**LETTER**

**Axis Communication Method for Algebraic Multigrid Solver**

**SUMMARY**

Communication costs have become a performance bottleneck in many applications, and are a big issue for high-performance computing on massively parallel machines. This paper proposes a halo exchange method for unstructured sparse matrix vector products within the algebraic multigrid method, and evaluate it on a supercomputer with mesh/torus network. In our numerical tests with a Poisson problem, the proposed method accelerates the linear solver more than 14 times with 23,040 cores.

**key words:** strong scaling, linear solver, Algebraic multigrid method

1. Introduction and Background

This paper reports the effectiveness of algebraic multi-grid (AMG) method [1],[2], which uses “axis communication” on a supercomputer with mesh/torus network.

AMG is one of the most efficient linear solvers for many applications. Its calculation complexity is $O(n)$, where $n$ is the number of unknowns. If $n$ is very large, the problem is usually very time consuming to solve directly, so AMG creates coarser levels with smaller matrix equations that resemble the original problem matrix equation. This phase is called setup. Then, the iterative phase solves the problem using the multi-level structure. This implies matrices at each level and between levels; the coarser level matrices tend to become denser and more unstructured than the matrix on the finer level. When a massively parallel machine is used, the coarser level matrices are distributed to each node, and the communication cost often becomes a performance bottleneck.

Sparse matrix vector products with each level and inter-level matrices form the majority of the AMG algorithm. Sparse matrix vector products are often based on one dimensional block row distribution; our AMG routines also use this distribution for all matrices. Peer to peer direct communication is usually used for halo exchange. This paper proposes a new communication method for AMG that improves its scaling performance and effectiveness.

Peer to peer communication costs can be modeled as a sum of latency and $\text{messagesize}/\text{bandwidth}$ times. Sparse matrix vector product has relatively a small halo in many cases but all processes must exchange their halo at the same time. Therefore, latency tends to become more important than message size or bandwidth. In particular, when one process is connected to many processes via its halo, it must receive small messages from many processes. In this case, the latency cost seems to worsen the halo communication performance severely. In addition, if the network is congested, the effect of message collision cannot be ignored. Therefore, we propose axis communication for the AMG method in order to reduce latency and message collision effects.

Axis communication assumes a 3D topology of processes. It aggregates messages sent from neighboring ranks and then transfers them along each axis in order. Although it uses multiple peer to peer communication sessions to ensure that a message arrives at its destination, it reduces the number of messages on the network. It can limit the number of messages a process receives from other processes at the same time, regardless of halo structure. That limit number becomes the number of nodes on the same axis.

Since the AMG solver has halo communications at all levels and between levels with sparse matrix vector product, the halo structures differ from one another. Therefore, we propose to incorporate axis communication into AMG so that it can switch the communication routines.

This contribution is organized as follows. Section 2 describes axis communication, and Sect. 3 explains how to incorporate axis communication to AMG solver. Section 4 evaluates the AMG solver with axis communication then conclusions are summarized in Sect. 5.

2. Axis Communication

This section describes the axis communication, which is a method to exchange halo data among processes. This method sends and receives messages along each axis in order, based on the process topology defined by the user. This process topology is not necessarily the same as the one the computing environment offers. The fastest topology for axis communication may be different from the one the environment specifies because each process may have different amount of communication data. Some processes may have no data to send and receive. Here, a process topology for axis communication is explained at first, then communication procedure is described.

This paper considers a process topology for axis communication as 3-D torus. The user can specify the number of processes a node has and the number of nodes in each
X,Y,Z-direction. Therefore, the topology is specified by 4 numbers as C, X, Y, Z. For example, the topology specified by C = 2, X = 2, Y = 3, Z = 4 means 48 processes connected 3-D torus with 2 nodes in X direction, 3 nodes in Y, 4 nodes in Z, and each node has 2 processes. Process ranks are assumed to be sequential in the same node; the ranks increase in the order of X, Y, Z-direction. Therefore, X,Y,Z-coordinates in the topology can be determined from the process rank r as (r/C)%X, (r/C)/X%Y, (r/C)/(X*Y), respectively.

The communication steps are listed as follows.

1. Each node determines one process that communicates messages between nodes, and that process collects all sent messages from the node.
2. The collected messages are sorted and aggregated by its destination process rank’s X-coordinate.
3. Sorted and aggregated messages of each node are exchanged only in X-direction.
4. Received messages are sorted and exchanged in Y-direction in the same way as step 2, 3
5. Received messages are sorted and exchanged in Z-direction the same way as step 2, 3
6. Received messages are distributed to destination processes in the node.

In our implementation, the process with the lowest rank in the node manages communication between nodes.

By recording the message sizes, destination process ranks and message buffer shuffling patterns beforehand, the 0 sized messages can be omitted and all operations come to buffer copying and send receive communications.

This communication method contains multi-step procedure and it may cause more delay than usual peer to peer direct communication, but it can prevent that many processes send messages to one process at the same time, which cause serious performance degradation. In addition, the messages are aggregated, so the network is less congested.

3. AMG Solver with Axis Communication

Axis communication is a method for exchange of halo data. The parallel AMG solver possesses halo information for each level’s problem matrix and for inter level operations. After each level is created in the setup phase, the solver tests the performance of the communication methods among usual peer to peer communication and axis communication with various process topologies. Given a set of topology combinations (e.g. as in Table 1, third column), the fastest communication pattern is determined for each halo exchange and it is used until the end of iterative phase. This corresponds to an autotuning feature of the library.

For example, if the AMG solver has 3 levels, then 3 communication tables for halo exchanges are created for all levels, and 2 communication tables for inter-level operation. For each communication table, all process topologies (C,X,Y,Z) are checked, and the best topology for axis communication is set. This means that the communication routine setting may become different for each communication table.

4. Numerical Test and Evaluation

4.1 Test Problem Setting

This paper compares our AMG solver (AMGS) [3] with axis communication, AMGS with usual peer to peer communication routine, and AMG-BICGSTAB solver of PETSc-gang (version 3.4.3) [4]. PETSc is a widely used library, and this paper uses its performance as a baseline. AMGS employs a smoothed aggregation based algebraic multigrid method [1]. PETSc-gang parameters are set as default. Although one might be able to get a better performance with a clever choice of those parameters, the default gives us a baseline for comparison. For consistency, the parameters for AMGS are the same in all test cases.

The problems are two 3 dimensional Poisson equation. The first one (referred to as 1st problem) is a straightforward isotropic Poisson equation and the second one (referred to as 2nd problem) is a ground water flow problem through heterogeneous porous media. This problem leads to a Poisson equation with varying heterogeneous diffusion coefficients from $10^{-5}$ to $10^{5}$.

The problem equations are input to the solvers as a sparse matrix equation, and AMG computes the coarser levels. Therefore, the coarser level matrices and inter level matrices become unstructured. The problem domain is cubic with $200 \times 200 \times 200$ vertexes. The problem matrix is distributed to each process using ParMETIS [5].

In order to consider the scenario where communication cost is increasingly higher, this test is done with flat MPI with a fixed problem size and increasing number of processes (strong scaling).

The machine used in our experiments is a Fujitsu FX10 Supercomputer System at the University of Tokyo [6]. Each node has a SPARC64 IXFx processor with 16 cores. It allows a user to specify 3-D torus process topology. We specified 3-D topologies of 16-core nodes and topologies for axis communication as shown in Table 1. Topologies for axis communication (C,X,Y,Z) can have arbitrary values, if $C*X*Y*Z$=number of processes. Here, we pick up 3 to 5 topologies like the topology of nodes the system manages.

Table 1 Nodes, cores, and topology setting.

| Cores | Topology of nodes | Topology for axis comm. |
|-------|-------------------|------------------------|
| 32    | 2 × 1 × 1         | (16,2,1,1) (4,8,1,1)   |
| 128   | 2 × 2 × 2         | (16,2,2,2) (4,8,2,2)   |
| 192   | 2 × 3 × 2         | (16,2,3,2) (4,8,3,2)   |
| 1536  | 4 × 6 × 4         | (16,4,6,4) (4,16,6,4)  |
| 3072  | 4 × 6 × 8         | (16,4,6,8) (4,16,6,8)  |

The parallel AMG solver possesses halo information for

\(C, X, Y, Z\)-direction. Therefore, the topology is specified by 4

\(C, X, Y, Z\)-direction. Therefore, the topology is specified by 4

numbers as \(C, X, Y, Z\). For example, the topology specified

by \(C = 2, X = 2, Y = 3, Z = 4\) means 48 processes

connected 3-D torus with 2 nodes in X direction, 3 nodes in

Y, 4 nodes in Z, and each node has 2 processes. Process

ranks are assumed to be sequential in the same node; the

ranks increase in the order of X, Y, Z-direction. Therefore,

\(X, Y, Z\)-coordinates in the topology can be determined from

the process rank \(r\) as \((r/C)\%X, (r/C)/X\%Y, (r/C)/(X*Y)\),

respectively.

The communication steps are listed as follows.

1. Each node determines one process that communicates

messages between nodes, and that process collects all

sent messages from the node.
2. The collected messages are sorted and aggregated by

its destination process rank’s X-coordinate.
3. Sorted and aggregated messages of each node are

exchanged only in X-direction.
4. Received messages are sorted and exchanged in

Y-direction in the same way as step 2, 3
5. Received messages are sorted and exchanged in

Z-direction the same way as step 2, 3
6. Received messages are distributed to destination

processes in the node.

In our implementation, the process with the lowest rank in

the node manages communication between nodes.

By recording the message sizes, destination process

ranks and message buffer shuffling patterns beforehand, the

0 sized messages can be omitted and all operations come to

buffer copying and send receive communications.

This communication method contains multi-step procedure and it may cause more delay than usual peer to peer direct communication, but it can prevent that many processes send messages to one process at the same time, which cause serious performance degradation. In addition, the messages are aggregated, so the network is less congested.

3. AMG Solver with Axis Communication

Axis communication is a method for exchange of halo data. The parallel AMG solver possesses halo information for each level’s problem matrix and for inter level operations. After each level is created in the setup phase, the solver tests the performance of the communication methods among usual peer to peer communication and axis communication with various process topologies. Given a set of topology combinations (e.g. as in Table 1, third column), the fastest communication pattern is determined for each halo exchange and it is used until the end of iterative phase. This corresponds to an autotuning feature of the library.

For example, if the AMG solver has 3 levels, then 3 communication tables for halo exchanges are created for all levels, and 2 communication tables for inter-level operation. For each communication table, all process topologies \((C,X,Y,Z)\) are checked, and the best topology for axis communication is set. This means that the communication routine setting may become different for each communication table.

4. Numerical Test and Evaluation

4.1 Test Problem Setting

This paper compares our AMG solver (AMGS) [3] with axis communication, AMGS with usual peer to peer communication routine, and AMG-BICGSTAB solver of PETSc-gang (version 3.4.3) [4]. PETSc is a widely used library, and this paper uses its performance as a baseline. AMGS employs a smoothed aggregation based algebraic multigrid method [1]. PETSc-gang parameters are set as default. Although one might be able to get a better performance with a clever choice of those parameters, the default gives us a baseline for comparison. For consistency, the parameters for AMGS are the same in all test cases.

The problems are two 3 dimensional Poisson equation. The first one (referred to as 1st problem) is a straightforward isotropic Poisson equation and the second one (referred to as 2nd problem) is a ground water flow problem through heterogeneous porous media. This problem leads to a Poisson equation with varying heterogeneous diffusion coefficients from $10^{-5}$ to $10^{5}$.

The problem equations are input to the solvers as a sparse matrix equation, and AMG computes the coarser levels. Therefore, the coarser level matrices and inter level matrices become unstructured. The problem domain is cubic with $200 \times 200 \times 200$ vertexes. The problem matrix is distributed to each process using ParMETIS [5].

In order to consider the scenario where communication cost is increasingly higher, this test is done with flat MPI with a fixed problem size and increasing number of processes (strong scaling).

The machine used in our experiments is a Fujitsu FX10 Supercomputer System at the University of Tokyo [6]. Each node has a SPARC64 IXFx processor with 16 cores. It allows a user to specify 3-D torus process topology. We specified 3-D topologies of 16-core nodes and topologies for axis communication as shown in Table 1. Topologies for axis communication \((C,X,Y,Z)\) can have arbitrary values, if \((C*X*Y*Z)=\)number of processes. Here, we pick up 3 to 5 topologies like the topology of nodes the system manages.

Table 1 Nodes, cores, and topology setting.

| Cores | Topology of nodes | Topology for axis comm. |
|-------|-------------------|------------------------|
| 32    | 2 × 1 × 1         | (16,2,1,1) (4,8,1,1)   |
| 128   | 2 × 2 × 2         | (16,2,2,2) (4,8,2,2)   |
| 192   | 2 × 3 × 2         | (16,2,3,2) (4,8,3,2)   |
| 1536  | 4 × 6 × 4         | (16,4,6,4) (4,16,6,4)  |
| 3072  | 4 × 6 × 8         | (16,4,6,8) (4,16,6,8)  |
4.2 Test Results

Figure 1, 2 show the performance result for AMGS with axis communication, AMGS without axis communication, and PETSc-gamg solver. The times do not include the distribution of the matrix. Here, we pay attention to how the total time increases with the parallelism.

PETSc and AMGS without axis communication solver exhibit performance degradation as the number of processes increases. On the other hand, performance degradation of AMGS with axis communication is less than those of other solvers. PETSc-gamg has a functionality which controls a degree of parallelism by redistributing the coarser level matrix. AMGS does not have such a functionality. This is the reason why PETSc-gamg had better performance than AMGS original version in Fig. 1. Table 2 shows the time for the setup phase and iterative phase for each problem setting. The setup phase has some overhead for checking various axis communication topology. The iterative phase with axis communication uses optimized communication routine among usual peer to peer communication routine and topology based communication routines. From this table, the performance of iterative phase of AMGS is improved by 1.6 and 2.8 times for the 1st and 2nd problems in the case of 1536 cores.

**Table 2** AMG solver time breakdown for each number of processes setup phase time[sec] and iterative phase time[sec]. “AMGS w axis” means AMGS with axis comm. and “AMGS w/o axis” means AMGS without axis comm. Level num. and Iteration num. show the number of levels and the iteration number of AMG-BiCGSTAB for convergence, respectively.

| Solvers       | 32     | 128    | 192    | 1536   | 3072   |
|---------------|--------|--------|--------|--------|--------|
| 1st problem   |        |        |        |        |        |
| AMGS w axis   | 6.2/22.8 | 1.8/6.0 | 1.3/3.9 | 1.9/2.8 | 2.6/3.4 |
| AMGS w/o axis | 6.1/23.0 | 1.7/5.9 | 1.2/3.9 | 1.7/4.7 | 2.2/18.5 |
| Level num.    | 4      | 4      | 4      | 4      | 4      |
| Iteration num.| 11     | 10     | 9      | 11     | 12     |
| 2nd problem   |        |        |        |        |        |
| AMGS w axis   | 8.2/20.1 | 2.3/6.5 | 1.7/4.1 | 2.8/1.1 | 4.5/1.1 |
| AMGS w/o axis | 8.1/20.0 | 2.2/6.9 | 1.6/4.4 | 2.5/3.1 | 4.0/4.2 |
| Level num.    | 5      | 5      | 5      | 5      | 5      |
| Iteration num.| 17     | 19     | 17     | 19     | 20     |

Table 3 has max numbers of neighboring processes from one process at all levels or between levels. They correspond to communication tables for smoothers at each level or ones for inter-level operations. From Table 3, the number of neighboring processes from one process becomes as large as 3007 for inter-level operation for 1st problem. It causes the performance degradation in the case of 3072 cores for 1st problem.

Our solver calculates the solution at the coarsest level by one process. Therefore the solution vector must be gathered to one process at the coarsest level. Thus, the number of neighboring processes becomes very large for inter-level operations, especially when problem matrix is distributed to many processes. From Table 2, the number of levels for 1st problem is less than that for 2nd problem. This leads to more distributed matrices at the second coarsest level for 1st problem than for 2nd problem. Thus max number of neighboring processes between levels for 1st problem is larger than that of 2nd problem. And the performance degradation for 1st problem becomes very large.

There are 2 approaches to deal with this increasing neighboring communication. First one is to redistribute the matrices for coarser levels in order to reduce the neighboring messages. However, this case needs an overhead to redistribute the matrices. Second one is to prepare communication routines like collective communication, which can deal with many neighboring communication efficiently. This paper took second approach and proposed axis communication method. It turned out to improve the performance very well from Fig. 1 and Fig. 2.

Next we consider the effectiveness of axis communication. Table 2 includes ten tests from 32 cores to 3072 cores for 2 problem cases, which contains 74 different halo
communications. In order to compare ordinary peer to peer communication routine and axis communication routine, we measure the communication time using MPI_Barrier in the test code. Figure 3 shows the improvement rates (ordinary peer to peer communication time/axis communication time) plotted on the max number of neighboring processes from one process in the halo communication. Here, the axis communication time corresponds to the fastest one among various topologies for axis communication. If the ratio is higher than 1.0, axis communication is faster than ordinary communication. From this figure, when max neighboring processes are more than 44, axis communication is always faster than ordinary communication. For the test cases where both numbers in Table 3 are more than 100, axis communication clearly shortens the time of iterative phase.

In the end, we tested 2nd Poisson problem with the domain size of 300 x 300 x 300 using up to 23040 cores (1440 nodes) to evaluate AMGS with the axis communication for a problem with larger domain size on more and more cores. The solver was executed by flat MPI model in the same way as Fig. 1 and Fig. 2. Figure 4 shows the result. In this figure, AMGS with axis comm. is always fastest among the three solvers. Axis communication turns out to improve the strong scaling performance very much.

5. Conclusion and Future Work

AMG usually creates coarser levels that are more un-structured than finer levels, and halo communication of the coarser level and inter-level operation becomes more complex than that of the finer level. Especially for strong scaling test, the communication costs become bigger, resulting in an overall performance degradation as the number of cores increases.

This paper described an axis communication method and incorporated it into an AMG library that can switch the communication strategy between ordinary routine and axis communication routine. Axis communication can limit the number of neighboring processes in halo communication. Therefore, it is effective especially when there are many neighboring processes from one process with small sized messages in the communication. In addition, ordinary peer to peer communication is also checked to minimize the communication time, and this implementation method will reduce the communication time with any sized problems. In numerical tests, this technique shortened the communication time and improved the strong scaling performance. In the test with 23040 cores, the total time is accelerated by up to 14 times.

In our experiments, axis communication works more effectively than ordinary peer to peer communication for halo exchange with more than 44 neighboring processes from one process. This paper investigated axis communication effectiveness on a supercomputer with mesh/torus network. In the near future, we plan to study its effectiveness on a systems with other types of network.

Acknowledgements

This research was supported by JST, CREST.

References

[1] P. Vanka, J. Mandel, and M. Brezina, “Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems,” Computing, vol.56, no.3, pp.179–196, 1996.
[2] A. Baker, R. Falgout, T. Kolev, and U. Yang, “Scaling hypre’s multigrid solvers to 100,000 cores,” in High-Performance Scientific Computing, ed. M.W. Berry, K.A. Gallivan, E. Gallopoulos, A. Grama, B. Philippe, Y. Saad, and F. Saied, pp.261–279, Springer London, 2012.
[3] A. Fujii, A. Nishida, and Y. Oyanagi, “Evaluation of parallel aggregate creation orders: Smoothed aggregation algebraic multigrid method,” in High Performance Computational Science and Engineering, ed. M. Ng, A. Doncescu, L. Yang, and T. Leng, IFIP — The International Federation for Information Processing, vol.172, pp.99–122, Springer US, 2005.
[4] S. Balay, M.F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, K. Rupp, B.F. Smith, and H. Zhang, “PETSc users manual,” Tech. Rep. ANL-95/11 - Revision 3.4, Argonne National Laboratory, 2013.
[5] G. Karypis and V. Kumar, “MeTiS: Unstructured graph partitioning and sparse matrix ordering system, Version 4.0,” http://www.cs.umn.edu/~metis, 2009.
[6] Information Technology Center, The University of Tokyo.