Speedup and Numerical Evaluation of Multiple-precision Krylov Subspace Method using GPU Cluster for Large-Sparse Linear System

Yuta Hirokawa*
School of Computer Science,
Tokyo University of Technology
Hachioji 192-0982, Japan
hirokawa@nal.ikulab.org

Taku Itoh
School of Computer Science,
Tokyo University of Technology
Hachioji 192-0982, Japan

Hiroto Tadano
Faculty of Engineering,
Information and Systems,
University of Tsukuba
Tsukuba 305-8577, Japan

Soichiro Ikuno
School of Computer Science,
Tokyo University of Technology
Hachioji 192-0982, Japan

ABSTRACT

Implementing Multiple-precision Krylov subspace method on GPU cluster for a large-sparse linear system is investigated, and the method is numerically evaluated. It is well known that a number of iteration of Krylov subspace method depend on accumulation errors, and the error may affect the calculation results. In order to settle these issues, the multiple precision operation Krylov subspace method is implemented on GPU cluster using GNU Multiple Precision Arithmetic Library (GMP) and CUDA Multiple Precision Arithmetic Library (CUMP), and the method is parallelized to get high performance. The result of computation shows that the variable preconditioned Bi-CGSTAB on GPU cluster is up to 16.38 times faster than that of CPU with OpenMP.

Keywords

GPU, Multiple-precision, Krylov Subspace Method, Variable Preconditioning, HA-PACS

1. INTRODUCTION

It is well known that a number of iteration of Krylov subspace method depend on accumulation errors, and the error may affect the calculation results. One of the settlement of these issues is the multiple precision operation, and libraries of the multiple precision operation have been developed. However, the multiple precision operation takes much execution time and transmission time.

*Yuta Hirokawa insisted his name be first.

GNU Multiple Precision Arithmetic Library (GMP) and The CUDA Multiple Precision Arithmetic Library (CUMP) are libraries for implementing multiple-precision arithmetic, and CUMP is developed for CUDA [3]. In addition, cooperate calculation is able to use in both libraries.

The purpose of the present study is to implement a code of multiple precision Krylov subspace method using GPU cluster of HA-PACS at Center for Computational Science, University of Tsukuba, and to evaluate the performance of the numerical code.

2. VP KRYLOV SUBSPACE METHOD

The variable preconditioned generalized conjugate residual (VPGCR) method has two nested iterations for GCR and variable preconditioning for GCR are called as outer-loop and inner-loop [2]. The variable preconditioned (VP) GCR method has the sufficient condition for convergence. The residual of the problem converges if the relative residual norm of inner-loop satisfies the inequality in each steps. That is to say, residual equation can be solved roughly by using some iterative method with only a few iteration. Thus, VP Krylov subspace method has an advantage of parallelization compare with conventional preconditioning strategy such as Incomplete Cholesky decomposition or LU decomposition.

By taking into account of above character, we propose variable preconditioned method with mixed precision that uses double precision operation for inner-loop and multiple precision operation for outer-loop [1]. Besides, we extend the algorithm of variable preconditioned method using Bi-CGSTAB method for outer-loop. The algorithm of variable preconditioned Bi-CGSTAB (VPBi-CGSTAB) method is shown in Fig. 1. In the present study, JOR method is adopted for inner-loop solver.

3. EVALUATION

In the present study, HA-PACS at Center for Computational Science, University of Tsukuba is used for evaluation. HA-PACS is GPU cluster, and the peak performance is 802 TFLOPS. In the present study, 32 nodes are used for

Let $\boldsymbol{x}_0$ be an initial guess.
$\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
$\boldsymbol{p}_0 = \boldsymbol{r}_0^* = \boldsymbol{r}_0$
for $k = 0, 1, ...,$ until $||\boldsymbol{r}||_2/||\boldsymbol{b}||_2 \leq \varepsilon$
  Roughly solve $A\boldsymbol{z}_k = \boldsymbol{p}_k$ using some iterative method
  $\alpha_k = \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)}{(\boldsymbol{r}_0^*, A\boldsymbol{z}_k)}$
  $\boldsymbol{t}_k = \boldsymbol{r}_k - \alpha_k A\boldsymbol{z}_k$
  Roughly solve $A\boldsymbol{w}_k = \boldsymbol{t}_k$ using some iterative method
  $\gamma_k = \dfrac{(A\boldsymbol{w}_k, \boldsymbol{t}_k)}{(A\boldsymbol{w}_k, A\boldsymbol{w}_k)}$
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{z}_k + \gamma_k \boldsymbol{w}_k$
  $\boldsymbol{r}_{k+1} = \boldsymbol{t}_k - \gamma_k A\boldsymbol{w}_k$
  $\beta_k = \dfrac{\alpha_k}{\gamma_k} \dfrac{(\boldsymbol{r}_0^*, \boldsymbol{r}_{k+1})}{(\boldsymbol{r}_0^*, \boldsymbol{r}_k)}$
  $\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k(\boldsymbol{p}_k - \gamma_k A\boldsymbol{z}_k)$
end for

Figure 1: The algorithm of VPBi-CGSTAB.

evaluation (about 100 TFLOPS). In addition, MPI is used for communication between each Rank, and OpenMP and CUDA is used for the parallelization in each Rank. When CUMP is used for calculation the overhead of the communication between GPUs is larger than the communication between Ranks using MPI because memory layout must be transformed. For this reason, in case of parallelization with CPU, unit Rank per node is adopted, and unit Rank per GPU is adopted for GPU parallelization.

The electromagnetic field analysis of a linear problem is discretized by using FEM with edge element, and the coefficient matrix becomes a singular matrix. The dimension size of the matrix is $N = 1,709,028$. Besides, the coefficient matrix becomes very sparse and symmetric matrix because an edge element is used for discretization. Only 42 nonzero elements include in unit column. The sparseness of the coefficient matrix is 99.99% (number of non-zero element is 27,549,822).

In Figure 2, we show the communication time of quadruple precisioned VPBi-CGSTAB for various Rank allocation. We can see from this figure that the communication time in case of unit GPU assigned for unit Rank decrease as the number of GPU increase. Because asynchronous communication can be implemented in case of GPU/RANK= 1, and the amount of data communication using PCI express in unit node becomes relatively larger than between the nodes communication. On the other hand, asynchronous communication cannot be implemented in case of GPU/RANK= 4.

The speedup of octuple precisioned VPBi-CGSTAB is shown in Figure 3. This figure indicates that VPBi-CGSTAB on GPU cluster is up to 16.38 times faster than that of OpenMP.

4. CONCLUSIONS

We have been developed the parallelized code of VPBi-CGSTAB method and the code has been evaluated on HA-PACS GPU cluster. Conclusions obtained in the present study are summarized as follows.

The communication time and calculation time of multiple precision operation increase drastically. Result of computation showed that the mixed precisioned Krylov subspace method could be reduced the communication time. In this sense, mixed precisioned Krylov subspace method is effective

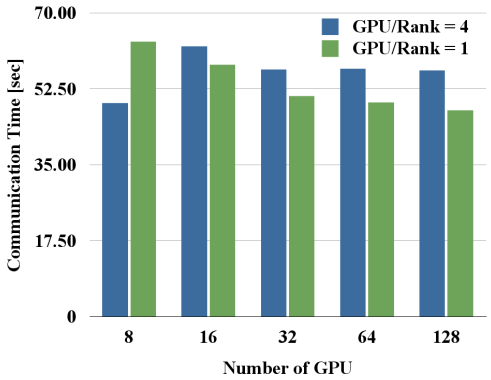

Figure 2: The communication time of quadruple precisioned VPBi-CGSTAB for various Rank allocation.

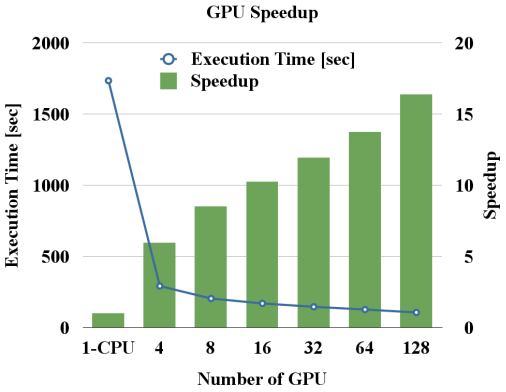

Figure 3: The speedup of octuple precisioned VPBi-CGSTAB.

solver for the large-sparse linear system.

5. ACKNOWLEDGMENTS

Numerical calculations for the present work have been carried out under the "Interdisciplinary Computational Science Program" in Center for Computational Sciences, University of Tsukuba.

6. REFERENCES

[1] S. Ikuno, Y. Kawaguchi, N. Fujita, T. Itoh, S. Nakata, and K. Watanabe. Iterative Solver for Linear System Obtained by Edge Element: Variable Preconditioned Method With Mixed Precision on GPU. *IEEE Transactions on Magnetics*, 48:467–480, 2012.

[2] K. Abe and S. L. Zhang. A variable preconditioning using the SOR method for GCR-like methods. *Int.J.Number.Anal Model.2*, (2):118–128, 2002.

[3] T. Nakayama and D. Takahashi. Implementation of Multiple-Precision Floating-Point Arithmetic Library for GPU Computing. *Proc. 23rd IASTED International Conference on Parallel and Ditributed Computing and Systems (PDCS 2011)*, pages 343–349, 2011.