<td>100,000</td>
<td>32</td>
</tr>
<tr>
<td>150,000</td>
<td>64</td>
</tr>
<tr>
<td>200,000</td>
<td>128</td>
</tr>
<tr>
<td>250,000</td>
<td>256</td>
</tr>
<tr>
<td>275,000</td>
<td>512</td>
</tr>
<tr>
<td>300,000</td>
<td>1024</td>
</tr>
</tbody>
</table>

- Prometheus MPI
- Prometheus NVSHMEM
- DGX SuperPOD MPI
- DGX SuperPOD NVSHMEM
- SUMMIT* MPI
- SUMMIT* NVSHMEM

*) **NOTE:** SUMMIT results have been obtained using 64^3x96 global volume and scaled by 8/6 (x and y-axis) to make up for 6 GPUs/node. At a given number of GPUs the local volume per GPU is identical for all systems.