A domain decomposition algorithm for improved staggered fermions on GPUs

Carleton DeTar, Justin Foley, Robert Sugar

June 15, 2013

Abstract

Lattice QCD is a numerical approach to the theory of the strong interaction. Calculations in this field answer fundamental questions about the nature of matter, provide insight into the evolution of the early universe, and play a crucial role in the search for new theories beyond the Standard Model of elementary particle physics. Massive computational resources are needed to achieve these goals, and Lattice QCD is a major HPC application. In recent years, lattice calculations have benefited from the use of GPU accelerators. However, in order to fully utilize the vast computational resources available on leadership-class platforms such as Blue Waters, new algorithms must be developed to extend strong scaling in lattice calculations to many hundreds or even thousands of GPUs. In particular, the development of efficient many-GPU algorithms for solving the large-scale linear systems that arise in lattice calculations is essential. In this report, we describe the implementation of domain-decomposition-based linear solvers for GPU calculations using the HISQ lattice formalism. Performance results obtained in multi-GPU calculations on Blue Waters are discussed. The algorithms described here also serve as a starting point for the development of more sophisticated methods which will be needed to effectively utilize future HPC technologies.

1 Introduction

Lattice QCD is a numerical treatment of Quantum Chromodynamics (QCD), the quantum field theory that describes the strong nuclear interaction. QCD is a theory of fundamental matter particles called quarks and particles called gluons that mediate the strong interaction. In this approach, space and time are approximated by a discrete four-dimensional grid. Quarks reside on the lattice sites and gluons are associated with links connecting adjacent lattice sites. Ultimately, physical quantities are computed by evaluating high-dimensional integrals numerically. Current state-of-the art calculations may involve integrals over more than a billion variables, which are usually evaluated using Markov-Chain Monte Carlo methods. Since its inception almost 40 years ago, Lattice QCD has developed into a major HPC application, and Lattice QCD codes are run at leadership-class facilities across the United States and abroad.

This project relates to the use of GPU accelerators to solve the massive linear systems that arise in lattice calculations. The solution of linear systems dominates Lattice QCD calculations, and iterative linear solvers were the first lattice codes ported to GPUs [1]. Unfortunately, large-scale multi-GPU calculations using conventional solver algorithms are communication bound and they exhibit poor scaling on large numbers of GPUs. The aim of this project is to implement novel preconditioners designed to reduce inter-processor communication in the solver, which will ultimately help to extend strong scaling on platforms like Blue Waters to a thousand GPUs and beyond.

The algorithms described here utilize the Highly-Improved Staggered Quark, or HISQ, fermion discretization used by the MILC collaboration. These algorithms have been integrated into QUDA [2, 3], a well-known open-source library for performing lattice calculations on NVIDIA GPUs.

2 Linear systems in lattice calculations

In lattice calculations, the linear system

$$A\phi = \eta \tag{1}$$

is solved to obtain ϕ. In this equation, A is a large, sparse, square matrix, and vectors ϕ, η are commonly referred to as quark fields. In the HISQ formalism, quark fields have three complex components per lattice site and $A = Q^\dagger Q$, where Q is the HISQ matrix. Hence, A is Hermitian positive-definite. The HISQ matrix is given by

$$\begin{aligned} Q_{s,s'} &= \frac{1}{2}\sum_{\mu=1}^{4}\Big(X_\mu(s)\delta_{s+\hat{\mu},s'} - X_\mu^\dagger(s')\delta_{s-\hat{\mu},s'} \\ &+ \epsilon_m W_\mu(s)\delta_{s+3\hat{\mu},s'} - \epsilon_m W_\mu^\dagger\left(s'\right)\delta_{s-3\hat{\mu},s'}\Big) + m\delta_{s,s'} \end{aligned} \tag{2}$$

In this expression, s, s' label lattice sites. μ runs over space-time directions, and $\hat{\mu}$ is a unit vector (one lattice spacing in length) in the μ direction. X_μ, W_μ are complex 3×3 matrices, and m is the quark mass, a positive real number. ϵ_m is a real-valued mass-dependent coefficient, which takes a small numerical value in practice. The matrices X_μ appearing in the one-hop terms are known as fat link variables, while the matrices W_μ that connect sites three lattice-spacing apart are called long link variables.

Usually, iterative Krylov methods are employed to solve Eq. 1. Given a starting guess for the solution, ϕ_0, and the corresponding residual vector $\rho_0 = A\phi_0 - \eta$, a sequence of approximate solutions $\phi_1, \phi_2, ..., \phi_n$ are constructed in the Krylov subspaces $\mathcal{K}_1, ...\mathcal{K}_n$, where $\mathcal{K}_n$ denotes the space spanned by $\{\rho_0, A\rho_0, ..., A^n\rho_0\}$.

In HISQ calculations, the Conjugate Gradient (CG) algorithm [4], which requires a Hermitian positive-definite matrix, is the solver of choice. The rate of convergence of the Krylov solver is governed by $\kappa(A)$, the condition number of the matrix A, which is the ratio of the absolute value of the largest eigenvalue of A to the absolute value of the smallest. A low condition number corresponds to a well-conditioned system. Preconditioning works by replacing the original linear system with an equivalent problem involving a matrix with a lower condition number. For example,

$$\bar{A}C^{\dagger}\phi = C^{-1}\eta, \quad \bar{A} = C^{-1}AC^{-1\dagger}, \tag{3}$$

where C is chosen so that $M = C^{\dagger}C$ approximates A in some sense. In order for preconditioning to be useful in practice, the calculation of the matrix-vector product $M^{-1}\rho$ must be relatively inexpensive. Clearly, $\bar{A}$ is Hermitian positive-definite, and CG can be applied to Eq. 3. In practice, the preconditioned CG algorithm solves Eq. 3 using a single evaluation of $M^{-1}\rho$ per iteration, and C^{-1} and $C^{\dagger -1}$ do not appear explicitly [5].

3 Accelerating Lattice QCD on GPUs

QUDA is an open-source library for performing multi-GPU calculations in Lattice QCD. The library is written in C++ and MPI and targets NVIDIA's CUDA architecture. QUDA includes linear solvers for multiple lattice discretization schemes, including CG solvers for the HISQ discretization.

Single-GPU benchmarks obtained using QUDA on K20X processors give performance figures of approximately 160 Gflops for linear solves using single-precision arithmetic. Performance in double-precision solves is about 80 Gflops. QUDA also includes mixed-precision solvers that implement the reliable update procedure proposed by Sleijpen and Van der Vorst [6]. In tests, QUDA's mixed-precision solvers produce results with double-precision accuracy, while almost matching single-precision performance.

The linear-solver performance observed in single-GPU benchmarks does not scale to large numbers of GPUs. In calculations involving multiple processors, the lattice is divided into regular subdomains and each processor operates on a single subdomain. The fat- and long-link matrices are copied to device memory only once, at the beginning of the solve. However, each solver iteration involves the transfer of quark-field data between processors acting on neighboring lattice subdomains. In practice, calculations using conventional solvers on a few hundred GPUs are completely communication bound

3.1 Non-overlapping additive Schwarz preconditioning

To improve multi-GPU solver performance, one can utilize preconditioning schemes that involve little or no inter-processor communication. Specifically, one can apply domain-decomposition, or Schwarz [7], preconditioning techniques. In the simplest approach - non-overlapping additive Schwarz preconditioning, the preconditioning operator M is defining by letting $M^{-1}\rho$ be the result of applying A^{-1} to the quark field ρ ignoring inter-processor communication. More formally,

$$M^{-1}\rho = \sum_{i} R_i^{\dagger} A_i^{-1} R_i \rho, \tag{4}$$

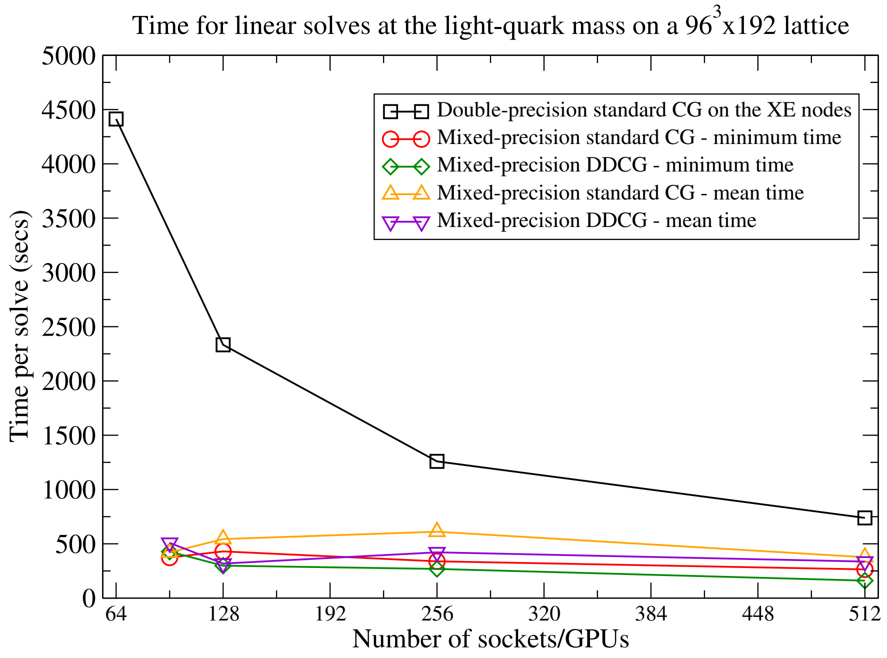

Figure 1: Times for linear solves at the light-quark mass on a $96^3{\times}192$ lattice on Blue Waters. 'DDCG' corresponds to the CG algorithm with non-overlapping additive Schwarz preconditioning. The black squares indicate data obtained using the MILC collaboration's software suite on the XE nodes. All other points were obtained by running QUDA on the XK7 nodes.

where the subscript i labels disjoint subdomains. The matrix R_i projects onto domain i, that is, the quark-field component $R_i\rho(s)$ is equal to $\rho(s)$ if the lattice site s lies in subdomain i, and is zero otherwise. Then, $A_i = R_i^\dagger A R_i$. Clearly, M defined in this way is Hermitian positive-definite and can therefore be used in preconditioned CG.

Since M^{-1} is a preconditioner, it need not be evaluated exactly. Hence, we use CUDA's half-precision data types in the preconditioner. Moreover, since (zero-field) Dirichlet boundary conditions are imposed on each subdomain, the matrices A_i are expected to be well conditioned and a few solver iterations should suffice to obtain $M^{-1}\rho$ to a reasonable accuracy. We can also use the minimal residual (MR) or steepest-descent algorithms in this step. Although much less robust than CG, these solvers involve fewer operations per iteration. Finally, the coefficient ϵ_m multiplying the long-link terms in the HISQ matrix is numerically very small. Omitting the long-link terms in the preconditioner reduces the number of arithmetic operations and global memory accesses needed to evaluate $M\rho$ by almost 50% without appreciably affecting the quality of the approximation to $A\rho$.

Tests of the algorithm were performed on a $96^3{\times}192$ lattice on the XK7 nodes on Blue Waters. The number of processors used ranged between 96 and 2304 GPUs. Two quark mass values were examined, corresponding to the light and strange quark

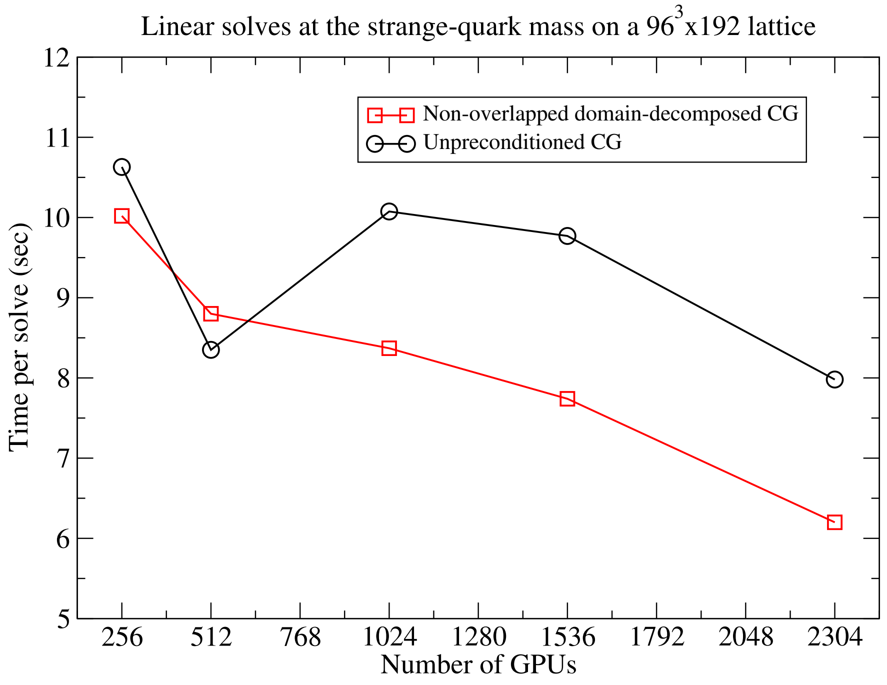

Figure 2: Solve times at the strange-quark mass on the XK7 nodes. On large numbers of GPUs, the use of non-overlapping additive Schwarz preconditioning reduces solve times by approximately 30%.

masses on this lattice. The outer solver iterations, used mixed single-double precision arithmetic, and the target relative residual in the solver ($||\rho||/||\eta||$) was set to 10^{-8}.

Between 5 and 10 MR iterations in the preconditioner gave optimal results, and Schwarz preconditioning was found to reduce the number of CG iterations by 40 to 50% for both quark masses and for different numbers of GPUs. Below a thousand GPUs, the reduction in inter-processor communication was offset by the additional cost of the preconditioning step, however, and additive Schwarz does not yet appear to offer any significant improvement over standard CG in this regime (Fig. 1). Fig. 1 also shows that the times to solution for linear solves at the light-quark mass are much lower on the GPU-enabled XK7 nodes than on the XE nodes for a given number of sockets. Fig. 2 shows GPU solver times for larger numbers of GPUs. In this case, domain decomposition does reduce average solve times by about 30%.

3.2 Restricted additive Schwarz preconditioning

We can mitigate boundary effects by allowing the lattice subdomains in the preconditioning step to overlap to some degree. This is represented graphically in Fig. 3. In this figure, different colors distinguish sites in disjoint lattice regions. Each lattice subdomain consists of an interior region, as well as boundary regions that coincide with sites in the interior regions of neighboring subdomains. At the beginning of the

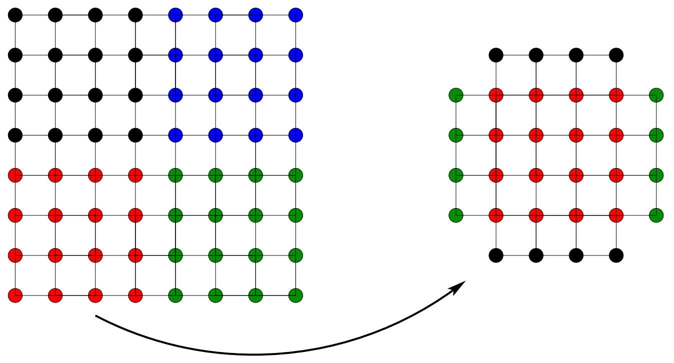

Figure 3: Overlapping domain decomposition of a 2D lattice. Each processor operates on a subdomain that consists of an interior region and boundary regions which coincide with sites in the interior regions of neighboring subdomains.

preconditioning step, quark-field data are fetched from neighboring subdomains and copied into the boundary regions. Let $\tilde{R}_i$ be the matrix that projects onto extended domain i. Then, $\tilde{A}_i^{-1}\tilde{R}_i\rho$ is evaluated on each subdomain, where $\tilde{A}_i = \tilde{R}_i^\dagger A \tilde{R}_i$. Finally the result is restricted to the interior region of each domain. Therefore,

$$M^{-1}\rho = \sum_i R_i \tilde{A}_i^{-1}\tilde{R}_i\rho, \tag{5}$$

where R_i is the projection matrix for non-overlapping decomposition, introduced earlier. This variant of Schwarz preconditioning is commonly referred to as Restricted Additive Schwarz [8].

Crucially, however, this definition of M^{-1} is not Hermitian and cannot therefore be used in a Conjugate Gradient solver. Alternative Krylov solvers that can handle non-Hermitian matrices include the BiCGstab and GMRes algorithms, and the Generalized Conjugate Residual, or GCR, algorithm, which has been used with considerable success in studies involving Wilson and Clover fermions [3, 9]. QUDA already featured a GCR solver with support for non-overlapping additive Schwarz preconditioning, and we modified this solver to support overlapping domains. Incorporating overlapping domain decomposition into QUDA involved significant reengineering of a number of low-level routines and data structures, which were not written with this type of application in mind.

QUDA now supports the Swiss-flag type decomposition illustrated in Fig. 3. Currently, the widths of the overlap regions at each end of a subdomain are constrained to be an even number number of lattice sites. However, domains need not overlap in all partitioned directions.

The vectors generated at each iteration of GCR are needed to extend the Krylov subspace in future iterations. Therefore, the size of the Krylov subspace that can be generated is restricted by the number of quark fields that can be held in memory, and the GCR algorithm has to be restarted periodically. In our tests, choosing the maximum

Krylov dimension, N_k, to be 20 gave reasonable convergence. On the $96^3{\times}192$ lattice and inverting at the strange-quark mass, the unpreconditioned GCR reached the desired tolerance in approximately 1800 iterations. However, on 256 GPUs it was possible to set $N_k = 60$ and the total number of iterations dropped below 800, with a similar reduction in time to solution.

In our tests, the lattice was partitioned in the y, z, and t directions, and we considered overlap widths of zero, two, and four lattice sites. In general, the use of overlaps in the preconditioner does result in a significant decrease in the number of outer GCR iterations. For example, using ten iterations of the MR algorithm and overlaps of width four in the z and t directions in the preconditioner reduced the number of GCR iterations from 800 to 210. However, this corresponded to a decrease in the time to solution of only about 10%. Similar relative improvement was observed at the light-quark mass.

In none of the tests, did the GCR algorithm with overlapping domains match the performance of of the CG solver using non-overlapped additive Schwarz preconditioning. It may be the case, however, that the overlapped additive Schwarz preconditioner fares better with an alternative outer solver, such as BiCGstab, and we will explore this possibility in the future.

4 Summary, on-going work, and future directions

We have performed a first study of domain-decomposition techniques in the linear solvers used in Lattice QCD calculations using the HISQ formalism. The ultimate aim of this program is to extend strong scaling on the XK7 nodes on Blue Waters up to and beyond a thousand GPUs.

We have implemented additive Schwarz preconditioning for the HISQ operator in the QUDA GPU library, and our code supports variable domain overlaps. The use of non-overlapping domains results in a Hermitian positive-definite preconditioner which can be incorporated into a CG solver. Allowing domains to overlap mitigates boundary effects but results in a preconditioning operator that is not Hermitian.

Tests performed on Blue Waters on a $96^3{\times}192$ lattice and using variable numbers of GPUs indicate that preconditioning reduces the number of CG iterations and, hence, the amount of inter-processor communication by 40 to 50%. Currently, this translates into a 30% reduction in solve times on large numbers of GPUs (≥ 1024). The use of overlapping domains in the preconditioning step in the GCR algorithm further reduces the number of outer solver iterations. However, this improvement is offset by the additional cost per iteration of GCR, which trails CG in performance.

This work connects closely to a number of current and future algorithm projects. One project relates to the way the long-link matrices appearing in the HISQ operator are stored in GPU global memory. Currently, each long link is stored as 18 floating-point numbers. However, the long-link matrices are unitary, and due to this additional constraint, they can be stored in a compressed format using 13 real numbers without loss of numerical precision. This compression will reduce memory-bandwidth pressure, and previous results from similar compression algorithms indicate a potential performance gain of up to 20% in the individual HISQ matrix kernels. The knock-on effect of this gain in kernel performance will be an increase in the proportion of overall time spent communicating data, which, in turn, may result in more significant performance gains from domain-decomposition methods.

In addition, the results presented here were obtained using CUDA's half-precision data types only in the preconditioning step. The outer solver iterations use mostly single-precision floating-point arithmetic with period reliable updates to obtain a final result that is double-precision accurate. So far, mixed-precision solvers that use half precision, instead of single precision, in the outer iteration have proven to be unstable. However, there are indications that a stable half-double precision solver could be obtained from a modified outer solver iteration combined with an efficient preconditioner. The use of half-precision arithmetic in the outer-solver iterations would result in a very significant increase in overall performance.

Furthermore, it may be that the optimal preconditioning scheme allows some degree of interprocessor communication. For example, the preconditioner might incorporate periodic exchange of the boundary data in certain directions. If we exclude the long-link terms in the HISQ matrix in the preconditioning step, the quark data that needs to be communicated corresponds to a boundary region that is only one lattice spacing in width. The software developed in this project can easily be modified to allow varying levels of inter-processor communication.

Finally, we note that the preconditioned solvers described in this report can be utilized as highly-parallel smoothers in a multigrid solver, A recent description of a lattice multigrid algorithm that uses GCR with multiplicative Schwarz preconditioning as a smoother can be found in Ref. [10].

References

[1] G. Egri et al. *Lattice QCD as a video game* Comput. Phys. Commun. 177, 631-639 (2007), arXiv:hep-lat/0611022.

[2] M.A. Clark et al. *Solving Lattice QCD systems of equations using mixed precision solvers on GPUs.* Comput. Phys. Commun. 181, 1517 (2010), arXiv:0911.3191[hep-lat].

[3] R. Babich et al. *Scaling Lattice QCD beyond 100 GPUs.* International Conference of High Performance Computing, Networking, Storage and Analysis (SC) 2011, arXiv:1109.2935[hep-lat].

[4] M.R. Hestenes and E. Stiefel. *Methods of Conjugate Gradients for Solving Linear Systems.* Journal of Research of the National Bureau of Standards 49 (6).

[5] J.R. Shewchuck. *An introduction to the Conjugate Gradient Method Without the Agonizing Pain.* `www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf`

[6] G. Sleijpen and H. van der Vorst. *Reliable updated residuals in hybrid Bi-CG methods* Computing 56 (2), 141-163 (1996).

[7] H.A. Schwarz. *Über einen Grenzübergang durch alternierendes Verfahren.* Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich 15 (1870) 272-286.

[8] X.-C. Cai and M. Sarkis. *A restricted additive Schwarz preconditioner for general sparse linear systems.* SIAM J. Sci. Comput., 21 792-797 (1999).

[9] M. Lüscher. *Schwarz-preconditioned HMC algorithm for two-flavour lattice QCD.* Comput.Phys.Commun. 165, 199-220 (2005).

[10] A. Frommer et al. *Aggregation based domain decomposition multigrid for the lattice Wilson Dirac operator.* arXiv:1303.1377[hep-lat].