Reducing Communication Costs Associated with Parallel Algebraic Multigrid

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

I. Problem and Motivation

Algebraic multigrid (AMG) is an iterative method for solving sparse linear systems of equations ($A\hat{x} = b$), such as discretized partial differential equations arising in various fields of science and engineering. AMG is considered an optimal solver, requiring only $\mathcal{O}(n)$ operations to solve a system of n unknowns. Standard computers contain neither the memory nor computing power to solve increasingly large systems of equations, resulting in a demand for parallel solvers, in which the system is distributed and solved across a large number of nodes. Current implementations of parallel AMG lack scalability due to high costs of intranodal communication, yielding increasingly long solve times associated with large systems.

A. Parallel Algebraic Multigrid (AMG)

Algebraic multigrid consists of two phases: the setup phase and an iterative solve phase. During the setup phase, a hierarchy is created, containing successively coarser approximations to original operator. In constructing a coarse level, an interpolation operator P_l is created, associating strong relationships between the fine grid on level l and the coarse grid on level $l+1$. The coarse grid operator is constructed through the triple matrix Galerkin product $A_{l+1} = P_l^T A_l P_l$. This sparse triple matrix product yields fill in the coarse operator, which is then propagated through all remaining levels of the hierarchy. As a result, the coarse levels become increasingly dense, as displayed in Figure 1

During the solve phase, an initial guess x to the true solution $\hat{x}$ is constructed and then iteratively refined. On each level, l, in the hierarchy, high-frequency error is eliminated from the solution through relaxation, such as weighted Jacobi. The remaining low-energy error is restricted to a coarser level, $l+1$, transforming the largest modes of this error to high frequency. This new high-frequency error can again be removed through relaxation. On the coarsest level l_c, any remaining error can be solved for directly from $e_{l_c} = A_{l_c}^{-1} r_{l_c}$. This remaining coarse error is then interpolated back to the fine grid, where it is added to the initial solution. This entire v-cycle, displayed in Figure 1, is then repeated until convergence. Hence, the solve phase consists of relaxation, calculating the residual $r_l = b_l - A_l x_l$, restricting the residual $r_{l+1} = P_l^T r_l$, and interpolating the error $e_l = P_l e_{l+1}$. Therefore, this phase is comprised of sparse matrix vector multiplication (SpMVs).

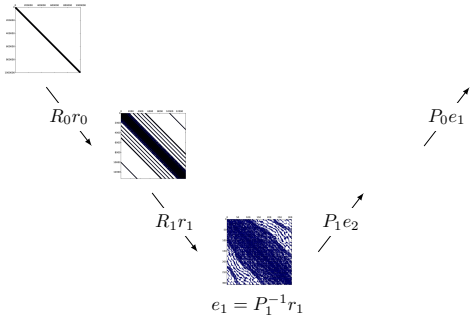

Fig. 1: AMG v-cycle

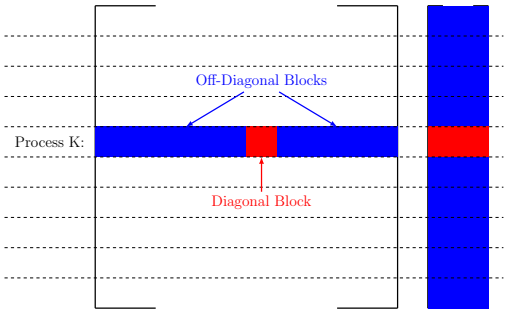

Fig. 2: System distributed across processors

The coarse grid fill acquired during the formation of each level yields relatively large, dense matrices near the middle of the hierarchy. The amount of work required to perform a SpMV, the main component of the solve phase, is $\mathcal{O}(nnz_l)$, where nnz_l is the number of nonzeros in A_l. However, there is an additional cost associated with communication when performing a parallel SpMV.

In parallel, the large system of equations $Ax = b$ is distributed across the processors such that every process holds a contiguous portion of the rows of A,

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

and the equivalent rows of both x and b. The matrix can be split into two blocks, as shown in Figure 2. The diagonal block contains columns of A corresponding to the values of x that are stored locally on the processor. The off-diagonal blocks, however, correspond to x-values that are stored on distant processors. Therefore, communication is required for all columns containing entries in the off-diagonal blocks of A. Hence, as the coarse levels of the hierarchy lose sparsity, communication requirements increase dramatically, yielding a large amount of time spent on levels near the middle of the hierarchy, as shown in Figure 3.

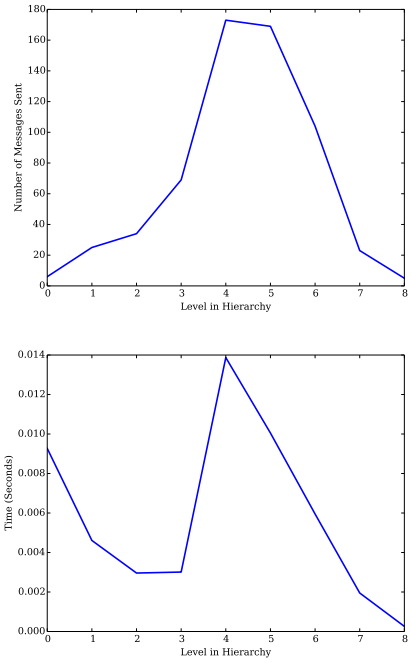

Fig. 3: Number of messages required for a single SpMV(top) and time spent on each level of the AMG hierarchy(bottom)

II. Background and Related Work

There has been a large amount of work targeted at reducing the communication costs associated with parallel AMG. Aggressive coarsening and distance-two interpolation, such as HMIS and PMIS , reduce the number of levels in the hierarchy by coarsening the problem more rapidly [6], [7]. However, these adjustments were applied to the problem being solved in Figure 3. Relatively large, dense grids remain near the middle of the hierarchy.

A. Non-Galerkin Coarse Grids

The method of non-Galerkin coarse grids [2] consists of forming each coarse matrix through the Galerkin product $A_{l+1} = P_l^T A_l P_l$ and then removing all insignificant fill. Relatively small entries are considered insignificant if they fall outside the minimal sparsity pattern $M = P_{I_l}^T A_l P_l + P_l^T A_l P_{I_l}$, where P_I is the injection operator. For a given drop tolerance γ, and entry is considered relatively small if $|a_{ij}| \leq \max_{k \neq i} |a_{ik}|$. If an entry a_{ij} is considered insignificant, its value is lumped to strong neighbors and the entry is eliminated($a_{ij} = 0$). Hence, sparsity is added back to each coarse level. This sparse approximation $\hat{A}_{l+1}$ is then used when constructing remaining levels of the hierarchy.

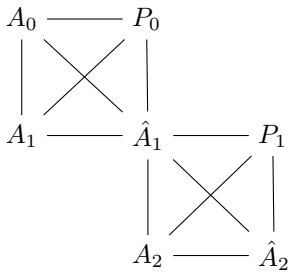

Fig. 4: Formation of non-Galerkin hierarchy

This non-Galerkin method can greatly reduce the amount of communication required on coarse levels with little affect on the convergence rate, yielding a significant improvement in time to solution, as shown in Figure 5. However, this figure displays that the reduction in solve time is dependent on choosing an appropriate drop tolerance γ, a variable that is problem dependent. If too small of a drop tolerance is chosen, little communication will be removed, and the solve time will approach that of the original Galerkin AMG. However, if γ is chosen to be too large, the convergence factor of the method will quickly approach 1.0, yielding an unconstrained increase in overall cost.

III. Uniqueness of the Approach

A unique alternative for reducing communication in parallel AMG, described in Section III-A consists of constructing the entire Galerkin hierarchy as usual and then adding sparsity back to coarse levels. Unlike previous approaches to reduce communication, this strategy leaves the structure of the hierarchy untouched. This allows a sparse approximation to be created in the setup phase in a lossless manner, allowing information to be reintroduced to the hierarchy during the solve phase.

While previous methods, such as the method of non-Galerkin coarse grids, have targeted removing communication with little effect on convergence, this method investigates the trade off between communication cost and convergence rate. Ideally, to minimize the overall costs of the method, any entry a_{ij} should be removed if the cost of retrieving the value x_j is greater than the entry's effect on convergence. As this method constructs the hierarchy normally, removing entries from coarse levels effectively introduces error into each SpMV in the solve phase. Therefore, the effect on convergence of removing an entry is proportional to the size of the

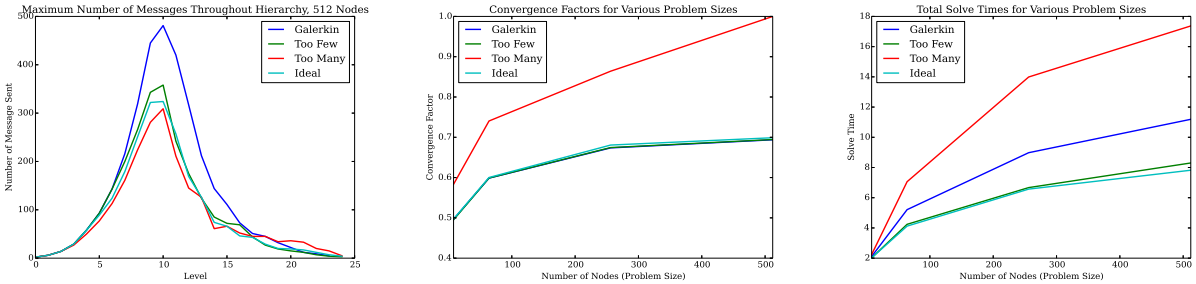

Fig. 5: The per-level communication costs(left), convergence rates for various problem sizes(middle), and solve times for various problem sizes(right), when dropping too few entries, too many entries, and the ideal number.

entry. Section IV-A considers all communication to have equal cost, similar to previous approaches, removing all entries below a given tolerance. Section IV-B improves upon this strategy by taking the topology of the network into account when determining the associated communication costs.

A. *Sparse Galerkin*

A variation to the method of non-Galerkin coarse grids, sparse Galerkin, creates the entire hierarchy before adding sparsity back to each coarse level. As described in Algorithm 1, relatively small entries are considered fill if they lie outside of the minimal sparsity pattern M. Any insignificant entry is removed, and its initial value is added to the diagonal. Figure 6 displays how sparse Galerkin is constructed in comparison to the original Galerkin hierarchy. Unlike the non-Galerkin dependency graph, shown in Figure 4, sparse Galerkin leaves the original Galerkin dependencies untouched.

Algorithm 1: sparsify

Input: A, P, S, A_c
Output: $\hat{A}_c$
$\mathcal{M} = P_I^T AP + P^T AP_I$
$\mathcal{N} = \emptyset$
$A_{sparse} = \emptyset$
for $A_{c_{ij}} \neq 0$ **do**
  if $\mathcal{M}_{ij} \neq 0$ *or* $|A_{c_{ij}}| > \gamma \; max_{k \neq i} |A_{c_{ik}}|$
    for $k < \#rows$ **do**
      $\mathcal{N}_{kj} = 1$
      $\mathcal{N}_{ki} = 1$
for $A_{c_{ij}} \neq 0$ **do**
  if $N_{ij} \neq 0$
    $\hat{A}_{c_{ij}} = A_{c_{ij}}$
  else
    $\hat{A}_{c_{ii}} = A_{c_{ij}}$

Sparse Galerkin is a lossless method for adding sparsity to coarse levels, retaining all information from the Galerkin hierarchy. The eliminated entries can be stored with little increase in overall storage, and adding these entries back to the matrix while subtracting their values from the diagonal would result in the original Galerkin hierarchy. Therefore, during the solve phase, if the convergence factor grows beyond a desired tolerance, some entries can be reintroduced into the hierarchy, effectively lowering the drop tolerance γ. Hence, the hierarchy can be edited during the initial iterations of the solve phase with little additional work to determine an optimal drop tolerance.

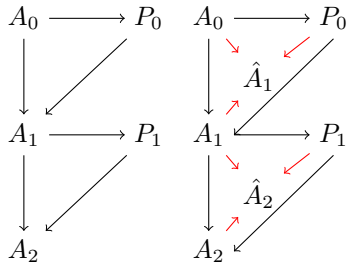

Fig. 6: Galerkin (left) vs. sparse Galerkin (right)

IV. Results and Contributions

My contributions to the paper include developing the alternative lossless approach for reducing communication costs in parallel AMG described in Section III-A; performing a study of the parallel performance of the lossless method in comparison the method of non-Galerkin coarse grids in Section IV-A; and improving the prediction of associated communication costs through the use of previously existing topology-aware methods, as described in Section IV-B.

A. *Sparse Galerkin Results*

The sparse Galerkin method investigates the trade-off between communication costs and convergence rates, assuming all messages have equal costs. Therefore, insignificant entries of A are removed only if any communication will be reduced as a result. Hence, the diagonal block remains unchanged, as all operations involving these entries are local. Furthermore, an entry in the off-diagonal block a_{ij} is only removed if every other element in column j can also be eliminated.

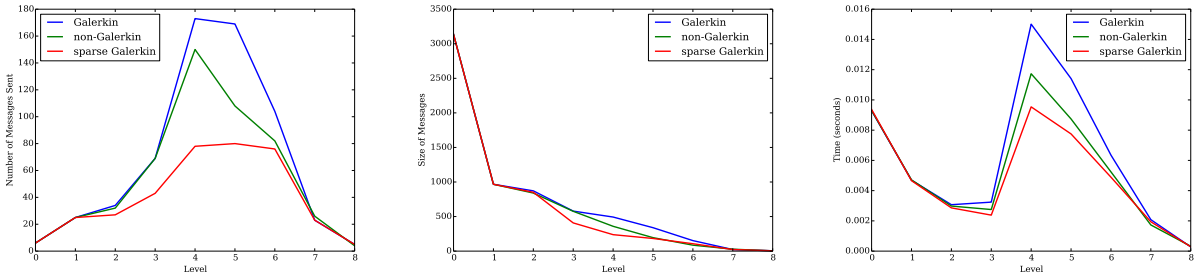

Fig. 7: The number of messages required for a SpMV on each level(left), size of these messages(middle), and time spent on each level of the AMG hierarchy(right) when using the various methods to solve a 3D Laplace problem on 8192 processors.

A weak scaling study of this method was performed for both a three-dimensional Laplace problem and a two-dimensional rotated anisotropic problem, with $\epsilon = 0.001$ and $\theta = \frac{\pi}{8}$. The problems were tested on 128, 1024, 4096, and 8192 processors of Lawrence Livermore National Laboratory's parallel computer Vulcan, with problem sizes equal to 10,000 degrees-of-freedom per process. Figure 7 shows the reduction in both communication required by a single SpMV as well as time spent on each level of the hierarchy for the Laplace problem on 8192 processors. Figure 8 shows both the setup and solve times required for solving Laplace problems of the various sizes. Figure 9 displays the setup and solve times for the weak scaling study of the rotated anisotropic diffusion problem. These tests indicate that sparse Galerkin can significantly reduce the solve time with little additional work in the setup phase. The per-level communication can be significantly reduced with little change to the convergence factor for ideal drop tolerances. However, if other drop tolerances are initially chosen, the method can be adjusted to approach these optimal drop tolerances in the first few iterations of the solve phase.

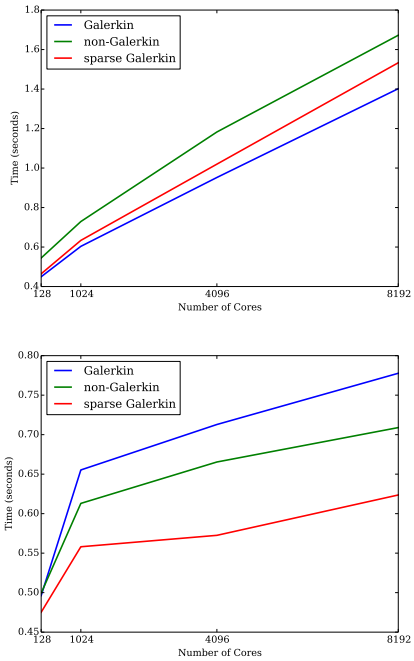

Fig. 8: The time spent in the setup phase(top) and solve phase(bottom) when solving a 3D Laplace problem of various problem sizes

B. *Topology-Aware Sparse Galerkin*

Topology-aware methods calculate the hop count between any two ranks; the number of network links that a message must traverse to get from one processor to the other [1]. With the use of existing topology-aware methods, the communication costs associated with entries of A can be adjusted based on the topology of the network.

Sparse Galerkin assumes that all communication has equivalent costs associated with it. However, the cost of internodal communication is negligible. Therefore, any change in convergence due to the removal of messages between two processes on a single node decreases performance. Figure 10 shows the solve times when only intranodal communication is removed for the Laplace and anisotropic problems. For these tests, internodal entries had little to no effect on convergence. However, as they also have negligible associated communication costs, retaining internodal communication results in nearly equivalent solve times.

This method can be further improved by considering that the costs associated with intranodal messages vary. The cost of a message depends on both the message size, or number of packets sent, and the distance these packets must travel. Furthermore, the most inhibitive costs associated with communication result from contention in the network. Network contention occurs when a message needs to traverse a given link that is being used by a separate message. All messages which must traverse this link will sit idle until the it becomes free.

There are relatively small improvements in communication costs when removing messages that traverse short distances. Messages that are sent to neighboring

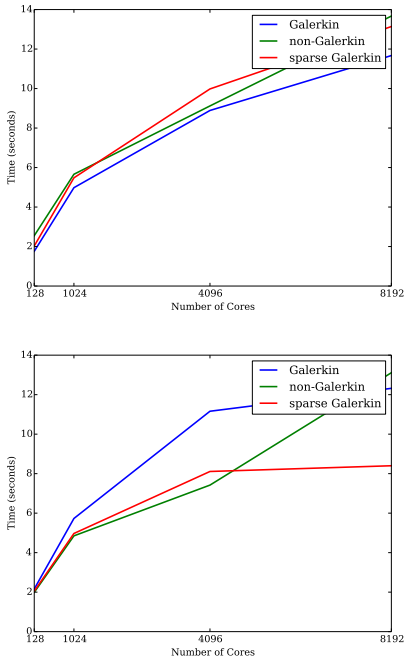

Fig. 9: The time spent in the setup phase(top) and solve phase(bottom) when solving a 2D rotated anisotropic diffusion problem of various problem sizes

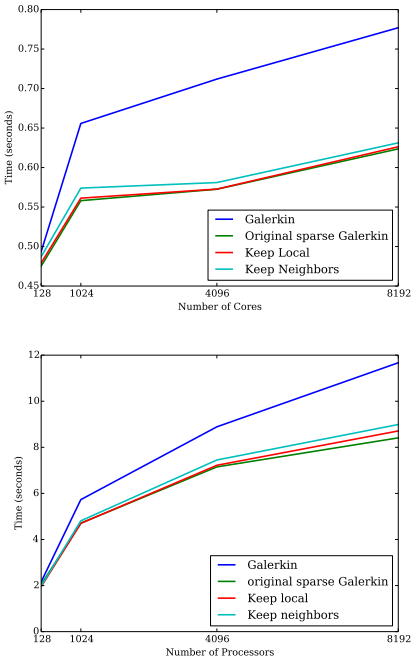

Fig. 10: Solve times for 3D Laplace (top) and 2D rotated anisotropic diffusion (bottom) problems, with topology-aware adjustments.

nodes must traverse only a single link. While the cost associated with communication between neighbors is not negligible, it is significantly cheaper than longer distance communication. As only one link is traversed, a message path does not need to be computed, and, hence, latency is minimized. Furthermore, network contention has little effect on this message cost as only one link is required. Figure 10 shows the solve times for a variation of sparse Galerkin in which all messages to immediate neighbors are retained. There is a slight increase in the overall solve time, due to the cost associated with communication between neighboring nodes. However, the increase in solve time is very slight, indicating that these messages have a relatively small cost of communication.

C. Future Work

This trade-off between communication costs and convergence rates should be investigated further. With topology-aware methods, the cost of communicating each message can be accurately predicted through performance models. Costly messages should have a larger drop tolerance associated with them, while small messages traveling short distances should only be dropped if the associated operator values are significantly small. Furthermore, removing elements from large, long distance messages only reduces the communication cost if the number of packets being sent is reduced. The method should more accurately predict the benefits in removing various types of communication, and adjust the drop tolerances for entries based on this associated cost.

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. 36 (2014), pp.C309-C334.

[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.

[6] H. De Sterck, R. D. Falgout, J. Nolting, and U. M. Yang, *Distance-Two Interpolation for Parallel Algebraic Multigrid* Numerical Linear Algebra with Applications 15 (2008) 115139.

[7] U. M. Yang, *On Long-Range Interpolation Operators for Aggressive Coarsening* Numerical Linear Algebra with Applications 17 (2010) 453-472.

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.