Reducing Communication Costs in the Parallel SpMV

Amanda Bienz, Luke Olson, and William D. Gropp
University of Illinois at Urbana-Champaign
201 N. Goodwin Ave.
Urbana, Illinois 61801, USA
{bienz2, lukeo, wgropp}@illinois.edu

ABSTRACT

Sparse matrix-vector multiplication is a dominant operation in many linear solvers. Parallel SpMVs are often dominated by communcication costs. This cost an be reduced by grouping messaged together based on network topology, reducing both the number of messages and bytes that traverse the network.

1. INTRODUCTION

On-going advances in parallel computing yield a potential to solve increasingly large problems with greater efficiency. The sparse matrix-vector multiply (SpMV) is a fundamental component of many high-performance computing methods, such as sparse linear solvers. In parallel, communication requirements often dominate the cost of each SpMV, resulting in poor scalability.

State-of-the-art high performance computers commonly consists of nodes, each containing many processes. Figure 1 shows a simple system containing three nodes with two processes each.

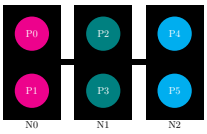

Figure 1: Simple parallel system of three nodes, each containing two processes

The cost of communication between processes consists of a start-up cost for each message as well as the cost associated with transporting a single byte. These costs, however, vary based on the endpoints of each message. For instance, intra-node communication is significantly less costly and inter-node messages. The differences between inter- and intra-node communication of both large and small messages are displayed in Table 1.

	Intra (small)	Intra (large)	Inter (small)	Inter (large)
Start-up	7.759e-07	2.439e-06	2.577e-06	7.623e-06
Per-Byte	4.090e-09	7.890e-10	9.751e-09	1.971e-09

Table 1: Communication costs for both large and small messages

2. BACKGROUND

Parallel sparse linear systems are often distributed across processes in a contiguous, row-wise partition, as displayed in Figure 2. Each process holds a local portion of the matrix, which contains both entries corresponding to vector values stored on node (represented as solid blocks) as well as non-zeros associated with portions of the vector that are stored on distant nodes (represented as dashed blocks). Therefore, matrices with a large number of non-zeros outside of the local band correspond to large inter-node communication requirements.

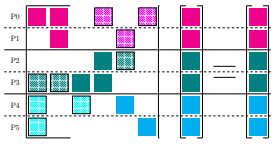

Figure 2: Matrix and vectors partitioned across parallel system in Figure 1

Standard SpMVs consist of sending necessary vector values directly to each process that needs them. For example, in the system displayed in Figure 2, P0 would send vector values to P3, P4, and P5 and receive vector values from P1, P3, and P5, as shown in Figure 3. Assuming the processes are partitioned across the nodes as in Figure 1, process P0 is sending two messages to processes on node N2. Furthermore, as the same vector values are sent to both processes, duplicate data is being sent across the network.

3. REDUCING INTER-NODE COMMUNICATION IN SPMV

Figure 3: Original communication, displaying process P0's sends (left) and receives (right) for sample system in Figure 2

Intra-node communication is much cheaper than inter-node communication, as shown through the parameters in Table 3. Inter-node communication can yield further delays due to injection bandwidth limits and network contention. Therefore, reducing inter-node communication requirements can greatly reduce the overall cost of communication, even at the penalty of additional intra-node communication.

When performing a SpMV, the cost of communicating vector values can be reduced by limiting the number and size of messages sent between nodes. This can be achieved by first gathering necessary values from all intra-node processes with the same destination node. These values are then all sent to a process on the destination node, where they are distributed to processes local to that node.

For example, when performing a SpMV on the example system in Figure 2, the values needed by node N0, v_4 and v_5, are gathered on process P4, as shown in Figure 4. Next, the collected messages are sent across the network to process P0, as shown in Figure 5. Finally, the received value v_4 is distributed to process P1, displayed in Figure 6.

Figure 4: Initial step: gathering intra-node communication

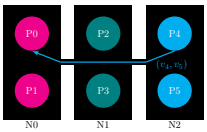

Figure 5: Middle step: sending data across network

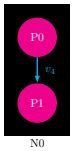

Figure 6: Final step: distributing received values through node

4. EXPERIMENTAL RESULTS

The SpMV with reduced inter-node communication was tested on each level of a linear elasicity algebraic multigrid (AMG) hierarchy, created with both MFEM and Hypre [2, 1], due to a large variety of communication patterns found in AMG hierarchies. An AMG hierarchy consists of successively coarser systems approximating the low-energy error of the original system. The finer levels require a small number of large messages to be communicated, while many coarse levels require a large number of small messages. The SpMV was also tested on a subset of 11 of the 15 largest 'real' matrices from the Florida Sparse Matrix Collection [3]. The four largest matrices were not tested due to the larger I/O overhead.

There was a significant decrease in both the number and size of messages sent across the network, while the intra-node communication increased greatly, as shown in Figure 7. The time required to perform a SpMV was reduced significantly, often by an order of maginitude for coarse levels of the AMG hierarchies. Figure 8 shows the time required to perform a SpMV on each level of various the AMG hierarchies, while Figure 9 shows the reduction in SpMV times for various UFL matrices.

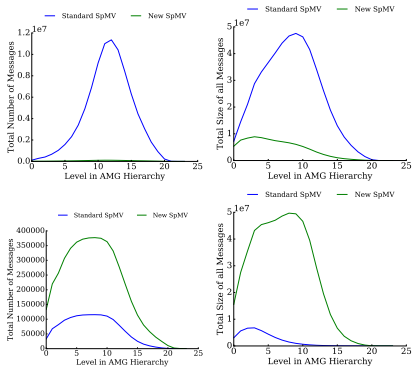

Figure 7: The total number (left) and size (right) of both inter- (top) and intra-node (bottom) communication for the AMG hierarchy for MFEM [2] linear elasiticy

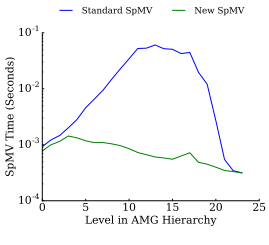

Figure 8: Time required for each level of the AMG hierarchy for MFEM [2] linear elasticity.

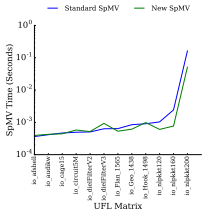

Figure 9: Time required to perform a SpMV on various UFL matrices.

Acknowledgment

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

5. REFERENCES

[1] HYPRE: High performance preconditioners. http://www.llnl.gov/CASC/hypre/.

[2] MFEM: Modular finite element methods. mfem.org.

[3] T. A. Davis and Y. Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38:1:1 – 1:25, 2011.