GPUS FOR LATTICE FIELD THEORY

Kate Clark, NVIDIA
AGENDA

Introduction and QUDA Retrospective

Recent Progress with Multigrid (on GPUs)

Strong scaling

The Community is Active

Some thoughts on the future
WHAT IS A GPU?

• Tesla V100 - Volta architecture (2017)
  – Massively threaded - 5120 processing cores
  – 7.5 FP64 / 15 FP32 / 125 FP16 Gflops peak
• Deep memory hierarchy
  – As we move away from registers
    • Bandwidth decreases
    • Latency increases
• Inverse memory hierarchy
  – 40 MiB register file (up to 255 registers / thread)
  – 10 MiB 128 KiB L1 / shared memory
  – 6 MiB coherent L2 cache
• Programmed using a diversity of approaches
  – CUDA C++ / Fortran / Python
  – OpenACC / OpenMP directives
  – Future: C++17 pSTL / Fortran 2018 DO CONCURRENT
NVIDIA POWERS WORLD'S FASTEST SUPERCOMPUTER

Summit Becomes First System To Scale The 100 Petaflops Milestone

122 PF
HPC

3 EF
AI

27,648
Volta Tensor Core GPUs
IBM AC 922

4/6 V100 GPUs
NVLink to GPU and P9
2 EDR IB
DGX-2
1 node supercomputer

2 PFLOPS | 512GB HBM2 | 10 kW | 350 lbs
DGX-2: FULL NON-BLOCKING BANDWIDTH

2.4 TB/s bisection bandwidth
SOME GPU CLUSTERS

Summit @ OLCF
#1 on TOP500
4600 IBM P9 nodes
6x Tesla V100 / node
100 GB/s NVLink
2x EDR IB / node

Piz Daint @ CSCS
#6 on TOP500
5272 Cray X50 nodes
1x Tesla P100 / node
Cray Aries

DGX SuperPOD @ NVIDIA
#22 on TOP500
96 DGX-2 nodes
16x Tesla V100 / node
300 GB/s NVLink P2P
8x EDR IB / node

NSC3 @ CCNU
18 AGX-2 nodes
8 Tesla V100 / node
100 GB/s NVLink P2P
GPU COMPUTING FOR LQCD, CIRCA 2009

Wilson Operator on a GPU

- Wilson-Operator (GTX 280)
  - Single: 129 Gflops (mat-vec), 110 Gflops (inverter)
  - Double: 39 Gflops (mat-vec), 32 Gflops (inverter)
  - Half: 205 Gflops (mat-vec), 160 Gflops (inverter)
- Wilson-Clover (+5-10% performance)
  - Single: 140 Gflops (mat-vec)
- Algorithms
  - Multi-precision inverter using Reliable Updates (Sleijpen/Van der Vorst)

GPUs for LQCD were bleeding edge

~100 GFLOPs per GPU

Single GPU only

Mixed-precision Krylov solvers were the state of the art

plenary Beijing 2009
QUDA

• “QCD on CUDA” - [http://lattice.github.com/quda](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 1.0

ANNOUNCING QUDA 1.0

Complete rewrite in modern C++
11 years in the making
See our poster for more details

Still many problems to solve in Lattice Field Theory...

Always wanting to build upon our collaboration with the community...come and join the fun

This week we announced QUDA 1.0
QUDA CONTRIBUTORS
10+ years - lots of contributors

Ron Babich (NVIDIA)
Simone Bacchio (Cyprus)
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)
Eloy Romero (William and Mary)
Hauke Sandmeyer (Bielefeld)
Guochun Shi (NCSA -> Google)
Mario Schröck (INFN)
Alexei Strelchenko (FNAL)
Jiqun Tu (Columbia)
Alejandro Vaquero (Utah University)
Mathias Wagner (NVIDIA)
André Walker-Loud
Evan Weinberg (NVIDIA)
Frank Winter (Jlab)
RAW GPU PERFORMANCE

QUDA 0.1 on Tesla

QUDA 1.0 on Volta
RAW GPU PERFORMANCE OVER TIME

15x in 10 years
(Moore’s law would have been >32x)
Speedup determined by measured time to solution for solving the Wilson operator against a random source on a V=24^364 lattice, $\beta=5.5$, $M_\pi = 416$ MeV. One node is defined to be 3 GPUs.
QUDA NODE PERFORMANCE OVER TIME
Multiplicative speedup through software and hardware

Speedup determined by measured time to solution for solving the Wilson operator against a random source on a $V=24^364$ lattice, $\beta=5.5$, $M_\pi = 416$ MeV. One node is defined to be 3 GPUs.
RECENT PROGRESS WITH MULTIGRID SOLVERS
WHY MULTIGRID?

- Optimality
- Speed
- Stability

Babich et al. 2010

Clark et al. (2016)

Osborn et al. 2010
From Titan running 2016 code to Summit running 2019 code we see >82x speedup in HMC throughput

Multiplicative speedup coming from machine and algorithm

Highly optimized multigrid for gauge field evolution

Mixed precision an important piece of the puzzle
  • double – outer defect correction
  • single – GCR solver
  • half – preconditioner
  • int32 – deterministic parallel coarsening
DEFLATED MG AT THE PHYSICAL POINT

KC, Dean Howarth and Evan Weinberg

Multigrid shifts the lowest eigenvalues to the coarse grids

Some Dirac operators (e.g., staggered and twisted mass) end up with pathological coarse-grid spectrum

For Twisted-clover operator, solution has been to add a fictitious heavy twist to the coarse operator to improve its condition number at the cost of decreased multigrid efficiency [Alexandrou et al]

Instead we deflate the coarse grid operator recovering optimal MG convergence and a 3x speedup over “mu scaling”
Time to solution is measured running QUDA on 4x DGX-1V nodes (32 GPUs) for solving the Twisted-mass + clover operator against a random source on a $64^3 \times 128$ lattice, $\beta = 1.778$, $\kappa = 0.139427$, $\mu = 0.000720$, solver tolerance $10^{-7}$.
STAGGERED MULTIGRID
Rich Brower, KC, Dean Howarth, Alexei Strelchenko, Evan Weinberg (Tuesday 15:20)

Naïve Galerkin projection does not work
Spurious low modes on coarse grids
System gets worse conditioned as we progressively coarsen

$16^2, \beta = 6.0, m = -0.07$

Compare to Wilson MG which preserves low modes with no cascade
STAGGERED MULTIGRID
Rich Brower, KC, Dean Howarth, Alexei Strelchenko, Evan Weinberg (Tuesday 15:20)

Transform into Kahler-Dirac form (arXiv: 0509026 Dürr) through unitary transformation

\[
\begin{pmatrix}
  m & 0 & -\frac{1}{2} U_x(2\bar{n}) & -\frac{1}{2} U_y(2\bar{n}) \\
  0 & m & -\frac{1}{2} U_y(2\bar{n} + \bar{\chi}) & \frac{1}{2} U_y(2\bar{n} + \bar{\gamma}) \\
  \frac{1}{2} U_x^*(2\bar{n}) & \frac{1}{2} U_y(2\bar{n} + \bar{\chi}) & m & 0 \\
  \frac{1}{2} U_y^*(2\bar{n}) & -\frac{1}{2} U_x(2\bar{n} + \bar{\gamma}) & 0 & m
\end{pmatrix}
\]

“Precondition” the staggered operator by the Kahler-Dirac block

No spurious low modes as we coarsen

![Graph](image-url)
HISQ

Level 2: 48 dof per site.
Solver: GCR, tolerance 0.25, max 16 iterations
Operator: Left-block Schur, 16-bit precision

Level 1: 3 dof per site.
Solver: GCR, tolerance $10^{-10}$
Smooother: CA-GCR(0,8)

KD preconditioned system “pseudo fine”

First real coarse grid

Level 2: 48 dof per site.
Solver: GCR, tolerance 0.25, max 16 iterations
Operator: Left-block Schur, 16-bit precision

Smooother: CA-GCR(0,2)

Second real coarse grid

Level 3: 128 dof per site. Solver: GCR, tolerance 0.25, max 16 iterations
Operator: Left-block Schur, 16-bit precision

Smooother: CA-GCR(0,2)

Level 4: 192 dof per site.
Solver: CA-GCR(16)
Operator: Left-block Schur, 16-bit precision

Level 5: 1024 vector SVD Deflation
Time to solution is measured running QUDA on 72x Summit nodes (432x GPUs) for solving the HISQ operator against a random source on a $96^3 \times 192$ lattice, $\beta = 6.72$, $a = 0.06$, $m_l = 0.0008$, $m_s = 0.022$, solver tolerance $10^{-10}$
FERMION SOLVERS

Combination of algorithm (multigrid) and machine (GPUs)

A single Volta can solve at 1 second per Wilson solve with local volume of \( V=32^3 \times 64 \) per GPU

A single node (DGX-2) can solve \( V=64^3 \times 128 \) at one second per solve

16 nodes of DGX-2 can solve \( V=128^3 \times 256 \) at one second per solve

Fermion solvers are not the challenge they used to be (caveats unbound)

Next frontier for: combining multi-rhs with multigrid
STRONG SCALING
MILC APEX LARGE BENCHMARK

Strong scaling HISQ RHMC \( V = 72^3 \times 144 \)
STRONG SCALING

Multiple meanings
- Same problem size, more nodes, more GPUs
- Same problem, next generation GPUs
- Multigrid - strong scaling within the same run
- We (the LQCD community) care about all of the above

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

Wilson Dslash

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

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

“strong scaling”

Tesla V100
CUDA 10.1
GCC 7.3
MULTI GPU BUILDING BLOCKS

- Halo packing Kernel
- Interior Kernel
- Halo communication
- Halo update Kernel
WHAT IS LIMITING STRONG SCALING

classical host staging

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-1,16⁴ local volume, half precision, 1x2x2x2 partitioning
NVSHMEM
GPU-centric communication

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

NVSHMEM features
- 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

NVSHMEM has been developed as an NVIDIA internal co-design with QUDA

Early access (EA2) available - please reach out to nvshmem@nvidia.com
### NVSHMEM + FUSING KERNELS

No extra packing and barrier kernels needed.

**36 µs**

**DGX-1, 16\(^4\) local volume, half precision, 1x2x2x2 partitioning**
DGX-2 STRONG SCALING
Global Volume $32^4$, Wilson-Dslash

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

GFlop/s vs. #GPUs

#GPUs

0
1
2
4
8
16

20,000
16,000
12,000
8,000
4,000
0
MULTI-NODE STRONG SCALING

DGX SuperPOD, Wilson, $64^3 \times 128$ global volume

- half GDR
- single GDR
- double GDR
- half NVSHMEM
- single NVSHMEM
- double NVSHMEM

GFlop/s vs. #GPUs
NVSHMEM IS NOT JUST FOR QUDA

BMW collaboration (Szabolcs Borzani)
THE COMMUNITY IS ACTIVE
LATTICE QCD WITH TENSOR CORES
KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

Follow up work from arXiv:1804.08593 (Guo, Mawhinney and Tu)

Multi-splitting Preconditioned CG
- Motivated by lack of network bandwidth in supercomputing centers (e.g., Summit)
- Block Jacobi preconditioner for (M)DWF that correctly applies the Dirichlet boundary condition for the e/o normal op
- Significantly reduces outer iterations
- Local preconditioner becomes prohibitively expensive
A = \left[ 1 - \kappa_b M_\phi^\dagger D_w^\dagger M_{5^-}^\dagger M_\phi^\dagger D_w^\dagger M_{5^-}^\dagger \right] \left[ 1 - \kappa_b M_{5^-}^{-1} D_w M_\phi M_{5^-}^{-1} D_w M_\phi \right]
LATTICE QCD WITH TENSOR CORES
KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

before 1st hopping term

\[ A = [1 - \kappa_b M_\phi ^\dagger D_w M_5^{\dagger -1} M_\phi ^\dagger D_w M_5^{\dagger -1}] [1 - \kappa_b M_5^{\dagger -1} D_w M_\phi M_5^{\dagger -1} D_w M_\phi] \]
LATTICE QCD WITH TENSOR CORES

KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

after 1st hopping term

\[ A = \left[ 1 - \kappa_b M_\phi^\dagger D_w^\dagger M_5^{-\dagger} M_\phi^\dagger D_w^\dagger M_5^{-\dagger} \right]\left[ 1 - \kappa_b M_5^{-1} D_w M_\phi M_5^{-1} D_w M_\phi \right] \]
LATTICE QCD WITH TENSOR CORES
KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

before 2ed hopping term

\[ A = \left[ 1 - \kappa_b M_{\phi}^\dagger D_w^\dagger M_5^{-\dagger} M_{\phi}^\dagger D_w^\dagger M_5^{-\dagger} \right] \left[ 1 - \kappa_b M_5^{-1} D_w M_{\phi} M_5^{-1} D_w M_{\phi} \right] \]
LATTICE QCD WITH TENSOR CORES
KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

after 2ed hopping term

\[
A = \left[ 1 - \kappa_b M_{\phi}^{\dagger} D_w M_5^{-\dagger} M_{\phi}^{\dagger} D_w M_5^{-\dagger} \right] \left[ 1 - \kappa_b M_5^{-1} D_w M_{\phi} M_5^{-1} D_w M_{\phi} \right]
\]
$$A = \left[ 1 - \kappa_b M_\phi^\dagger D_w M_5^{-\dagger} M_\phi^\dagger D_w M_5^{-\dagger} \right] \left[ 1 - \kappa_b M_5^{-1} D_w M_\phi M_5^{-1} D_w M_\phi \right]$$
LATTICE QCD WITH TENSOR CORES
KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

before 4th hopping term

\[ A = \left[ 1 - \kappa_b M^\dagger_\phi D^\dagger_w M^{-\dagger}_5 \right] \left[ 1 - \kappa_b M^{-1}_5 D_w M_\phi M^{-1}_5 D_w M_\phi \right] \]
LATTICE QCD WITH TENSOR CORES
KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

Modern GPU have high throughput tensor-core functionality

\[ D = \begin{pmatrix} A_{0,0} & A_{0,1} & \ldots & A_{0,15} \\ A_{1,0} & A_{1,1} & \ldots & A_{1,15} \\ \vdots & \vdots & \ddots & \vdots \\ A_{15,0} & A_{15,1} & \ldots & A_{15,15} \end{pmatrix} + \begin{pmatrix} B_{0,0} & B_{0,1} & \ldots & B_{0,15} \\ B_{1,0} & B_{1,1} & \ldots & B_{1,15} \\ \vdots & \vdots & \ddots & \vdots \\ B_{15,0} & B_{15,1} & \ldots & B_{15,15} \end{pmatrix} \]

FP16 or FP32  \hspace{1cm} \text{FP16}  \hspace{1cm} \text{FP16 or FP32}

Tesla V100
FP64: 7.5 TFLOPS
FP32: 15 TFLOPS
Tensor: 125 TFLOPS

\[
\left( 1 - \kappa_b^2 M_\phi^\dagger D_w^\dagger M_5^{-\dagger} M_\phi^\dagger D_w^\dagger M_5^{-\dagger} \right) \left( 1 - \kappa_b^2 M_5^{-1} D_w M_\phi M_5^{-1} D_w M_\phi \right)
\]

fuse  \hspace{1cm} fuse  \hspace{1cm} fuse  \hspace{1cm} fuse
LATTICE QCD WITH TENSOR CORES
KC, Chulwoo Jung, and Robert Mawhinney, Jiqun Tu

Increasingly beneficial as one strong scales

<table>
<thead>
<tr>
<th>nodes</th>
<th>local volume</th>
<th>solver</th>
<th>inner iter</th>
<th>(outer) iter</th>
<th>r. u.</th>
<th>performance/node</th>
<th>time</th>
<th>speed up</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>16 - 24 - 12 - 24</td>
<td>CG</td>
<td>–</td>
<td>42133</td>
<td>471</td>
<td>4.66</td>
<td>486.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSPCG</td>
<td>05</td>
<td>16903</td>
<td>195</td>
<td>1.56(01)/5.45(35)/37.29(53)</td>
<td>456.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSPCG</td>
<td>06</td>
<td>14860</td>
<td>173</td>
<td>1.56(01)/5.51(31)/37.60(58)</td>
<td>412.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSPCG</td>
<td>07</td>
<td>13787</td>
<td>161</td>
<td>1.56(01)/5.48(26)/37.49(60)</td>
<td>460.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSPCG</td>
<td>08</td>
<td>12922</td>
<td>151</td>
<td>1.56(01)/5.44(26)/37.55(63)</td>
<td>469.5</td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>16 - 12 - 12 - 24</td>
<td>CG</td>
<td>–</td>
<td>42427</td>
<td>474</td>
<td>3.85</td>
<td>296.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSPCG</td>
<td>05</td>
<td>17625</td>
<td>203</td>
<td>1.26(01)/4.54(37)/36.21(52)</td>
<td>271.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSPCG</td>
<td>06</td>
<td>15425</td>
<td>179</td>
<td>1.27(01)/4.55(33)/36.26(57)</td>
<td>262.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSPCG</td>
<td>07</td>
<td>14409</td>
<td>168</td>
<td>1.26(01)/4.57(30)/36.39(60)</td>
<td>268.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSPCG</td>
<td>08</td>
<td>13597</td>
<td>159</td>
<td>1.27(01)/4.53(28)/36.35(63)</td>
<td>276.0</td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>16 - 12 - 12 - 12</td>
<td>CG</td>
<td>–</td>
<td>42482</td>
<td>474</td>
<td>2.93</td>
<td>195.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSPCG</td>
<td>05</td>
<td>18250</td>
<td>210</td>
<td>1.00(01)/3.68(34)/34.62(45)</td>
<td>183.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSPCG</td>
<td>06</td>
<td>15959</td>
<td>185</td>
<td>1.01(01)/3.68(35)/34.79(54)</td>
<td>159.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSPCG</td>
<td>07</td>
<td>14985</td>
<td>174</td>
<td>1.01(01)/3.68(32)/35.06(58)</td>
<td>163.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSPCG</td>
<td>08</td>
<td>14287</td>
<td>167</td>
<td>1.00(01)/3.69(20)/34.76(61)</td>
<td>169.1</td>
<td></td>
</tr>
</tbody>
</table>

6144 GPUs (Summit)

matrix multiplication and linear algebra
preconditioning inversion

1.4x speed up

preconditioned CG

NVIDIA
The Neutron Lifetime on Sierra Early Science

In 2.5 weekends on Sierra - we accomplished 5x more than in 1 year on Titan

Machine-to-machine, compared to Titan @ ORNL (Titan is 18,688 nodes 16-core AMD + 1 K20X/node)

For our research
Sierra (~4300) is ~10 times faster than Titan
Summit (~4600) is ~15 times faster than Titan

These machines are disruptively faster than previous computers


Simulating the weak death of the neutron in a femtoscale universe with near-Exascale computing
Berkowitz, Clark, Gambhir, McElvain, Nicholson, Rinaldi, Vranas, Walker-Loud, Chang, Joó, Kurth, Orginos
First-class C++11 application with broad coverage for LQCD (and beyond) workflow

Embrace modern programming paradigms

Develop branch: excellent multicore CPU performance
- Contains functionality equivalent to QMP, QIO, QDP++, and much of Chroma

- SU(N), U(1), multiple fermion representations
- Gauge evolution: HMC, RHMC, EOFA
- Wilson, Clover, Domain wall, Mobius, + various 5D overlap
  - Improved staggered valence action

- Multiple solvers
  - Mixed precision Krylov solvers (CG, GMRES, BiCGstab ...)
  - Block solvers
  - Polynomial preconditioned Lanczos & deflation
  - Experimental multigrid

- Split Grids for valence solves - avoid communication
  - NB: Not the same as trivial parallelism:
    - Eigenvectors held globally, deflate many RHS
    - Solves are local. 3x gain on Cori
Aim to target multiple accelerators (Nvidia, AMD, Intel Aurora system)

Presently compile all C++ files through NVCC (no .cu files)
  - Minimise vendor specific code to a few macros
  - Study HIP, OpenMP 5.0, SyCL options for other targets
  - Assume host and device have a unified memory space (UVM, SVM)

(recent) Single source high performance kernels:
  - same kernel gives good performance on GPU and CPU

All Fermion operators now accelerated
  - Wilson, Wilson5D, Improved Staggered
  - 5D Mobius terms also

Entire CG runs on GPU. Full (R)HMC next target.

Intra node GPU-GPU communication uses NVlink; multi node under test
The key operation in many Lattice QCD simulations is the inversion of the fermion matrix. It requires a 4-dimensional stencil which calculates the product of a vector $v$ by a sparse matrix known as the Dslash operator and stores the result on a discretized 4-dimensional derivative.

The HISQ action suppresses unphysical interactions of quarks by smoothing each link with a weighted sum of neighboring links referred to as smearing. It is mainly used in the force calculation for the Hybrid Monte Carlo algorithm and can take up to 40% of the total runtime.

**HISQ Link Smearing**

The Lepage Smearing Kernel: $D_{\text{slash}}$ Kernel: stems from a discretized 4-dimensional derivative.

For many Lattice QCD applications, a large number of fermion matrix inversions are performed on a single gauge field. In order to exploit reuse of these gauge fields, we can apply the Dslash operation for multiple right-hand sides (rhs) at once. Increasing the number of rhs from one to four more than doubles the arithmetic intensity (Flop/byte) of the Dslash operation.

**Multi-GPU Dslash Performance**

  - architecture independent code
  - high and low level interface written in C++
  - support for multiple Lattice actions

- KNL, SKL and HSW use HotQCD code
- no overlap of communications and computation
- no Naik reconstruct
- single precision
- high and low level interface written in C++

**Solution Performance Problems**

- Grid Dslash achieves optimal performance
- Grid Lepage smearing performance is limited by short-comeings of parallel transport algorithm
- one parallel region for each link multiply in a staple,
- too many synchronization points
- unnecessary loads and stores of intermediate results

---

**Grid FOR HISQ**

Patrick Steinbrecher (Poster)

**Single-GPU Dslash Performance Comparison**

- Volta V100, Grid
- Knights Landing
- 2x Skylake
- 2x Haswell
- fp32, single node
- KNL, SKL and HSW use HotQCD code

**HISQ Dslash**


---

**GRAND FOR HISQ**

Performance of Staggered Fermion Kernels using Grid

Patrick Steinbrecher, Swagato Mukherjee

Brookhaven National Laboratory - USA

Express LOQCD using the Nim language

High-level expressions transformed into different target
CUDA
OpenCL
OpenMP

In production by the LSD collaboration
First Study of Nf=2+1+1 Lattice QCD with Physical Domain-Wall Quarks
Ting-Wai Chiu (Thursday, Hadron Spectroscopy, 14:40)

- Quarks: optimal DWF with $N_s = 16$, $\lambda_{\text{max}} / \lambda_{\text{min}} = 6.20 / 0.05$.
- Gluons: plaquette gauge action at $\beta = 6/g^2 = 6.20$.
- Lattice size: $64^3 \times 64$, $a \approx 0.062$ fm, $M_\pi L \approx 3$, $L \approx 4$ fm.
- For the one-flavor, use the Exact One-Flavor pseudofermion Action (EOFA).
- For the 2-flavor, use the two-flavor algorithm for DWF.
- Thermalization: 1 unit of DGX-1 ($\approx 8$ months), TWC, arXiv: 1811.08095
- Production runs: 4 DGX-1 ($\approx 2$ months) $\rightarrow$ 25 DGX-1 ($\approx 2$ months).
  Generated $\approx 1250$ trajectories $\rightarrow$ $\approx 250$ configurations.
- Topological charge is measured, and $\chi_{\text{top}}$ is determined.
- Mass spectra of $\pi^\pm, K^\pm, D^\pm, \bar{c}s, \bar{c}c, \Omega, \Omega_c, \Omega_{cc}, \Omega_{ccc}$

OPTIMAL DWF
Ting-Wai Chiu
Thursday 14:40

Moderate dense-node GPU clusters well suited for HMC
NUMERICAL STOCHASTIC PERTURBATION

Kenichi Ishikawa (Wednesday 9:20)

High-order numerical stochastic perturbation theory of the twisted Eguchi-Kawai model (Batched GEMMs and FFTs)

4x Pascal P100

Dual socket SkyLake Gold

Fig. 2  Performance comparison from CPU and GPU versions ($N = 225$). Left: CUDA(FFT-CONV) version, Right: CPU(FFT-CONV) version.
MACHINE LEARNING QCD ON GPUS
Albergo, Kanwar, Shanahan (Thursday, A&M 15:00)

Proof of concept study using machine-learned generative flow model to propose configurations in a Markov chain (Metropolis-Hastings) (arXiv:1904.12072)

Removal of critical slowing down in $\phi^4$ theory

Work in progress to extend to gauge theories

Trained the neural network using an 80x GPU cluster running PyTorch
SOME THOUGHTS ON THE FUTURE
Next Generation

Perlmutter (2020)

More flops
* more parallelism

More memory
* bandwidth
* capacity

4x NVIDIA "Volta-next" GPU
- > 7 TF
- > 32 GiB, HBM-2
- NVLINK

1x AMD CPU
4 Slingshot connections
- 4x25 GB/s

GPU direct, Unified Virtual Memory (UVM)

2-3x Cori
THE PROTRACTED DEATH OF MOORE’S LAW
BEYOND THE EXASCALE

Cannot weak scale to infinite volume \[ C \sim m^{-1} a^{-6} V^{9/8} \]

Even LQCD could be running out of parallelism
   Expose all sources parallelism

Pose everything as a matrix or batch problem where possible, e.g.,
   Block solvers
   Block contractions
**SOURCES OF PARALLELISM**

1. **Grid parallelism**
   Volume of threads

2. **Link matrix-vector partitioning**
   \[ 2 \ N_{\text{vec}} \text{-way parallelism} \]
   \((\text{spin} \ast \text{color})\)

3. **Stencil direction**
   8-way parallelism

4. **Dot-product partitioning**
   4-way parallelism

\[
\begin{pmatrix}
  a_{00} & a_{01} & a_{02} & a_{03} \\
  a_{10} & a_{11} & a_{12} & a_{13} \\
  a_{20} & a_{21} & a_{22} & a_{23} \\
  a_{30} & a_{31} & a_{32} & a_{33}
\end{pmatrix}
\begin{pmatrix}
  b_0 \\
  b_1 \\
  b_2 \\
  b_3
\end{pmatrix}
= \begin{pmatrix}
  a_{00} & a_{01} \\
  a_{10} & a_{11} \\
  a_{20} & a_{21} \\
  a_{30} & a_{31}
\end{pmatrix}
\begin{pmatrix}
  b_0 \\
  b_1
\end{pmatrix}
+ \begin{pmatrix}
  a_{02} & a_{03} \\
  a_{12} & a_{13} \\
  a_{22} & a_{23} \\
  a_{32} & a_{33}
\end{pmatrix}
\begin{pmatrix}
  b_2 \\
  b_3
\end{pmatrix}
\]
MULTIPLE RIGHT-HAND SIDES

48$^3$x12, HISQ, single precision, one code

- Volta (2017)
- Pascal (2016)
- Maxwell (2014)
- Kepler (2012)
- Fermi (2010)

Graph showing performance in GFlop/s over # rhs from 1 to 16 with Volta showing a 3.5x increase and Fermi showing a 2.5x increase.
### Reworking the LAPH Pipeline

**KC, Ben Hoerz, Dean Howarth, Colin Morningstar, André Walker-Loud, et al**

2 nucleon (2 baryon) and 2 hadron (ππ, Kπ) and meson-baryon scattering cross sections

<table>
<thead>
<tr>
<th></th>
<th>Classical approach</th>
<th>Parallelism / Intensity</th>
<th>Modern approach</th>
<th>Parallelism / Intensity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3-d Laplace eigenvectors</strong></td>
<td>Lanczos</td>
<td>T x V₃</td>
<td>Batched-Block-Lanczos</td>
<td>B x T x V₃ / AI ~ B</td>
</tr>
<tr>
<td><strong>Clover-fermion solves</strong></td>
<td>Sequential multigrid</td>
<td>V₄</td>
<td>Block multigrid</td>
<td>Nₓ x V₄ / AI ~ N₉hs</td>
</tr>
<tr>
<td><strong>Sink projections</strong></td>
<td>Sequential inner products</td>
<td>T x V₃ / AI ~ 1</td>
<td>Blocked inner productions =&gt; Matrix multiply</td>
<td>Nₓ x Nₓ x T x V₃ / AI ~ (Nₓ x Nₓ) / (N₁ + Nₓ)</td>
</tr>
<tr>
<td><strong>Current Insertions</strong></td>
<td>Sequential insertions (morally inner products)</td>
<td>T x V₃ / AI ~ 1</td>
<td>Blocked insertions =&gt; Matrix multiply</td>
<td>Nₓ² x T x V₃ / AI ~ (Nₓ²) / (2Nₓ)</td>
</tr>
</tbody>
</table>

Goal is a single pipeline with no intermediate storage
SUMMARY

GPUs are the *de facto* platform for deploying Lattice Field Theory computations

Multigrid methods and GPUs are a potent combination

The community continues to be active in developing methods and software
   Many ways to run on GPUs: QUDA, GRID, Nim, OpenACC/OpenMP, etc.

GPU-centric computing is the scalable future

Lattice Field Theory is a well-placed application to saturate post-Exascale computing