Reducing Network Contention Associated with Parallel Algebraic Multigrid

Amanda Bienz, Luke Olson (Advisor)
University of Illinois at Urbana-Champaign
Urbana, IL 61801

I. Introduction

Algebraic multigrid (AMG) is an iterative method often used to solve PDEs arising in various fields of science and engineering. The method is optimal, requiring only $\mathcal{O}(n)$ work to solve a system of n unknowns. Parallel implementations of AMG, however, lack scalability. As problem size increases, parallel AMG suffers from increasingly dense communication patterns, yielding network contention and reduced efficiency.

A. Parallel Algebraic Multigrid (AMG)

AMG consists of two phases: the setup phase, in which the hierarchy is constructed, and the iterative solve phase. When constructing some level l of the hierarchy, an interpolation matrix P_l is formed, associating strong relationships between the coarse and fine points. The coarse grid operator is then formed using the Galerkin product, $A_{l+1} = P_l^T A_l P_l$. This triple matrix produce yields fill-in on the coarse grid, which is then amplified throughout the hierarchy.

The linear system $Ax = b$ is then solved using two main methods. Initially, the high-frequency error in x is relaxed through a few iterations of a smoother. The resulting residual is then restricted to a coarser grid, transforming the remaining error back to high-energy. This process is repeated on successively coarser grids until the resulting problem is small enough to solve directly. This error is then interpolated up the hierarchy to the fine grid where it is added to the previous x. The entire process is repeated until convergence. This solve phase, which consists of relaxation, restriction, and interpolation, is composed mainly of sparse matrix-vector multiplies (SpMVs).

When computing a parallel SpMV, Ax, the operator is distributed across all active processors such that each contains a contiguous portion of the rows of A, as well as the equivalent rows of x. The local portion of A can be split into two blocks, as shown in Figure 1. The diagonal block contains columns of A that correspond to local x-values, while the off-diagonal block entries correspond to solution values stored on distant processors. The amount of communication required for a SpMV, and hence that required in each iteration of AMG, is directly related to the number of non-zeros in the off-diagonal block of A.

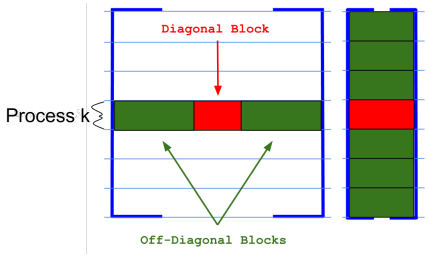

Fig. 1. The local portion of a matrix is split into a diagonal block, containing entries associated with local solution values, and off-diagonal blocks ,corresponding to distant solution values.

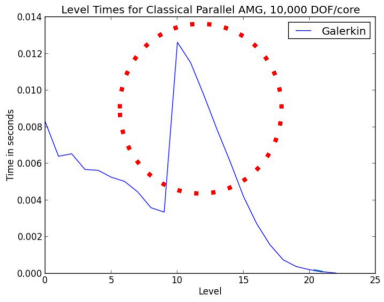

Fig. 2. Time spent on each level when solving a Poisson problem on Sierra, 256 nodes,

The fill resulting from the triple-matrix Galerkin product yields increasingly dense operators on successively coarser levels of the AMG hierarchy. Hence, the coarse levels have large communication requirements, increasing the time required to perform a SpMV. In some cases, relaxation on coarse levels near the middle of the hierarchy can take longer than on the original fine level, as shown in Figure 2

II. Reducing Network Contention

Large amounts of communication create network contention, which yields an increase in the cost associated with communication. Network contention occurs when a message needs to travel through a link that is already occupied with a separate message. The contention for a given network is influenced by the number of messages, the size of these messages, and the distance these messages must travel. To reduce network contention, any combination of these contributing factors can

This work was performed in collaboration with William Gropp (University of Illinois at Urbana-Champaign); Jacob Schroder and Rob Falgout (Lawrence Livermore National Laboratory)

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344, LLNL-POST-658223.

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant Number DGE-1144245.

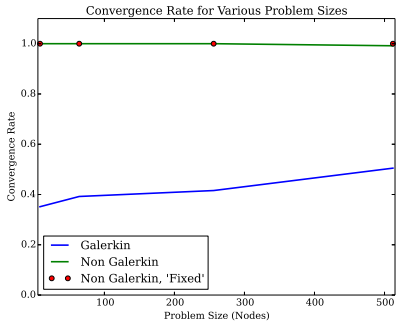

Fig. 3. Convergence of Galerkin AMG, a non-Galerkin method where significant entries have been removed, and a 'Fixed' non-Galerkin method where removed entries have been reintroduced to the matrix

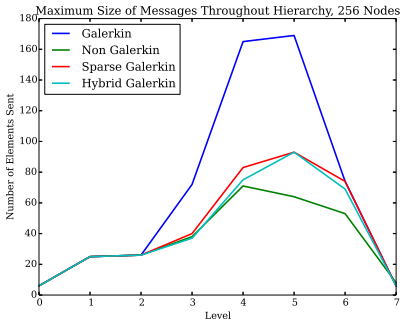

Fig. 4. The number of messages and size of these messages for solving a Poisson problem with 256 nodes, 10,000 dof/core via the alternative non-Galerkin approaches

be improved upon.

A. Non-Galerkin Coarse Grids

The total amount of communication can be reduced by removing fill from the coarse grid operators. The method of non-Galerkin coarse grids uses the Galerkin product to form coarse grids, and then filters the operator, removing relatively small entries [2]. Removing coarse-grid fill while maintaining a minimal sparsity pattern can significantly reduce the amount of communication required with little effect on convergence. The insignificant entries in a matrix vary with problem type. While it is possible to predict the communication cost associated with a given entry, it is not easy to determine its effect on convergence rate. During the solve phase, convergence can be improved upon by re-introducing entries into the matrix. However, removing an entry via non-Galerkin affects the sparsity of all successive coarse grids. So, adding entries back to a matrix does not guarantee an improvement in convergence. Figure 3 shows that if significant entries are removed from a coarse level, 'fixing' the method by reintroducing these entries does not necessarily improve convergence.

B. Alternative Non-Galerkin Methods

An alternative to non-Galerkin is to form the entire Galerkin hierarchy and then sparsity coarse grids via the method of non-Galerkin coarse grids. Since no coarse grids are dependent on the sparsified operators, re-introducing entries into the matrix guarantees to improve convergence. This approach, which safer, is potentially slower, as fewer entries can possibly be removed from coarse levels. There are two alternative approaches: sparse Galerkin, which uses the previous Galerkin level in the minimal sparsity pattern, and hybrid Galerkin, which uses the previously sparsified level in the minimal sparsity pattern. Figure 4 shows that, for a Poisson problem, the alternative approaches remove nearly as much communication as the original non-Galerkin approach. Further communication can be removed by relaxing the non-Galerkin requirements, such as the minimal sparsity pattern, because coarser grids no longer rely on the sparsified finer grid.

C. Topology-Aware Methods

Topology-aware methods can calculate the number of hops between any two processors [1]. This hop count can be used to estimate the communication cost associated with a given entry. Topology-aware methods can be applied to the alternative non-Galerkin approaches to target entries with high communication costs. Removing only messages with a large number of hops can reduce network contention nearly as much as removing all entries, as shown in Figure 5. These methods

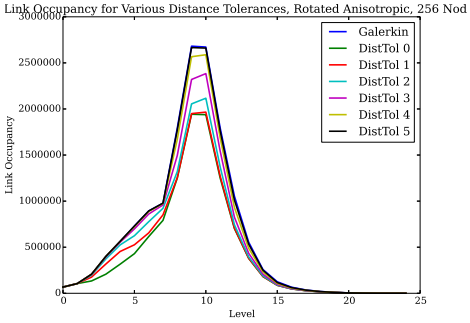

Fig. 5. Link occupancy (number of packets sent $\times$ distance they are sent) when alternative non-Galerkin is run on all entries greater than distance tolerance away

can also be used to remove a larger number of entries than originally removed in the non-Galerkin approaches. The drop tolerance can be increased on only long-distance messages, and elements in the minimal sparsity pattern can be removed if their associated communication costs are high. Furthermore, significant entries with large communication costs can be updated asynchronously with little affect on convergence.

REFERENCES

[1] Abhinav Bhatele, Eric Bohm, and Laxmikant V. Kale, *Optimizing communication for Charm++ applications by reducing network contention, concurrency, and computation* Practice and Experience (EuroPar special issue, Vol. 23, Issue 2

[2] R. D. Falgout and J. B. Schroder, *Non-Galerkin Coarse Grids for Algebraic Multigrid* SIAM J. Sci. Comput. (to appear)

[3] Hypre: *High performance preconditioners.* http://www.llnl.gov/CASC/hypre/.

[4] IBM Blue Gene Team, *The IBM Blue Gene/Q System* IBM, Somers NY, USA. www.hpc.cineca.it/sizes/default/files/bgq_description.pdf.

[5] J. W. Ruge and K. Stuben, *Multigrid Methods*, S.F. McCormick, ed., Frontiers Appl. Math., SIAM, Philadelphia, 1987