Solving Domain Wall Dirac Equation Using Multisplitting Preconditioned Conjugate Gradient

Jiqun Tu

1Department of Physics, Columbia University

1The 36th International Symposium on Lattice Field Theory, July 23, 2018 @ 16:10

Talk based on: Duo Guo, Robert D. Mawhinney, and Jiqun Tu, [arXiv:1804.08593].
Special thanks to Norman Christ, Chulwoo Jung, and Christopher Kelly.

### The RBC & UKQCD collaborations

<table>
<thead>
<tr>
<th>BNL and BNL/RBRC</th>
<th>Tianle Wang</th>
<th>University of Liverpool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yasumichi Aoki (KEK)</td>
<td>Evan Wickenden</td>
<td>Nicolas Garron</td>
</tr>
<tr>
<td>Mattia Bruno</td>
<td>Yidi Zhao</td>
<td><strong>MIT</strong></td>
</tr>
<tr>
<td>Taku Izubuchi</td>
<td><strong>University of Connecticut</strong></td>
<td></td>
</tr>
<tr>
<td>Yong-Chull Jang</td>
<td>Tom Blum</td>
<td>David Murphy</td>
</tr>
<tr>
<td>Chulwoo Jung</td>
<td>Dan Hoying (BNL)</td>
<td>Peking University</td>
</tr>
<tr>
<td>Christoph Lehner</td>
<td>Luchang Jin (RBRC)</td>
<td>Xu Feng</td>
</tr>
<tr>
<td>Meifeng Lin</td>
<td>Cheng Tu</td>
<td><strong>University of Southhampton</strong></td>
</tr>
<tr>
<td>Aaron Meyer</td>
<td><strong>Edinburgh University</strong></td>
<td></td>
</tr>
<tr>
<td>Hiroshi Ohki</td>
<td>Peter Boyle</td>
<td>Jonathan Flynn</td>
</tr>
<tr>
<td>Shigemi Ohta (KEK)</td>
<td>Guido Cossu</td>
<td>Vera Guelpers</td>
</tr>
<tr>
<td>Amarjit Soni</td>
<td>Luigi Del Debbio</td>
<td>James Harrison</td>
</tr>
<tr>
<td><strong>UC Boulder</strong></td>
<td>Tadeusz Janowski</td>
<td>Andreas Juettner</td>
</tr>
<tr>
<td>Oliver Witzel</td>
<td>Richard Kenway</td>
<td>James Richings</td>
</tr>
<tr>
<td>Columbia University</td>
<td>Julia Kettle</td>
<td>Chris Sachrajda</td>
</tr>
<tr>
<td>Ziyuan Bai</td>
<td>Fionn O’haig</td>
<td><strong>Stony Brook University</strong></td>
</tr>
<tr>
<td>Norman Christ</td>
<td>Brian Pendleton</td>
<td>Jun-Sik Yoo</td>
</tr>
<tr>
<td>Duo Guo</td>
<td>Antonin Portelli</td>
<td>Sergey Syritsyn (RBRC)</td>
</tr>
<tr>
<td>Christopher Kelly</td>
<td>Tobias Tsang</td>
<td><strong>York University (Toronto)</strong></td>
</tr>
<tr>
<td>Bob Mawhinney</td>
<td>Azusa Yamaguchi</td>
<td>Renwick Hudspith</td>
</tr>
<tr>
<td>Masaaki Tomii</td>
<td><strong>KEK</strong></td>
<td></td>
</tr>
<tr>
<td>Jiqun Tu</td>
<td>Julien Frison</td>
<td></td>
</tr>
<tr>
<td>Bigeng Wang</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Move Over, China: U.S. Is Again Home to World’s Speediest Supercomputer

June 8, 2018

**Figure 1:** The New York Times’s comment on SUMMIT becoming world’s most powerful supercomputer.
Figure 2: Half precision Möbius domain wall fermion CG weak scaling with local volume of $16 \times 12^3 \times 12$. 6 NVIDIA Volta GPUs on each compute node. Numbers provided by Chulwoo Jung.
Motivation

• Inter-processor communication is the bottleneck for Dirac equation solving.
• For measurement there are many approaches available to improve the situation: Lanczos, EigCG, split-grid, multigrid, etc.
• Not the case for evolution.
• Need an (better) algorithm to reduce the communication overhead and exploit the fascinating local GPU flops.
• Do more work locally!
Previous Work

- Domain Decomposition/Multiplicative Schwarz [M. Lüscher 2004].
- Addtive Schwarz [Y. Osaki 2000] and [R. Babich 2011].
For reference see [D. O’leary 1985].

\[ Ax = b : A_l x_l + A_s x_s + A_r x_r = b_s \]
Solve

\[ A_l x_l + A_s x_s + A_r x_r = b_s, \]

Rearrange into an iterative form

\[ A_s x_s^{(k+1)} = b_s - A_l x_l^{(k)} - A_r x_r^{(k)} \]
\[ = b_s - (A x^{(k)} - A_s x_s^{(k)}) \]
\[ = r^{(k)} + A_s x_s^{(k)} \equiv \hat{b}_s^{(k)} \]

For each cycle,

- use communication to calculate the right-hand-side \( \hat{b}_s \).
- solve \( A_s x_s^{(k+1)} = \hat{b}_s^{(k)} \) locally.
- the updated solution \( x_s^{(k+1)} \) will be used to ready the next cycle.

Get \( A_s \) for each node by chopping off all off-block-diagonal terms: applying zero Dirichlet boundary condition.
Even-odd preconditioning:

\[
\begin{pmatrix}
M_5 & -\kappa_b M^4_{eo} \\
-\kappa_b M^4_{oe} & M_5
\end{pmatrix}
\begin{pmatrix}
\psi_e \\
\psi_o
\end{pmatrix}
= \begin{pmatrix}
\phi_e \\
\phi_o
\end{pmatrix},
\]

then instead we solve

\[
D_{PC}\psi_e = \hat{\phi}_e, \quad D_{PC} \equiv M_5 - \kappa_b^2 M^4_{eo} M_5^{-1} M^4_{oe},
\]

\[
M^4_{oe/ eo} = D^w_{x,y} (b_5 \delta_{s,t} + c_5 D^5)
\]

\[
D^w_{x,y} = \sum_{\mu} \left[ (1 + \gamma_\mu) U_{\hat{\mu}, \mu}^\dagger \delta_{x-\hat{\mu}, y} + (1 - \gamma_\mu) U_{\mu, x}^\dagger \delta_{x+\hat{\mu}, y} \right].
\]

Using CG:

\[
D_{PC}^\dagger D_{PC}\psi_e = D_{PC}^\dagger \hat{\phi}_e
\]
• 4 hopping terms in the normal operator:

\[
A = D_{PC}^{\dagger} D_{PC} \\
= (M_5 - \kappa_b^2 M_{eo}^4 M_5^{-1} M_{oe}^4)^\dagger (M_5 - \kappa_b^2 M_{eo}^4 M_5^{-1} M_{oe}^4)
\]

• This means we need to enforce Dirichlet boundary condition on \( D_{PC}^{\dagger} D_{PC} \) instead of the individual hopping terms \( M_{eo/oe}^4 (D_{x,y}^w) \).

• Need to include the \textit{snake} terms: terms that hop out of the boundary and hop back.

• Seems obvious but not trivial to implement.
The Normal Operator

The snake terms:
before 1st hopping term
before 1st hopping term
after 1st hopping term
before 2ed hopping term
after 2ed hopping term
before 3rd hopping term
before 4th hopping term
- The algorithm converges with inclusion of the snake terms.
- The convergence rate is slow.
- Similar to [M. Lüscher 2004] we use its first cycle with zero initial guess as a preconditioner for CG.
- We use plain CG for the preconditioner solve. Instead of setting a precision stopping condition we iterate for a fixed number of times (the inner iteration count).
As a Preconditioner

\[ r_0 = b - Ax_0 \]
\[ z_0 = M^{-1}r_0 \]
\[ p_0 = z_0 \]
\[ k = 0 \]

\textbf{while} have not converged \textbf{do}

\[ \alpha_k = \langle r_k, z_k \rangle / \langle p_k, Ap_k \rangle \]
\[ x_{k+1} = x_k + \alpha_k p_k \]
\[ r_{k+1} = r_k - \alpha_k Ap_k \]
\[ z_{k+1} = M^{-1}r_{k+1} \leftarrow A_s x_s^{(k+1)} = r^{(k)} + A_s x_s^{(k)} \]

\textbf{end while}

\textbf{only first cycle, zero initial guess, iterate a fixed number of times}\n
\[ \beta_k = \langle z_{k+1}, r_{k+1} \rangle / \langle z_k, r_k \rangle \]
\[ p_{k+1} = z_{k+1} + \beta_k p_k \]
\[ k = k + 1 \]
As a Preconditioner

\[ A \]

\[ M = \bigoplus_s A_s \]
• Although starting from a different origin, this is now effectively the same with additive Schwarz if one treats the Dirichlet boundary condition correctly.
• Inclusion of the snake terms is crucial.
• Naming issue: [A Unified Representation and Theory of Algebraic Additive Schwarz and Multisplitting Methods, A. Frommer 1997].
• Multisplitting Preconditioned CG(MSPCG).
Result: $32^3 \times 64$

Figure 3: MSPCG solve on a $32^3 \times 64$ lattice ($a^{-1} = 1.37$ GeV) with physical pion mass. Test performed on CORI at NERSC on 128 KNL nodes.
Figure 4: MSPCG solve on the same lattice. Test performed on 64 nodes at Piz Daint. Solving $D^\dagger D x = b$ instead of $D^\dagger D x = D^\dagger b$. Numbers from Kate Clark.
Figure 5: MSPCG solve on a $64^3 \times 128$ lattice ($a^{-1} = 2.36$ GeV) with physical pion mass. Plain CG takes 18092 iterations to converge to the same precision ($10^{-10}$). KNL at CORI.
Figure 6: MSPCG solve on a $80^2 \times 96 \times 192$ lattice ($a^{-1} = 3.00$ GeV) with physical pion mass. Test performed on CORI at NERSC with 1024 KNL nodes.
• We observe that the number of iterations for outer CG is greatly reduced even if the inner preconditioner is solved in a sloppy way, e.g. iterating only 3-6 times.

• Our observation is supported by several theoretical works, say, [G. Golub 1999] and [V. Simoncini 2003].

• Thus the number of preconditioner solve is a parameter that can be tuned to achieve maximum speed up.
For $16 \times 12^3 \times 12$ local volume on 4 Volta GPUs,

| preconditioner | 14.13 Tflops |

With the same local volume on 1024 6-Volta-nodes,

| outer | 1.55 Tflops/GPU |

Assuming a factor of 3 in outer iteration count reduction with 6 inner iterations, the speed up from MSPCG is:

\[
\frac{\left(\frac{3}{1.55}\right)}{\left(\frac{6 \times 1.87}{14.13 \times (6/4)} + \frac{1}{1.55}\right)} = 1.65
\]
• First tested in CPS.
• Fully implemented in Grid\(^1\) and Quda\(^2\) with help from Qlattice\(^3\).
• Great thanks to Kate Clark from NVIDIA.

---

1 [https://github.com/paboyle/Grid](https://github.com/paboyle/Grid)
2 [https://github.com/lattice/quda](https://github.com/lattice/quda)
3 [https://github.com/waterret/Qlattice](https://github.com/waterret/Qlattice)
• The amount of inter-processor communication could be reduced at the expense of more local floating point computation by using the multisplitting algorithm as a preconditioner for CG.

• If the local floating point computation is cheap enough this greatly speeds up domain wall fermion Dirac equation solving.
Future Work

- On going work on Quda: Speed up preconditioner dslash as much as possible.
- The same approach is expected to work for staggered fermion as well.
- Spectrum analysis of the matrix $A$ and the preconditioner $M$. 