Scalable Non-blocking Krylov Solvers for Extreme-scale Computing

Paul R. Eller, William Gropp
University of Illinois at Urbana-Champaign
{eller3,wgropp}@illinois.edu

ABSTRACT

This study investigates preconditioned conjugate gradient method variations designed to reduce communication costs by decreasing the number of allreduces and overlapping communication with computation using a non-blocking allreduce. Experiments show scalable PCG methods can outperform standard PCG at scale and demonstrate the robustness of these methods.

To develop the most optimal Krylov methods we need a clear understanding of the factors limiting performance at scale. Detailed timings and network counters are used to more thoroughly measure the performance of these methods. Performance models with penalty terms are developed that provide reasonable explanations of observed performance and guide development of optimizations. The effectiveness of scalable PCG methods and these performance analysis tools is demonstrated using Quda and Nek5000, two HPC applications seeking improved performance at scale.

1 INTRODUCTION

The preconditioned conjugate gradient method (PCG) is a popular method for solving sparse linear systems at scale. PCG requires frequent blocking allreduce collective operations that can limit performance at scale. We investigate PCG variations designed to reduce communication costs by decreasing the number of allreduces and by overlapping communication with computation using a non-blocking allreduce.

While scalable Krylov solvers improve performance at scale there are opportunities for further optimization. Performance analysis tools that measure performance variation and process network counters in detail are developed to better analyze performance at scale. Performance models help us understand key factors limiting performance at scale by comparing the expected performance with different assumptions to observed performance. These performance models guide us to develop optimizations that improve Krylov solver performance.

The Quda and Nek5000 applications will demonstrate the effectiveness of these solvers within real-world applications. Both applications are designed to perform well at scale but have room for improvement. These applications will require implementing and analyzing a variety of non-blocking pipelined Krylov solvers and use a variety of preconditioners. Initial experiments within these applications show effective performance consistent with the expected performance from previous studies.

This research addresses gaps in the literature by providing detailed performance studies for scalable Krylov solvers. Many existing studies primarily focus on the numerical properties of these methods with minimal focus on performance in practice. Furthermore few studies implement and analyze scalable solver performance in real-world applications. This is critical for demonstrating the effectiveness of these solvers due to the many challenges real-world applications face that are not found in test problems.

2 RESEARCH HIGHLIGHTS AND PLANS

This research provides a detailed study of scalable Krylov solvers that take advantage of non-blocking allreduces to overlap communication and computation. The first study focuses on analyzing the performance of scalable non-blocking pipelined PCG solvers [1]. The second study provides a detailed performance analysis and modeling study of scalable PCG solvers that produces further optimizations to improve performance at scale and is in preparation to be published [2]. The final study focuses on analyzing scalable Krylov solvers within two applications and further optimizing application performance using the tools we have developed.

2.1 Scalable PCG Solvers

We can rearrange PCG to decrease the number of allreduces or overlap communication and computation using recurrence relations to rearrange the order of key kernels to improve parallel performance. This process produces four scalable variations including PIPE2CG [1], a new method we developed guided by performance results that can hide the allreduce cost when the work per core becomes too small for PIPECG to overlap effectively.

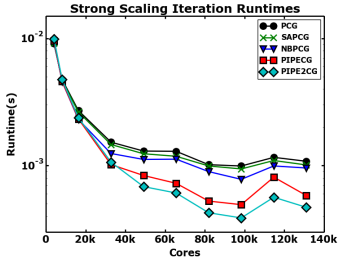

Figure 1: Strong scaling for 27-point Poisson matrix with 512^3 rows

Effectively implementing these methods requires taking advantage of merged vector operations that compute vector operations element-wise to avoid additional vector reads. Effectively using non-blocking allreduces requires calling MPI_Test() or using progress threads to make progress.

Weak scaling tests show more consistent performance for non-blocking solvers and significant slowdowns for blocking solvers at scale. Strong scaling tests in figure 1 show further pipelined methods can scale to higher core counts. A collection of experiments demonstrate the robustness of these methods. Notable results include noise experiments suggesting blocking methods make progress at the speed of the slowest core in each iteration while non-blocking methods make progress at close to the speed of the median core in each iteration, allowing non-blocking methods to reduce the impact of noise. Experiments with Poisson matrices and finite element problems show higher speedups for matrices with more nonzeros per row and fewer rows per core due to effectively overlapping the allreduce with computation while limiting the increased vector operation overhead.

2.2 Performance Analysis at Scale

To develop the fastest and most scalable Krylov solvers we need to develop a detailed understanding of the factors impacting performance at scale. Our previous work developed models that captured general solver trends but failed to produce accurate predictions for actual runtimes. Therefore we develop tools to measure performance variation and collect network performance counters to help understand performance at scale. This information is read into analytic performance models that show factors with the largest impact on performance and guide the development of optimizations.

A collection of blocking and non-blocking allreduce, halo exchange, and PCG routines are analyzed to better understand the effectiveness of overlapping communication and computation and to understand network performance. They demonstrate non-blocking routines tend to experience less congestion and overlap can hide the impact of congestion.

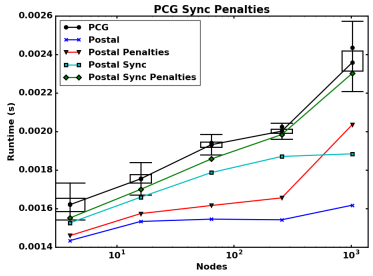

Figure 2: PCG with max-rate, hops, and congestion penalties with and without sync penalty

The postal model is used with penalty terms to model parallel performance. A max-rate penalty that limits off-node bandwidth when using multiple cores per node, a hop penalty to account for varying message distances, a congestion penalty to account for network contention, and a synchronization penalty to account for slowdown at synchronization points due to performance variation are used to produce more accurate performance models as shown in figure 2.

Optimizations are developed based on the performance models and tested using 3-d halo exchanges. MPI protocol changes allow larger messages to use the eager protocol and obtain more effective overlap. Node-aware algorithms minimize off-node communication to improve performance. Topology-aware algorithms assign processes so each node communicates with neighbors to reduce message distance and further improve performance over node-aware algorithms (figure 3). This study will be continued by developing PCG solvers using node-aware and topology-aware communication to improve performance.

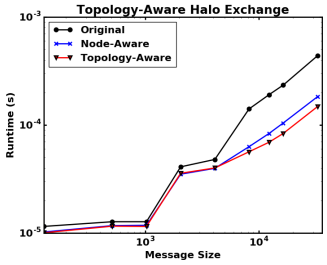

Figure 3: 3-d halo exchange using original, node-aware, and topology-aware algorithms

Similar experiments will be run on a system with a Cray Aries dragonfly network. This network provides improved performance due to reduced message distances and dynamic routing that can help reduce network congestion. These experiments will allow us to better understand the performance barriers on improved network topologies. Our network analysis tools are currently being adapted for Cray Aries systems and we have started running initial experiments.

2.3 Optimizing Application Performance

An application study demonstrating the effectiveness of non-blocking pipelined Krylov solvers for real-world problems will complete my thesis. These solvers will be used within the Quda and Nek5000 applications to improve performance at scale. Quda is a lattice quantum chromodynamics application with structured grids for GPU-based systems, while Nek5000 is a computational fluid dynamics application with unstructured grids designed for CPU-based systems. These applications will allow me to explore non-blocking PCG, GMRES, BiCGStab, and GCR solvers. These applications use a diverse collection of preconditioners allowing me to explore the effectiveness of these preconditioners for non-blocking solvers. Our performance analysis tools will be used with these applications to better understand their performance and further optimize the code. Initial experiments with Nek5000 and Quda have shown improved performance for non-blocking pipelined PCG solvers and reasonable accuracy consistent with the expected performance from previous studies.

REFERENCES

[1] Paul R. Eller and William Gropp. 2016. Scalable Non-blocking Preconditioned Conjugate Gradient Methods. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '16)*. IEEE Press, Piscataway, NJ, USA, Article 18, 12 pages. http://dl.acm.org/citation.cfm?id=3014904.3014928

[2] Paul R. Eller and William D. Gropp. 2018. Performance Modeling Scalable Krylov Solvers for Structured Grid Problems. *In preparation: TBD* (2018).