Reducing Communication Costs in the Parallel Algebraic Multigrid

Amanda Bienz, Luke N. Olson (advisor), William D. Gropp (mentor)
University of Illinois at Urbana-Champaign
{bienz2, lukeo, wgropp}@illinois.edu

***Abstract*—Algebraic multigrid is an optimal sparse linear solver that suffers from poor parallel scalability due to large communication requirements. This costly communication can be improved through changes to both the algorithm and parallel implementation. The amount of data required to be communicated can be reduced through sparsification of the matrices, yielding slower convergence but significantly cheaper iterations, and therefore, an overall speedup. Furthermore, the data that must be communicated can be sent between processes with increased efficiency through the use of topology-aware methods. The number of messages injected into the network can be reduced greatly at the cost of additional intra-node communication, yielding cheaper communication on coarse levels of AMG.**

I. Problem Statement

Algebraic multigrid (AMG) is an iterative method for solving sparse linear systems, such as discretized elliptic partial differential equations, arising from various fields of science and engineering. As parallel computers advance, there is potential to solve increasingly large problems with more accuracy. Furthermore, systems which are currently too expensive to solve can be spread across an increased number of processes, yielding the potential to find a solution at a reasonable cost. However, large communication costs associated with AMG prevent both weak scalability, such as solving larger problems efficiently, as well as the strong scalability required to solve current systems on a larger number of processes.

AMG consists of two phases: a setup phase, in which a hierarchy is created containing successively coarser approximations to the original system, and a solve phase, which iteratively updates an initial guess until convergence. Figure 1

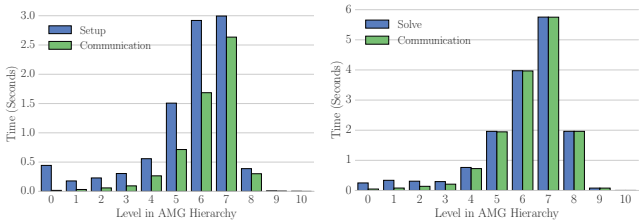

Fig. 1. The total time vs time required for communication during setup (left) and solve (right) of a 2D rotated anisotropic diffusion hierarchy with Hypre on 8192 cores of Blue Waters, with the fine level containing 10 000 degrees of freedom per core.

profiles the cost of the dominant operations of each phase for a rotated anisotropic diffusion hierarchy created and solved with Hypre [2] on 8192 processes of Blue Waters [1]. These profiles illustrate an increase in both setup and solve costs on the coarse levels over the original, larger, fine level due to an increase in communication requirements.

The efficiency and scalability of AMG can be greatly improved by reducing the cost of communication, specifically on coarse levels. This can be accomplished by reducing the communication costs associated with any dominant operation. In particular, these costs can be reduced by communicating less data or re-routing this data to incur fewer costs when sending a message.

II. Research Highlights

The remainder of this paper presents thesis highlights, namely methods for reducing the cost of communication in AMG and improving scalability through alterations to both the underlying algorithm and the parallel implementation. A variation of the standard Galerkin AMG algorithm is presented in Section II-A, in which less data is communicated during the solve phase, resulting in slower convergence but less costly iterations, resulting in overall speedup [4]. Section II-B investigates a method of communicating with more efficiency on coarse levels of AMG, with a performance model for analyzing the various costs of standard communication in Section II-B1, a description of the new communication method in Section II-B2, and experimental results of the effect on various dominant operations of AMG in Section II-B3 [5]. Finally, current work and future directions are discussed briefly in Section III.

A. Altering Algorithm with Sparsification

The cost of the AMG solve phase is dominated by communication in sparse matrix-vector multiplies (SpMVs), particularly on coarse levels near the middle of the hierarchy. The increase in communication on coarse levels results from an increase in number of nonzeros per row, as each coarse level is formed through a sparse triple matrix product. However, many of the nonzeros resulting from fill-in during matrix multiplication are significantly small in magnitude and have little influence on matrix operations. Therefore, these entries can often be removed with little effect on convergence.

Falgout et al.'s method of non-Galerkin coarse grids [6] systematically removes entries deemed insignificant. If a value at (i, j) falls outside some required sparsity pattern M and has magnitude smaller than some tolerance γ times the maximum off-diagonal magnitude in row i, then the value is removed and lumped to neighbors. Therefore, each removed entry

contributes to a reduction in communication. The entries are removed immediately after creating a coarse level, influencing all remaining levels in the hierarchy. While this method has potential to remove a large per-iteration cost, more than offsetting any reduction in convergence rate, the effectiveness is dependent on a drop tolerance that can vary per problem as well as per level of the hierarchy. A small tolerance will result in little change to per-iteration costs. However, a large drop tolerance can result in the removal of significant entries, yielding a method that may fail to converge. Furthermore, as successively coarse levels are created after the removal of small entries, reintroducing previously removed entries may not recover convergence.

Each coarse level of the AMG hierarchy can instead be sparsified after the formation of the entire Galerkin AMG hierarchy, yielding coarse levels that do not depend on previously sparsified operators. Therefore, after the hierarchy is created, all values that meet the previously described criteria are removed. However, as this removal no longer affects other operators, it is sufficient to lump removed values to the diagonal rather than all strong neighbors, allowing for entries with no strong connections to be removed. There are two variations of this method: sparse Galerkin, in which the pattern M on each level ℓ is based on the finer original operator $A_{\ell-1}$, and hybrid Galerkin, in which M is based on the previous sparsified operator $\hat{A}_\ell$. Figure 2 compares the dependencies of these methods against the standard Galerkin method and the previously described non-Galerkin. Figure 3

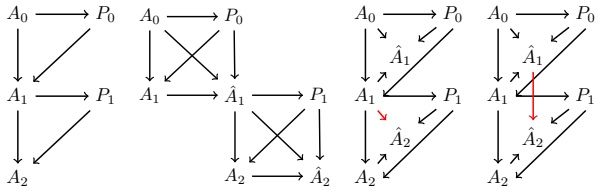

Fig. 2. The dependencies of each coarse (and sparsified coarse) level for the methods, from left to right: Galerkin, non-Galerkin, sparse Galerkin, and hybrid Galerkin.

displays the reduction in solve time over the standard Galerkin AMG for each of these approaches when solving a 2D rotated anisotropic diffusion problem. For these results, five series of drop tolerances were tested and the best results for each method were chosen.

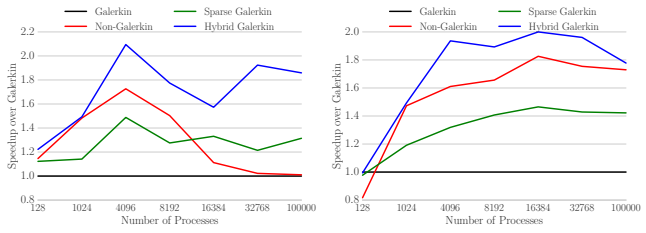

Fig. 3. Speedup over Galerkin when solving weakly (left) and strongly (right) scaled 2D rotated anisotropic diffusion problems with the various AMG methods.

As with non-Galerkin, the sparse and hybrid Galerkin methods depend on drop tolerances that can vary with each matrix. However, as the Galerkin hierarchy is formed before sparsification, removed entries can be retained, and an adaptive solve phase can reintroduce entries as needed. Therefore, if per-iteration cost remains high, a larger drop tolerance can be selected. If too many entries are removed and convergence suffers, entries can be reintroduced, first to the finest levels and then to coarser operators. Figure 4 shows an example of adaptively solving by first selecting too large of a drop tolerance and slowly reintroducing entries as needed. The per-iteration cost remains lower than Galerkin as convergence is achieved.

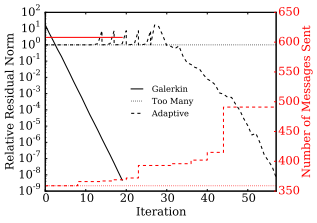

Fig. 4. The relative residual (black) vs number of messages sent (red) at each iteration of the solve phase for a 2D rotated anisotropic diffusion problem on 8192 cores. The solid line shows the typical Galerkin reduction in residual, with 600 messages sent by a single process each iteration. The light dotted line shows that the lack of convergence when dropping too many entries. The bold dotted line shows the adaptive solve, during which entries are added back while convergence suffers.

B. Modifying Implementation with Topology-Awareness

State-of-the-art supercomputers consist of a large number of multicore nodes connected through a network, such as Blue Waters, a Cray machine consisting of 22 636 XE nodes, each containing 16 Bulldozer cores, and connected through a 3D torus interconnect. The cost of communicating messages varies with the locations of the sending and receiving processes, as on-socket communication is significantly cheaper than sending across multiple links of the network. However, standard communication algorithms send messages between two processes regardless of their location in the network. The remainder of this section will focus on methods for improving communication in AMG on Blue Waters, but can be extended to other topologies.

1) Performance Models: The cost of communication during a parallel operation can be profiled through performance modeling. These models can be used to analyze the source of large costs during communication. Performance modeling is an important first step in optimizing the efficiency of a parallel implementation, as it indicates where the implementation should be altered to reduce costs. A common performance model, the alpha-beta model, breaks the communication cost into per-message latency, or start-up costs, and per-byte transport costs. This model can be used to determine if the cost of communication is dominated by the number or size of messages.

In modern parallel computers, such as Blue Waters, communication depends not only on the number and size of messages being communicated, but also on a variety of other factors, such as the locations of the send and receive processes, the

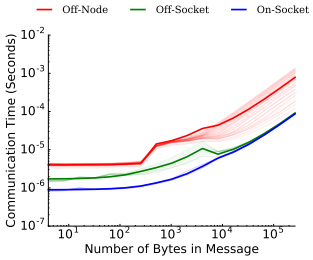

Fig. 5. The measured times (light lines) to communicate a message between two processes and corresponding alpha-beta models (bold lines), for processes on the same socket, on different sockets of the same node, or on different nodes.

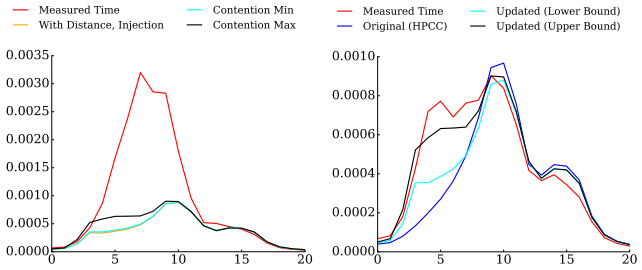

Fig. 6. Updated alpha-beta models, with additional penalties for injection bandwidth and network contention (left) do not accurately match measured times for communication during a SpMV on each level the 2D rotated anisotropic diffusion AMG hierarchy. However, when a small portion of messages are sent at a time, network contention and queue search times are reduced, and the model matches the measured times (right).

size of messages a single network interface controller (NIC) must inject into the network, and the number of bytes passing through the network at any given time. The standard alpha-beta model can be improved through the use of topology-aware methods, which map a virtual process rank to a physical core in the system. Figure 5 shows both the measured costs and associated alpha-beta models when different measures for latency and bandwidth are used for on-socket, on-node, and off-node communication, illustrating a drastic difference in cost of on-node and off-node communication. Therefore, the alpha-beta model can be updated to contain multiple alpha and beta parameters to cover each combination of on-node and off-node with short, eager, and rendezvous sending protocols.

Additional per-byte transport penalties should be added to the model when a large number of bytes are injected into the network from multiple processes sharing a single NIC, as injection bandwidth limits the speed of transport, and when a large number of bytes are injected into the network from multiple NICs, as the network becomes contended as multiple packets must traverse a single link at once. The number of bytes to be injected into the network by a single NIC can be calculated by summing the cumulative size of inter-node messages on each process local to a node. If there are enough bytes being injected by multiple processes, it can be assumed that injection bandwidth is reached. However, penalizing network contention is more difficult, as packets have a large number of possible paths to traverse the network. Therefore, transport costs can be penalized based on the average number of bytes to traverse any link, yielding a lower bound, or and upper bound based on the maximum possible number of bytes to traverse any link.

Figure 6 displays the measured cost of performing the communication required during a SpMV on each level of the 2D rotated anisotropic diffusion hierarchy in comparison to the original alpha-beta model and the model updated with penalties. The models do not capture all costs near the middle of the hierarchy, due to inaccuracies in network contention parameters and the lack of modeling queue search times. However, if only $\frac{1}{20}^{th}$ of the messages are communicated at a given time, reducing both network contention and queue search times, the models accurately capture the cost of communicating during each SpMV.

These models indicate that inter-node communication is significantly more costly than intra-node, as Figure 5 yielded a difference in alpha and beta parameters that is only amplified when injection bandwidth and network contention are taken into account. Therefore, inter-node communication should be limited, and replaced with intra-node communication when possible.

2) Topology-Aware Parallel Communication: Communication on coarse levels near the middle of the AMG hierarchy can be improved through the use of topology-awareness, trading inter-node communication for an increase in intra-node messages. Standard communication on these coarse levels requires communication of a large number of inter-node messages. When using multiple processes per node, there are likely multiple messages being sent between the same two nodes. Furthermore, there is likely duplicate data being sent between two nodes, as the same values may need to be communicated to two processes which lie on the same node. Figure 7 displays a standard communication process between 8 processes split across two nodes. All processes that must send data to process q send it directly, regardless of their physical locations.

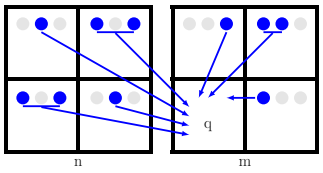

Fig. 7. The standard process for sending data to some rank q.

Topology-aware parallel (TAP) communication aggregates data on-node to reduce the number and size of messages injected into the network. Figure 8 displays the process in which data is sent from node n to m. Any necessary data is first sent to process p, local to node n, before being sent as a single message to process q on node m. Process q then sends this data to appropriate processes local to node m. It is important to note that this figure only displays a portion of the communication. It is assumed that there are many nodes active in communication, and all processes on node n are communicating inter-node messages to other nodes not shown in the figure. Therefore, this method outperforms alternatives such as the combination of MPI and OpenMP with a single

process per node, as TAP communication allows all processes to communicate at once, rather than having a master process performing all communication.

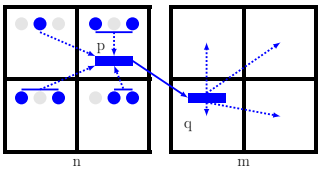

Fig. 8. An updated topology-aware method for communication between any processes on two nodes.

3) Topology-Aware Parallel Matrix Operations: Topology-aware parallel (TAP) communication can be applied to the dominant matrix operations of AMG: sparse matrix-vector multiplication (SpMV) and sparse matrix-sparse matrix multiplication (SpGEMM), replacing the standard communication step in each operation with TAP communication. Figure 9 shows that TAP communication requires a large increase in intra-node communication but a reduced number of inter-node messages when performing a TAPSpMV on each level of an unstructured MFEM linear elasticity hierarchy on 8192 processes [3]. There is a comparable changes to the number of bytes communicated, with an increase in the size of intra-node messages and decrease in number of bytes injected into the network. Furthermore, TAPSpGEMMs result in a similar increase in number and size of intra-node messages, while reducing the number and size of messages injected into the network.

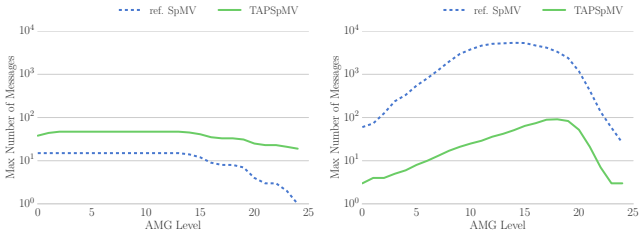

Fig. 9. The number of intra- (left) and inter-node (right) messages when performing a SpMV at each level of a linear elasticity hierarchy.

As a result, the total cost of performing a SpMV or SpGEMM on coarse levels of a linear elasticity hierarchy can be greatly reduced with the use of TAP communication, as shown in Figure 10. Furthermore, the TAP communication can greatly extend the strong scalability of these operations, with the cost of performing a SpMV on each level of the hierarchy at various process counts displayed in Figure 11.

III. Current and Future Work

The topology-aware performance modeling is ongoing research, as the current model does not accurately capture network contention and queue search times. Furthermore, this model is being updated to match the max-rate model more accurately [7]. The topology-aware communication research is being extended to the various operations in AMG, such as communication during the formation of prolongation operators. RAPtor, an algebraic multigrid solver comparable in performance to Hypre, is in the process of being developed and will allow for topology-aware communication throughout. In the future, TAP communication could be extended to other solvers, and be used to optimize AMG as a preconditioner, influencing Krylov subspace solvers when preconditioned by AMG.

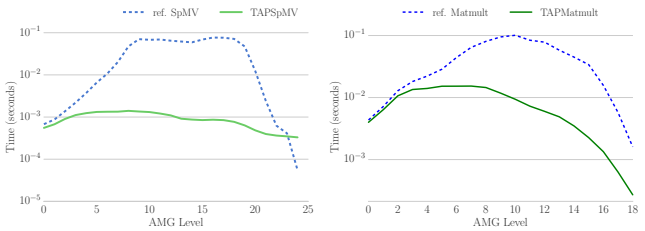

Fig. 10. The time required to perform a SpMV (left) and SpGEMM (right) at each level of the linear elasticity hierarchy.

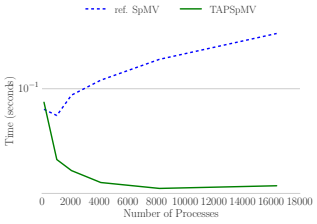

Fig. 11. The time required to perform a SpMV at every level of the strongly scaled linear elasticity hierarchy at various process counts.

References

[1] Blue Waters. https://bluewaters.ncsa.illinois.edu/.
[2] HYPRE: High performance preconditioners. http://www.llnl.gov/CASC/hypre/.
[3] MFEM: Modular finite element methods. mfem.org.
[4] A. Bienz, R. D. Falgout, W. Gropp, L. N. Olson, and J. B. Schroder. Reducing parallel communication in algebraic multigrid through sparsification. *SIAM Journal on Scientific Computing*, 38(5):S332–S357, 2016.
[5] A. Bienz, W. D. Gropp, and L. N. Olson. Tapspmv: Topology-aware parallel sparse matrix vector multiplication. *(submitted)*.
[6] R. D. Falgout and J. B. Schroder. Non-Galerkin coarse grids for algebraic multigrid. *SIAM Journal on Scientific Computing*, 36(3):C309–C334, 2014.
[7] W. Gropp, L. N. Olson, and P. Samfass. Modeling MPI communication performance on SMP nodes: Is it time to retire the ping pong test. In *Proceedings of the 23rd European MPI Users' Group Meeting*, EuroMPI 2016, pages 41–50, New York, NY, USA, 2016. ACM.