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

Yuta Hirokawa[1,a] Taku Itoh[1] Hiroto Tadano[2] and Soichiro Ikuno[1]

1: School of Computer Science, Tokyo University of Technology, Hachioji, Tokyo 192-0982, Japan
2: Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba, Ibaraki 305-8577, Japan
a: hirokawa@nal.ikulab.org

1. Introductions

Krylov subspace method is a effective solver for a large-sparse linear system.

The character of Krylov subspace method
1. Easy to parallelize because the method is constituted by addition of vectors, inner product and multiplication of matrices and vectors.
2. Iteration number depend on accumulation of errors.

↓

The multiple precision can lead us reducing the accumulation errors and iteration number of Krylov subspace method.

Multiple precision operation
1. Any size of multiple precision can be used
2. It takes much execution time for the operation and the transmission. Thus, there are many libraries of multiple precision operation for CPU and GPU to get high performance calculation

↓

GPU cluster must be needed for the large scale simulation because unit GPU has a few amount of memory.

Purpose of the present study is to develop 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. Multiple-precision Arithmetic on GPU

We adopt following libraries for implementing multiple-precision arithmetic on Krylov subspace method.
- CPU: GMP [1]
  (GNU Multiple Precision Arithmetic Library)
- GPU: CUMP
  (CUDA Multiple Precision Arithmetic Library)

CUMP and GMP are libraries for implementing multiple-precision arithmetic, and CUMP is developed for CUDA. In addition, cooperate calculation is able to use in both libraries [2].

3. Evaluation linear system

The electromagnetic field analysis of a linear problem is discretized by using FEM with edge element, and the coefficient matrix becomes a singular matrix [3].

The dimension size of the matrix is N = 1,709,028. Besides, the coefficient matrix becomes very sparse and symmetric matrix, if 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).

4. Standard Bi-CGSTAB

The multiple-precision Bi-CGSTAB is effective for the linear system obtained by FEM with edge element. (Fig. 1)

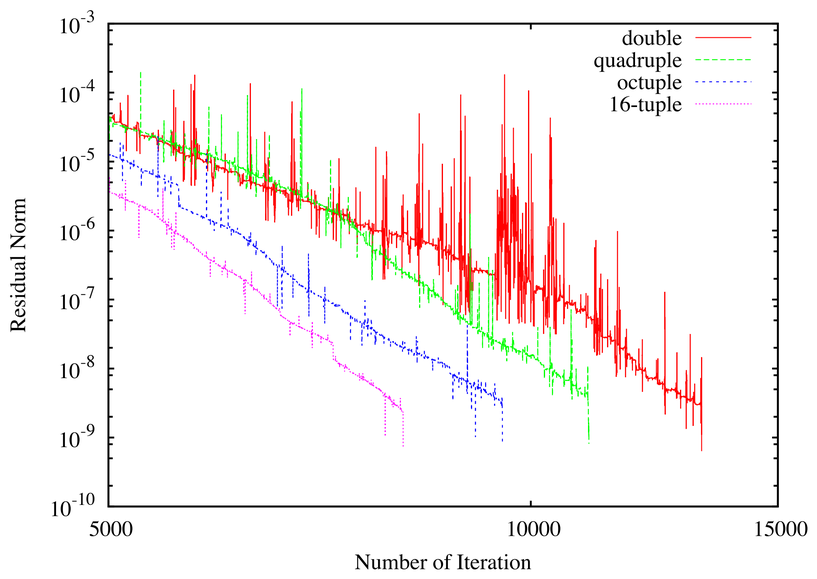

Fig. 1: The residual histories of Bi-CGSTAB.

5. Variable Preconditioned Krylov subspace method

Variable Preconditioned Krylov subspace method [3]
1. Some iterative method can be applied for variable preconditioning.
2. It is easy to parallelized, and easy to get high performance.

↓

Applicable iterative method for can be selected. (Fig. 2)

Convergence Theorem

In VPGCR, the residual of the problem converges if the relative residual norm of inner-loop satisfies following inequality in each steps. [4]

$$\frac{||\boldsymbol{r}_{k+1} - A\boldsymbol{z}_{k+1}||_2}{||\boldsymbol{r}_{k+1}||_2} < 1$$

We adopt variable preconditioned Krylov subspace method with mixed precision that uses double precision operation for inner-loop and multiple precision operation for outer-loop, and we extend the algorithm of variable preconditioned method using various Krylov subspace method for outer-loop.

JOR method is adopted for inner-loop because of stabilized residual decreasing.

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 $k = 0, 1, \ldots,$ 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

Fig. 2: The algorithm of VPBi-CGSTAB

6. Parallelization

HA-PACS at Center for Computational Science, University of Tsukuba is used for evaluation [5]. HA-PACS is GPU cluster, and the peak performance is 802 TFLOPS. In the present study, 32 nodes are used for evaluation. (about 100 TFLOPS, see Table. 1)

Table 1: Evaluation Environment

CPU	Intel E5 2.6GHz x2
CPU Memory	128GB
CPU FLOPS	332.8GFLOPS/Node
GPU	NVIDIA M2090 x4
GPU Memory	24GB/Node
GPU FLOPS	2660GFLOPS/Node
Infiniband	Mellanox Connect-X3 dual head
CUDA	4.2.9
GMP	5.0.5
CUMP	1.0.1
MPI	MVAPICH2 1.8.1
CPU Compiler	icc 13.0 (-O3 -march=native)
GPU Compiler	nvcc 4.2 (-O3 -arch=sm_20)

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. (see Fig 3 and 4)

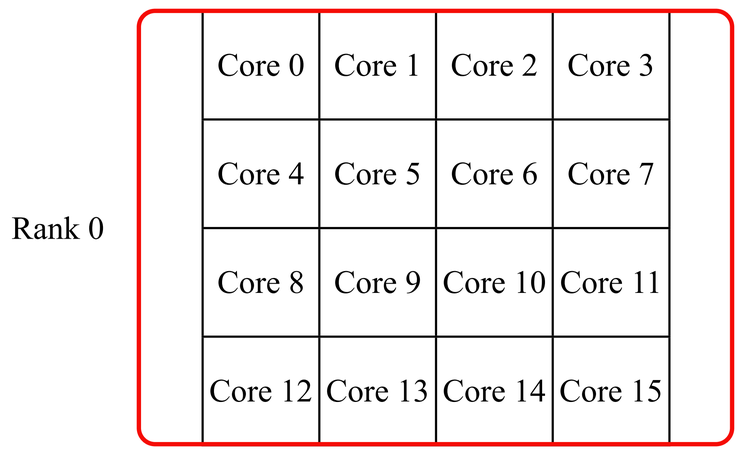

Fig. 3: The schematic view of the parallelization with CPU in unit node.

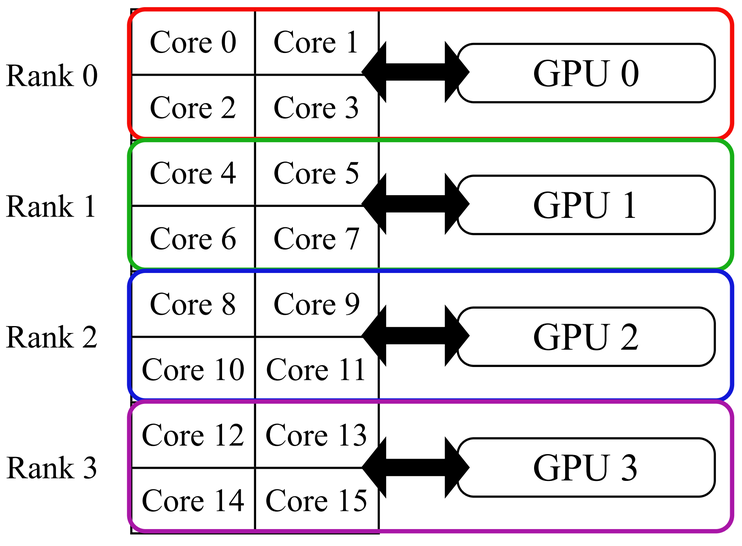

Fig. 4: The schematic view of the parallelization with GPU in unit node.

7. Evaluation

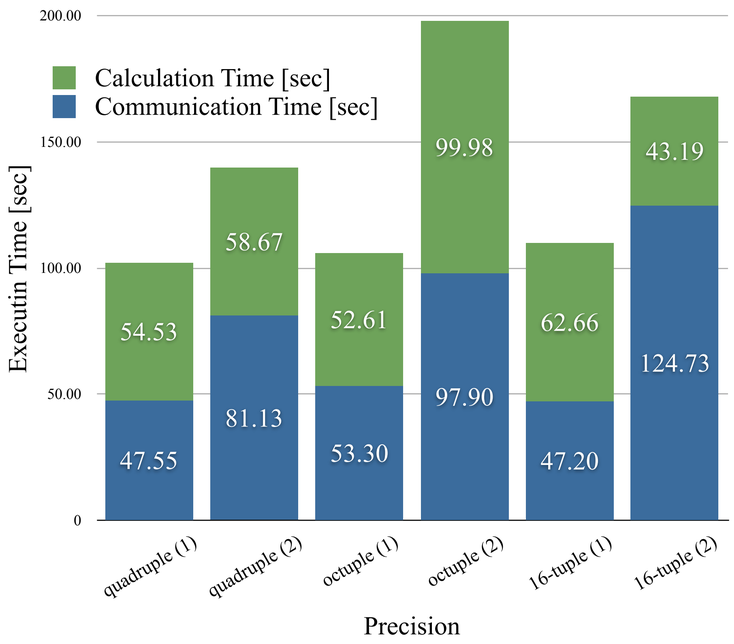

Fig. 5: The execution time of VPBi-CGSTAB for various multiple precision calculation in outer-loop. Here, (1) denotes a double precision vector communication, and (2) denotes a multiple precision vector communication.

Mixed precisioned VP Krylov subspace method can be reduced the communication time because the necessary vector for matrix-vector multiplication is derived from inner-loop. In this study, inner-loop is calculated with double precision, and double precision vector is only communicated. (Fig. 5)

The communication time including the sharing vector is reduced over 60 %

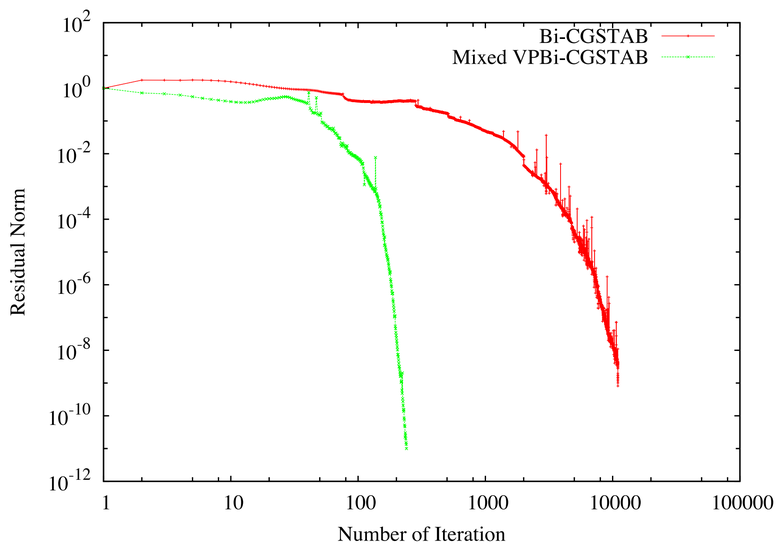

Fig. 6: The residual history of quadruple precisioned VPBi-CGSTAB and standard Bi-CGSTAB.

The residual history of the quadruple precisioned VPBi-CGSTAB is evaluated. The inner-loop is fixed as double precision operation (see Fig. 6).

The number of iteration could be reduced by using VPBi-CGSTAB, and higher accuracy could be obtained.

However, the tendency of the residual history of over octuple precision is almost same as quadruple one.

Evaluation of performance for multiple precisioned VPBi-CGSTAB (see Fig. 8 to 10)

We achieve that the VPBi-CGSAB on GPU cluster is up to 16.38 times faster than that of OpenMP.

In case of the parallelization (unit node per Rank), speedup, speedup decrease as number of Rank increase because of the overhead. (see Fig. 7)

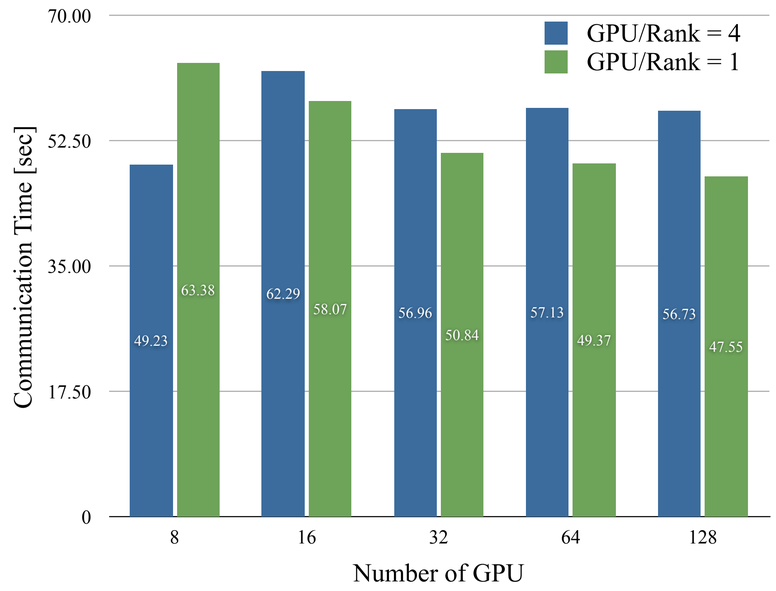

Fig. 7: The communication time of quadruple precisioned VPBi-CGSTAB for various Rank allocation.

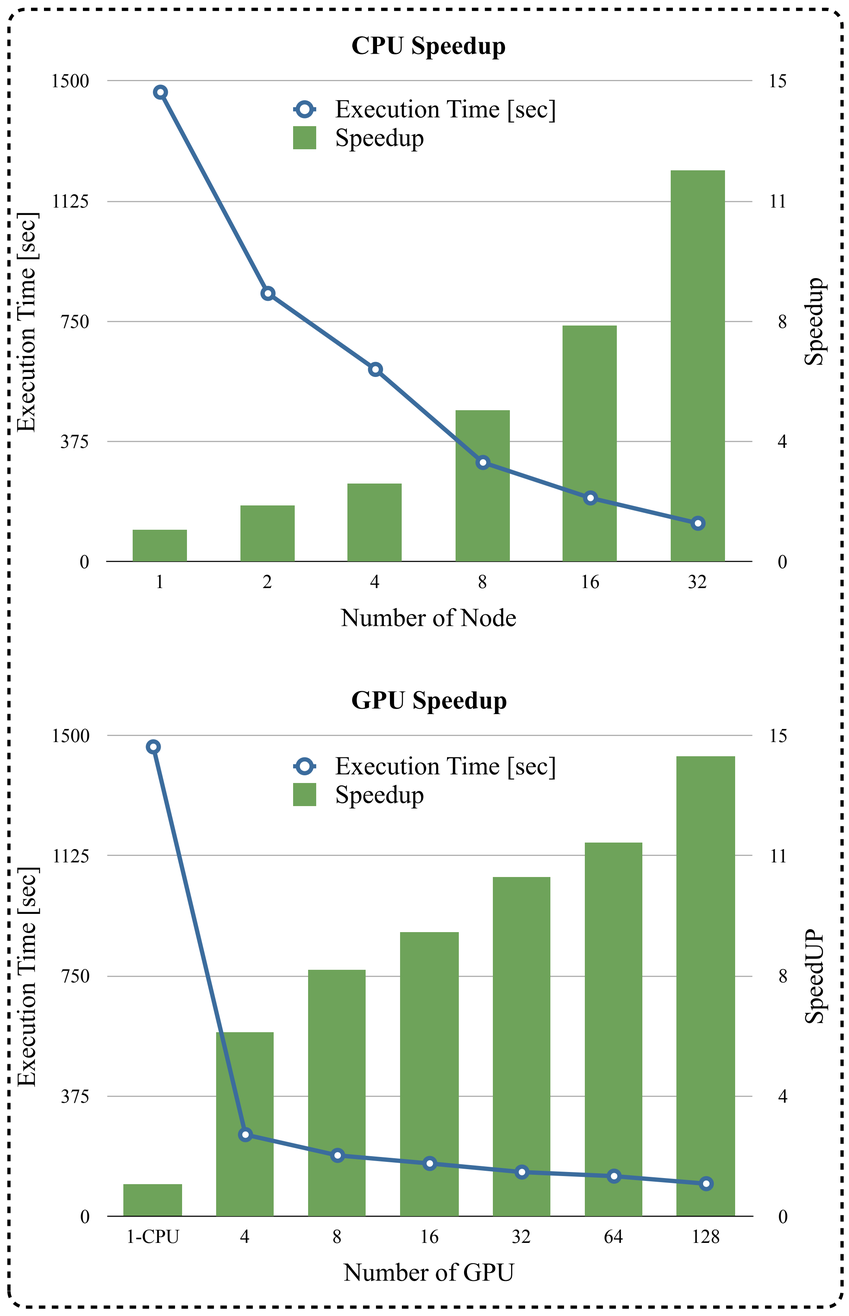

Fig. 8: The speedup of quadruple precisioned VPBi-CGSTAB.

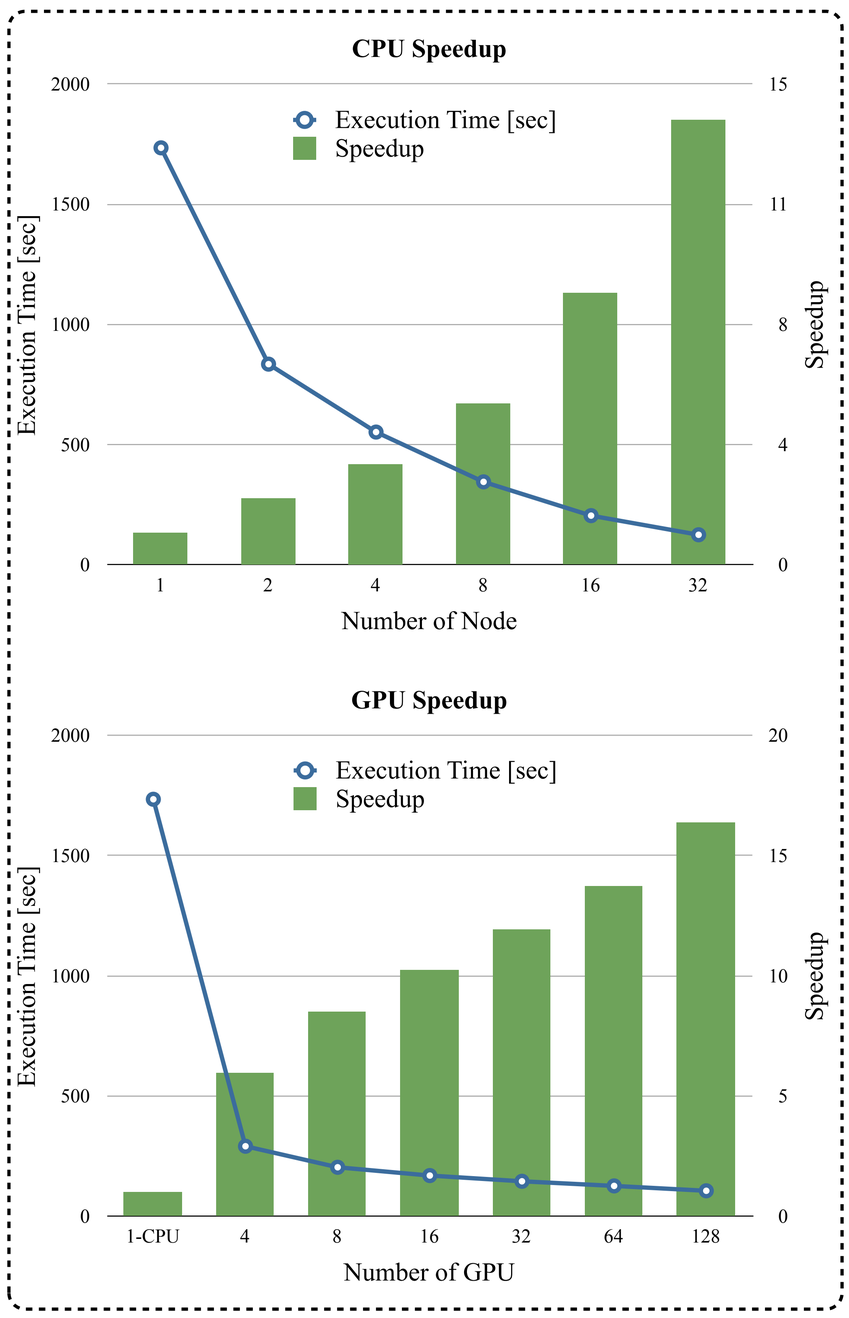

Fig. 9: The speedup of octuple precisioned VPBi-CGSTAB.

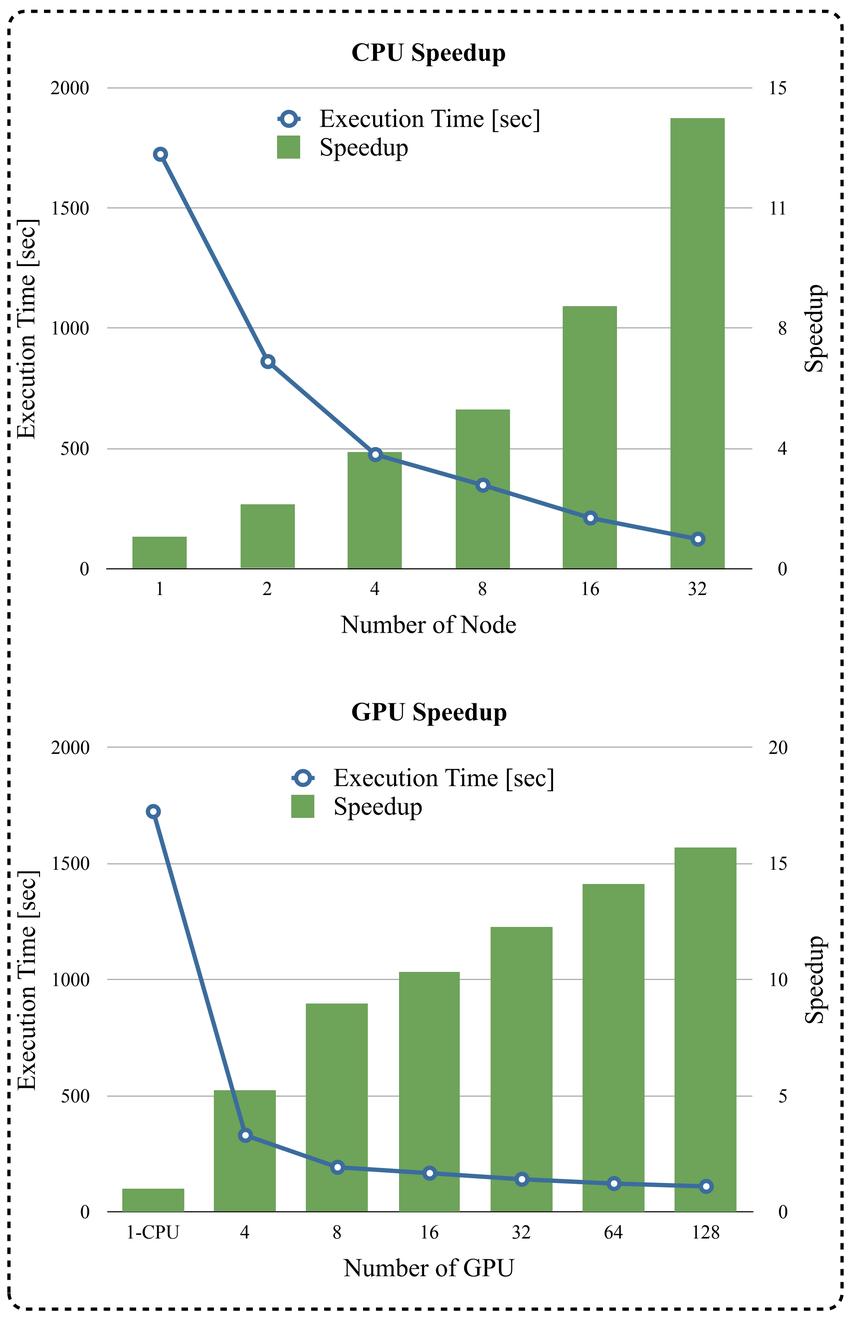

Fig. 10: The speedup of 16-tuple precisioned VPBi-CGSTAB.

8. Conclusions

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 solver for the large-sparse linear system.

The accuracy of VPBi-CGSTAB depend on the precision of inner-loop. The further evaluation and experiment is necessary. This is our future work.

Acknowledgement

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

References

1) The GNU Project: GMP, The GNU Multiple Precision Arithmetic Library, http://gmplib.org/
2) T. Nakayama and D. Takahashi: “Implementation of Multiple-Precision Floating-Point Arithmetic Library for GPU Computing”, Proc 23rd IASTED Internal Conference on PDCS 2011, pp. 343 - 349 (2011) .
3) International Compumag Society, Testing Electromagnetic Analysis Methods. http://www.compumag.org/jsite/team.html
4) K. Abe and S. L. Zhang: “A variable preconditioning using the SOR method for GCR-like methods”, Int.J.Number.Anal Model.2, No.2, pp. 118 - 128 (2002).
5) Center for Computational Sciences, University of Tsukuba: HA-PACS Project, http://www.ccs.tsukuba.ac.jp/CCS/research/project/ha-pacs