BG/Q Code status

Chulwoo Jung
Brookhaven National Laboratory
(with sincere apologies & thanks to Bob Mawhinney, Balint Joo, James Osborn..)

May 4, 2012
USQCD All hands meeting
Fermilab
## IBM BG/Q

<table>
<thead>
<tr>
<th></th>
<th>BG/L</th>
<th>BG/P</th>
<th>BG/Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core/node</td>
<td>2</td>
<td>4</td>
<td>16+1(+1)</td>
</tr>
<tr>
<td>SIMD/core</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Clock speed (Mhz)</td>
<td>700</td>
<td>850</td>
<td>1600</td>
</tr>
<tr>
<td>Threads/node</td>
<td>2</td>
<td>4</td>
<td>16×4</td>
</tr>
<tr>
<td>Peak/node (GFlops)</td>
<td>5.6</td>
<td>13.9</td>
<td>205</td>
</tr>
<tr>
<td>L1/core (KB)</td>
<td>32</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>L3/node (MB)</td>
<td>4</td>
<td>8</td>
<td>32 (L2)</td>
</tr>
<tr>
<td>Memory (GB) / node</td>
<td>1</td>
<td>2</td>
<td>16</td>
</tr>
<tr>
<td>DDR Bandwidth (GB/s)</td>
<td>5.6</td>
<td>13.6</td>
<td>40</td>
</tr>
<tr>
<td>Torus Dimension</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Bandwidth (GB/s)</td>
<td>2.1</td>
<td>5.1</td>
<td>~40</td>
</tr>
<tr>
<td>Watt/Rack (KW)</td>
<td>~20</td>
<td>~40</td>
<td>~80</td>
</tr>
<tr>
<td>Linpack GF/W</td>
<td>0.23</td>
<td>0.37</td>
<td>1.68</td>
</tr>
</tbody>
</table>
### BFM CG performance (GFlops/node, 1 rack, w SPI)

<table>
<thead>
<tr>
<th>Fermion</th>
<th>( V )</th>
<th>( L_s )</th>
<th>Double</th>
<th>Single*</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWF(Shamir)</td>
<td>( 6^3 \times 8 )</td>
<td>8</td>
<td>34</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>( 6^3 \times 8 )</td>
<td>16</td>
<td>41</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>( 8^3 \times 8 )</td>
<td>8</td>
<td>43</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>( 8^3 \times 8 )</td>
<td>16</td>
<td>34</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>( 10^3 \times 8 )</td>
<td>8</td>
<td>32</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td>( 10^3 \times 8 )</td>
<td>16</td>
<td>24</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>( 12^3 \times 12 )</td>
<td>8</td>
<td>20</td>
<td>47</td>
</tr>
<tr>
<td>Wilson</td>
<td>( 6^3 \times 8 )</td>
<td></td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>( 8^3 \times 8 )</td>
<td></td>
<td>24</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>( 10^3 \times 8 )</td>
<td></td>
<td>33</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>( 12^3 \times 12 )</td>
<td></td>
<td>26</td>
<td>50</td>
</tr>
<tr>
<td>WilsonTM</td>
<td>( 6^3 \times 8 )</td>
<td></td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>( 8^3 \times 8 )</td>
<td></td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>( 10^3 \times 8 )</td>
<td></td>
<td>29</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>( 12^3 \times 12 )</td>
<td></td>
<td>24</td>
<td>45</td>
</tr>
</tbody>
</table>
Weak Scaling for BAGEL DWF CG Inverter

Tests were performed with the STFC funded DiRAC facility at Edinburgh

16K, $8^4 \times 8$ local volume, double precision
Integration/optimization status

BFM/BAGEL (Peter Boyle)

- DWF(5d,4d), Mobius, Wilson, WilsonTM Inverters, with QMP and/or SPI
- QDP++ interface included. In use with UKHADRON (Chroma)
- Provide threading framework rather than just dslash for other routines (eg. Arnoldi/Lanczos (R. Arthur) EigCG (Qi Liu))
- Single precision with SPI not yet fully debugged/optimized. So far only works with 1 MPI per node & contiguous mapping (logical nearest nodes should be also physically nearest)
  A bit slower with QMP/MPI
- Clover? : Similar structure to WilsonTM. Just needs the clover term!

Chulwoo Jung  Brookhaven National Laboratory (with sincere :  BG/Q Code status)
CPS

- DWF(5d), WilsonTM integrated into CPS. Mobius in progress (Hantao Yin).
- Mixed precision solver, MADWF.... forthcoming
- Other parts??
  Relevant part for DWF evolution threaded with openMP, getting 5-10GFlops in double precision.
  So far only with openMP and GCC, no QPX, room for improvement.
  A lot of routines in Lattice QCD are memory bandwith bounded (flops/Byte < 1)
  → should be able to get reasonably close to "theoretical" peak by organizing the code to minimize the bandwidth, with better compiler (XLC..)
Chroma (From Balint Joo)

- Current status
  - Chroma built and run on prototype hardware in 2011
  - No linkage with optimized/assembly kernels yet
  - Currently working on deployment at ANL (also LLNL) and Bagel/BFM linkage

- Targets for 2012
  - Deploy QDP++/Chroma, link to optimized libraries (e.g. Bagel)
  - Optimize important QDP++ expressions
  - Port and check effectiveness of site-vectorized code for BG/Q

- Threading?
  - QDP++ has a simple thread dispatching mechanism
  - Should work with OpenMP and QMP (if it is ported)
MILC: See James’ talk!
Software Targets for 2012

- Site Vectorized Dslash and Intel MIC (c.f. Chips talk earlier)
  - In collaboration with Intel Parallel Computing Labs
  - Aim initially for “level 3” Wilson-Clover solver
  - QDP++ & Chroma with compatible data layout (‘scalarvec’)
  - All x86 compatible targets benefit: Xeon, Interlagos (& Cray XE)
- BG/Q
  - Deploy QDP++/Chroma, link to optimized libraries (e.g. Bagel)
  - Optimize important QDP++ expressions
  - Port and check effectiveness of site-vectorized code for BG/Q
- Gauge Generation on GPU based Leadership Class Systems
  - Chroma on CPU + QUDA, Chroma over QDP-JIT + QUDA
  - In collaboration with QUDA developers, QDP-JIT with Frank Winter
- Chroma general maintenance: e.g. integrating updated QIO etc
Current Status

- **Site Vectorized Dslash**
  - Dslash exists, ongoing optimization & cleanup
  - Most optimization so far on single socket,
  - NUMA aware branch, & MPI boundary communications exist.

- **GPUs**
  - Have run Wilson HMC (2 flavor, with Hasenbusch prec.) on TitanDev
  - Both Chroma(CPU)+QUDA and Chroma(QDP-JIT)+QUDA
  - For Cray XK, QDP-JIT needs kernels pre-generated elsewhere
  - Work needed on Clover, and non-solver Level 3
  - Concern: lack of mixed precision multi-shift solver for RHMC.
    - various workarounds available (eg. solve in SP, polish up in DP etc)

- **BG/Q**
  - Chroma built and run on prototype hardware last year
  - No linkage with optimized/assembly kernels yet
  - Currently working on deployment at ANL (also LLNL) and Bagel BFM linkage
QCD on GPUs: Current Status and Future Plans

Mike Clark
FNAL
4th May 2012
Outline

- Current Status of QUDA
- QUDA development roadmap
- Kepler preview
Anatomy of a GPU

- Deep memory hierarchy (bandwidth or latency)
- This is what the exascale will look like
- Efficient implementation must use exploit all locality
- Fortunately LQCD has lots of locality
QCD and NVIDIA

- Why does NVIDIA care about QCD?
  - Important HPC application
  - QUDA used to debug current toolkit and SDK
- QCD is one of the HPC benchmarks run on all future GPU simulators
  - Ron Babich working on NVIDIA Exascale GPU project
  - Ensures that future GPUs will always be great at QCD
  - Influencing design decisions (e.g., cache size)
QUDA is a library of solvers and performance critical kernels for lattice QCD on CUDA GPUs

- Highly optimized multi-GPU implementations of the Dirac operators
- ‘Standard’ Krylov Solvers for QCD: CG(NE), BiCGStab
- Domain-decomposition preconditioned solvers (additive and multiplicative)
- Now includes auxiliary kernels for HMC
- Key Optimizations
  - Spatial-cache blocking
  - Memory-traffic-reducing transformations
  - Mixed-Precision (16 bit, 32 bit, 64 bit) solvers
  - Field Compression
  - Aggressive Autotuning

http://lattice.github.com/quda

QUDA Consortium:
Ron Babich (NVIDIA), Rich Brower (BU), Mike Clark (NVIDIA), Justin Foley (Utah), Bálint Jóó (Jefferson Lab), Guochun Shi (NCSA), Alexei Strelchenko (Cyprus), Kostya Petrov (Peta QCD), Joel Giedt (RPI), Steve Gottlieb (Indiana)
QUDA on Fermi

- Fermi is a two-year-old architecture so performance very well understood
- Fallacy that Fermi has too small a cache for QCD
  - Actually ideal balance between registers and shared memory for Wilson spatial cache blocking (SP)
  - Wilson spatial-cache blocking achieves 79% of an infinite cache (SP)
- Capacity solver performance:
  - Chroma Wilson-clover: 1 GPU ~ 100 cores
  - MILC Improved staggered: 1 GPU ~ 50 cores
- Issues
  - PCIe 2.0 and IB comms must be pipelined through host memory
  - Register-per-thread limitation causes large reduction in DP perf.
Benchmarks: + GPU (BiCGStab)

Strong Scaling: $48^3 \times 512$ Lattice (Weak Field), Chroma + QUDA

Results from TitanDev
- $48^3 \times 512$ aniso clover
- scaling up 768 GPUs

Friday, May 4, 12
Schwarz Methods aka Domain Decomposition (Schwarz, 1870)
- Partition the problem into domains
- Impose Dirichlet BC and solve independently
- Use union of domain solutions as a preconditioner

Multiplicative Schwarz:
- Update domains sequentially and use most recent solutions \textit{Block Gauss Seidel}
- Coloring algorithms used to parallelize

Additive Schwarz:
- Update domains simultaneously \textit{Block Jacobi}
- Algorithm trivially parallel

\[
K = D_{\Omega}^{-1} + D_{\Omega^*}^{-1} + D_{\partial\Omega}^{-1}D_{\partial\Omega^*}^{-1}
\]
Benchmarks: + GPU (DD+GCR)

Results from TitanDev
- $48^3 \times 512$ aniso clover
- scaling up 768 GPUs
Bound by what?

- Timing breakdown at 768 GPUs
- This tells us how we will scale
  - Doubling the GPU speed, same PCIe $\Rightarrow 1.63x$
  - Doubling the GPU and PCIe $\Rightarrow 1.80x$
- Increasingly latency bound due to orthogonalization
- Now need to think about both comms and latency
- Do more useful local work $\Rightarrow$ overlapping blocks?
- Do less reductions $\Rightarrow$ modGMRES?

<table>
<thead>
<tr>
<th></th>
<th>Percentage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>65%</td>
<td>kernel with no communication</td>
<td></td>
</tr>
<tr>
<td>15%</td>
<td>kernel with local communication</td>
<td></td>
</tr>
<tr>
<td>11%</td>
<td>orthogonalization (global reductions)</td>
<td></td>
</tr>
<tr>
<td>7%</td>
<td>solver restart (local and global)</td>
<td></td>
</tr>
</tbody>
</table>
QDP-JIT

- What about the parts of Chroma that QUDA doesn’t accelerate?
- QDP-JIT - port QDP++ to CUDA [https://github.com/fwinter/qdp](https://github.com/fwinter/qdp)
- Designed to interoperate with QUDA
- QUDA accelerates time-critical routines
- QDP-JIT accelerates everything else
- Uses JIT to create CUDA kernels at runtime
- Allows Chroma to run unaltered on (multiple) GPUs
Chroma HMC

- Hot-off-the-press benchmark
- Thanks Balint and Frank
- Wilson HMC, $32^3 \times 96$
- Simple CG solver
- Proof of concept
- Vast improvements possible
- Solo Wilson Dslash done on CPU
- GPUDirect
- QUDA $\Leftrightarrow$ QDP-JIT interop
- DD solver
QUDA Clover Plans

- Implement missing clover HMC functionality
- Clover term
  - $F_{\mu\nu}$ kernel in place
- Need clover computation, trace det and clover inverse
- Generalize fat-link computation for stout-link generation
- Allow QDP-JIT to directly share send/receive GPU buffers
- All ingredients then in place for full clover gauge generation
- Target is summer
HISQ Fermions

- QUDA 0.4.1 will include support for
- Thanks to Justin and Guochun
- Multi-GPU HISQ force, fat-link, gauge-force
- Almost everything in place for full RHMC
- Proof-of-concept $V = 64^3 \times 96$ RHMD evolution
- Run on 128 C2070 at FNAL
- 1x C2070 = 24 CPU cores
- GPUDirect issue?

<table>
<thead>
<tr>
<th>Function</th>
<th>Calls</th>
<th>Time (s)</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG</td>
<td>6</td>
<td>149.8</td>
<td>2.2%</td>
</tr>
<tr>
<td>Multi-CG</td>
<td>515</td>
<td>5832.46</td>
<td>62.23%</td>
</tr>
<tr>
<td>Fatlink</td>
<td>102</td>
<td>410.323</td>
<td>6.03%</td>
</tr>
<tr>
<td>GF</td>
<td>300</td>
<td>282.855</td>
<td>4.16%</td>
</tr>
<tr>
<td>FF</td>
<td>100</td>
<td>1162.96</td>
<td>13.8425%</td>
</tr>
<tr>
<td>Others</td>
<td></td>
<td>562.988</td>
<td>6.70113%</td>
</tr>
<tr>
<td>total_time</td>
<td></td>
<td>8.401388e+03</td>
<td></td>
</tr>
</tbody>
</table>

current

<table>
<thead>
<tr>
<th>Function</th>
<th>Calls</th>
<th>Times</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-CG</td>
<td>515</td>
<td>3597.6</td>
<td>63%</td>
</tr>
<tr>
<td>Fatlink</td>
<td>102</td>
<td>410.323</td>
<td>7%</td>
</tr>
<tr>
<td>GF</td>
<td>300</td>
<td>282.855</td>
<td>5%</td>
</tr>
<tr>
<td>FF</td>
<td>100</td>
<td>814.07</td>
<td>14%</td>
</tr>
<tr>
<td>Others</td>
<td></td>
<td>562.988</td>
<td>11%</td>
</tr>
<tr>
<td>total_time</td>
<td></td>
<td>5.667e+03</td>
<td></td>
</tr>
</tbody>
</table>

fixed (3 weeks from now )
HISQ Fermions Plans

- Performance potholes
- Missing kernels: outer products, long link
- Fully GPU-resident pipeline
- Fix naik-epsilon refinement issue
- Adopt QDP/C-style HISQ force
- Staggered fermion has less spatial locality than Wilson
- Performance is significantly worse
- Deploy multi-source solvers to vastly improve the spatial locality ⇒ more pseudofermions in RHMC
- Implement a domain-decomposition solver
- Use OpenACC for non-accelerated portions of MILC?
Domain-Wall Fermions

- QUDA 0.4.1 will include support for multi-GPU domain-wall fermions
- Thanks to Joel Giedt, Alexei Strelchenko and Chris Schroeder
- Single GPU dslash (Fermi SP, not yet optimized): 188 Gflops
- Add spatial-cache blocking (as Wilson): 246 Gflops
- Gauge-field reuse (as staggered): 420 Gflops
- Not yet deployed (in the US, in production in Cyprus)
- Need to get the CPS-QUDA interface into main trunk of CPS
- Joel working on qcdlib interface for solver
QUDA plans

• Exploit GPU Direct 2.0
• peer-2-peer communication within a node
• improves intra-GPU bandwidth and latency
• leaves PCIe bandwidth for IB
• Two summer projects
• Adaptive multigrid
• Kepler performance optimizations
• Please make requests
• Or even better - please join the fun
and now a word from my sponsors.....
May 14th-17th
San Jose, California

300 Sessions
120 Research Posters
100 Sponsors and Exhibitors
2 Co-located Events - LANL AHPC Symposium & InPar
1 Emerging Companies Summit

http://www.gputechconf.com
We’re hiring...
Domain Decomposition

- Non-overlapping blocks - simply have to switch off inter-GPU communication
- Preconditioner is a gross approximation
- Use an iterative solver to solve each domain system
- Require only 10 iterations of domain solver \( \Rightarrow \) 16-bit
- Need to use a flexible solver \( \Rightarrow \) GCR
- Block-diagonal preconditioner impose \( \lambda \) cutoff
- Finer Blocks lose long-wavelength/low-energy modes
  - keep wavelengths of \( \sim O(\Lambda_{\text{QCD}}^{-1}) \), \( \Lambda_{\text{QCD}}^{-1} \sim 1\text{fm} \)
- Aniso clover: \( (a_s=0.125\text{fm}, a_t=0.035\text{fm}) \) \( \Rightarrow \) 8\(^3\)x32 blocks are ideal
- 48\(^3\)x512 lattice: 8\(^3\)x32 blocks \( \Rightarrow \) 3456 GPUs
Flops versus speedup

Babich, Clark, Joo, Shi, Brower, Gottlieb, “Scaling Lattice QCD beyond 100 GPUs,” SC’11
Near-term Software for SciDAC 3
ALCF, MILC/HISQ, QDP/C, Multigrid, FUEL edition

James C. Osborn
Argonne Leadership Computing Facility

USQCD All Hands Meeting
May 4-5, 2012
FNAL
ALCF BG/Q “Mira” Racks
T&D racks “Cetus” and “Vesta”
ALCF BG/Q Hardware

First two racks delivered 1/14/2012
Test & Development (T&D) machines
   One rack: Very Early Access System (VEAS)
Sixteen *Mira* racks delivered

Vesta  Mira  Cetus

Surveyor  Intrepid  Challenger
Evolving timeline for Mira availability

ESP access to Mira likely soon

- ESP projects *may* have access to Mira as early as summer 2012
- Time for Early Science runs is likely to be earlier than expected
  - Possibly second half of CY2012
- Get on the T&D *now*
  - ESP is currently a top priority but there are many pressures on the hardware
  - Important for INCITE proposal preparation, especially computational readiness on BG/Q

Plan for a 2013 INCITE proposal on Mira

- Mira is committed to go live for INCITE on October 1, 2013 with 768M core-hours for allocation
- Start date of production INCITE time is likely to happen earlier in CY 2013
- Guidance for INCITE 2013
  - Propose science based on a 3B core-hour pool, 100-300M per project
  - 2013 INCITE Allocation scenarios
    - 768M
    - 2B core-hours
    - 3B core-hours

2012

Q1 2012

VEAS and T&D

Possible Mira

ESP Mira Access

Possible INCITE

INCITE

2013

Q4 2013

HISQ on Q

- MILC + SciDAC QOPQDP and supporting libraries
  - QOPQDP provides optimized HISQ solver, fermion and gauge forces
- Need to use 32-64 threads per node
- Threading strategies:
  - “MPI everywhere”: multiple MPI ranks per node, no threading
    - no changes to MILC
    - may not be as efficient
  - OpenMP (parallel loops)
    - Exists in QLA, and also some parts of MILC
    - Noticeable overhead for fine-grained parallelism
  - Application level threads
    - Whole application (or major routines) are fully threaded
    - Threads work on independent parts of loops as they are encountered
    - Needs occasional synchronization
    - Exists in development version of QDP/C, works, needs further testing and cleanup
    - Haven't tested it in QOPQDP yet
Initial tests and optimization for BG/Q

- Focus on MPI-everywhere to start, transition to fully threaded code once up and running with reasonable efficiency

- Initial optimizations
  - QLA: add optimized QPX code for key routines
    - IBM provided inline asm versions of a few QLA routines a while back
    - Have rewritten some in xlc intrinsics to aid modification and maintainability
    - Staggered matrix-vector product gets up to 30% peak
  
  - QMP: use low level (SPI) communications instead of MPI (with Heechang Na)
    - Avoids MPI overheads, reduces latency by up to 8x for small messages (1.5-3x typical)
    - Have working version of QMP using SPI
    - SPI use not automatic, user must declare send/recv pair to use SPI instead of MPI
    - Added synchronization optimization calls to QMP
      - QMP_clear_to_send(message, FLAG)
      - FLAG = { QMP_CTS_DISABLED, QMP_CTS_NOT_READY, QMP_CTS_READY }
## HISQ CG solver performance
128 nodes using $32^3 \times 64$ lattice (8x8x16x16 per node)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>GFLOPS/node (Peak%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain MILC or QOPQDP compiled with xlc -O3 MPI, simple mapping</td>
<td>~ 10 Gflops/node (5%) peak</td>
</tr>
<tr>
<td>As above, with optimized mapping (64 threads = 2x2x4x4 block)</td>
<td>~ 11 Gflops/node (5.3% peak)</td>
</tr>
<tr>
<td>As above, compiled with -O5 (looks correct, but not thoroughly checked)</td>
<td>~ 12 Gflops/node (5.9% peak)</td>
</tr>
<tr>
<td>QOPQDP + QPX code in QLA optimized mapping (c32 mode)</td>
<td>~ 16 Gflops/node (7.6% peak)</td>
</tr>
<tr>
<td>As above + QMP-SPI</td>
<td>~ 20 Gflops/node (9.7% peak)</td>
</tr>
</tbody>
</table>

Some overheads remain: MPI global sums, QDP communications bookkeeping
MPI-everywhere upper limit probably around 12-15%
Now time probably better spent working on fully threaded code

Wilson clover
128 nodes using $32^3 \times 64$ lattice (8x8x16x16 per node)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain QOPQDP, xlc -O3, QMP-MPI, optimal mapping</td>
<td>~12 Gflops/node (5.9% peak)</td>
</tr>
<tr>
<td>with QMP-SPI</td>
<td>~15 Gflops/node (7.1% peak)</td>
</tr>
</tbody>
</table>

QPX code not available yet
Wilson clover multigrid (3 level)
128^3 x 96 quenched lattice (Karsch+collabs.)
512 BG/Q nodes (64 ranks/node), plain C, xlc -O3, QMP-MPI

BiCGStab total speed
~ 5.8 Tflops
(5.6% peak)

Multigrid effective rate
~ 184 Tflops
(176% peak)
Wilson clover multigrid
total time for fixed number of solves
New FUEL HMC framework
(Framework for Unified Evolution of Lattices)

- High level layer focused on gauge configuration generation
  - motivation is to have flexible HMC framework to support wide range of beyond standard model theories
  - algorithmic abstraction: generation algorithm independent of gauge group, action, etc.
  - easy to write new high-level algorithms, tune parameters
  - serves as wrapper for efficient “level 3” routines
  - easy to plug in new routines
  - new routines can be written in any other language/framework

- Uses scripting language Lua
  - Small
  - Easy to port (ANSI C89)
  - Easy to use, yet powerful
  - Easy to embed and interface with libraries
Initial prototype

- Initially using SciDAC QDP/C and QOPQDP libraries to provide needed routines

- Supports SU(3) lattice generation with HISQ
  - using RHMC (Rational Hybrid Monte Carlo) algorithm
  - also supports HMC with mass preconditioning

- Matches conventions of current MILC code (and can parse same input files)

- Being used by Lattice BSM collaboration for 8f HISQ (G. Fleming's talk)
Current plans

- Use as testbed for algorithmic research
  - test HMC algorithms for large $N_f$
  - improved integrators (e.g. force gradient)
  - plug in improved solvers (e.g. multigrid)

- Add Wilson and domain wall quark support
- Add QUDA (GPU) routines as alternatives
- Add SU(2) support
- Plus many other actions mentioned in HEP SciDAC 3 proposal