2015 International Symposium on Nonlinear Theory and its Applications
NOLTA2015, Kowloon, Hong Kong, China, December 1-4, 2015

Implementation of Communication Avoiding Technique on Krylov Subspace Method

Gong Chen[1], Yoshihisa Fujita[2], Taku Itoh[3], Hiroaki Kurokawa[1] and Soichiro Ikuno[1]

[1] School of Computer Science, Tokyo University of Technology 1404-1 Katakura, Hachioji 192-0982, Japan
[2] Department of Energy Engineering and Science Nagoya University, Toki, Gifu 509-5292, Japan
[3] College of Industrial Technology, Nihon University, Narashino, Chiba 275-8575, Japan
Email: g2115014@edu.teu.ac.jp, ikuno@cs.teu.ac.jp

Abstract—In the present study, the communication avoiding technique is numerically evaluated. Generally, the algorithm of Krylov subspace method is very simple, so that the method have a good chemistry with parallelization techniques. However, communication time becomes top issue to derive the high performance calculation. To avoid the communication issue, we have implemented the $k-$skip conjugate gradient (CG) method, and the numerical character have been evaluated. In case of $k-$skip CG method, the linear system can be solved by $k = 1$ and $k = 2$. However, the residual norm behaves unstable as increase the value of k. This is because that the orthogonality of residual vector vanishing on $k-$skip account.

1. Introduction

Recently, the performance of the Central Processing Unit (CPU), the Graphics Processing Unit (GPU) and Many Integrated Core (MIC) have increased in each of the past ten years. In addition, these devices are constituted by multitudes of processing units. Therefore, a parallelization scheme must be implemented on the simulation code in order to educe the performance of the devices.

As is well known that the speedup of the parallelization technique is governed by Amdahl's law. The simulation code can be divided into two parts. One is the parallelizable part, the other part consist of preprocessing, sequential calculation part. Thus, the value of speedup S is defined by following equation.

$$S = \frac{1}{1 - r + \frac{r}{n}}. \tag{1}$$

Here, r denotes a ratio of parallelizable part in the code, and n denotes a number of process. Equation (1) indicates that although the calculation cost decreases as increases the number of processing unit, the lower limit inevitably exist i.e. $1 - r$. According to the idea of Amdahl's law, the calculation time decreases as the number of processing unit increases. However, the communications between the processing units are excluded. Generally speaking, the communication time increases as number of processing unit increases.

Let $\boldsymbol{x}_0$ be an initial guess.
Set $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
Set $\boldsymbol{p}_0 = \boldsymbol{r}_0$
for $k = 0, 1, \cdots,$ until $\|\boldsymbol{r}_k\|_2/\|\boldsymbol{b}\|_2 \leq \varepsilon$ **do**
$\alpha_k = \dfrac{(\boldsymbol{r}_k, \boldsymbol{r}_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)}$
$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$
$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k$
$\beta_k = \dfrac{(\boldsymbol{r}_{k+1}, \boldsymbol{r}_{k+1})}{(\boldsymbol{r}_k, \boldsymbol{r}_k)}$
$\boldsymbol{p}_{k+1} = z_{k+1} + \beta_k \boldsymbol{p}_k$
end for

Figure 1: The algorithm of the conjugate gradient (CG) method. Here, A denotes a coefficient matrix of a linear system and b denotes a known vector.

The conjugate gradient (CG) method is one of a solver for a large-sparse linear system. In addition, the algorithm of CG method is very simple, and the most of all the procedure of the method is constituted by addition of vectors, inner products and multiplication of matrices and vectors as shown in Fig. 1. These operations are very easy to get a high performance by parallelization. However, the communication between processing units (PUs) must be necessary for parallelized inner products calculation, and the amount of the communication time increases as the size of the system increase. That is to say, the communication time is bottle neck for the effective parallelization [1, 2].

A variable preconditioning method has been developed as a new preconditioning strategy for Krylov subspace methods [3]. The Variable Preconditioned (VP) Krylov subspace method has two nested iteration for main Krylov subspace method and variable preconditioning for Krylov subspace method are called as outer-loop and inner-loop. In the preconditioning procedure, the residual equation is solved roughly by using some iterative method with only a few iteration. Therefore, the algorithm of VP Krylov subspace method is very simple, so that the method have a good chemistry with parallelization techniques. However, the cost of communication i.e. moving data between levels of the memory hierarchy and each processor, is consider-

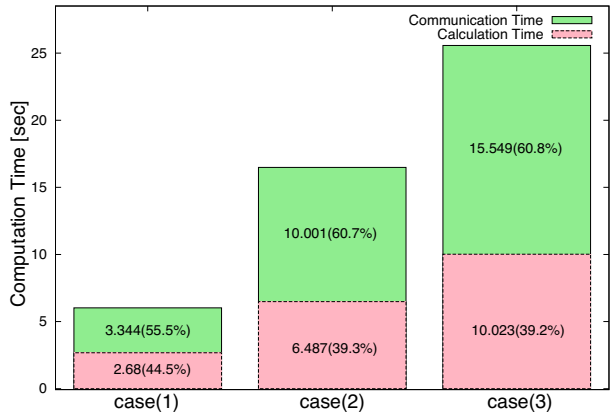

Figure 2: The computation time of CG for various types of linear system.

ably higher than the cost of the computations. Thus, in order to enhance the performance of VP Krylov subspace method, new strategies for the communication bound kernels should be explored to minimize communication and data movement [5, 6, 7, 8].

The purpose of the present study is to implement Krylov subspace method with communication avoiding techniques, and to evaluate the numerical characters of the method.

2. Communication Time of Krylov Subspace Method

Let us first investigate the communication time of standard CG on GPU. Three types of coefficient matrices are adopted for the evaluation, and the dimension size of the matrices are case(1): 428650, case(2): 6,010,480, case(3): 10,641,602, respectively [9]. The computation time of the standard CG on GPU is shown in Fig. 2. This figure indicates that the communication time increases as the dimension size increases. Furthermore, about 55 % to 60 % of total computation time spent for the communication time of transferring data between CPU and GPU global memory. That is to say, the communication time is bottle neck for the effective parallelization.

In the original variable preconditioned Krylov subspace method has the sufficient condition for convergence. The residual of the linear system converges if the relative residual norm of inner-loop satisfies the inequality in each steps. The condition is derived from the behavior of monotonic decreases of GCR. That is to say, the solver for the outer-loop should be included the behavior of monotonicity. In the previous study, we have extended the algorithm of variable preconditioned method using various Krylov subspace method for outer-loop, and the convergence characteristics of VP Krylov subspace method have been numerically evaluated [4]. The algorithm of variable preconditioned conjugate gradient (VPCG) method is shown in Fig.

Let $\boldsymbol{x}_0$ be an initial guess.
Set $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
Roughly solve $Az_0 = \boldsymbol{r}_0$ using some iterative method
Set $\boldsymbol{p}_0 = z_0$
for $k = 0, 1, \cdots,$ until $\|\boldsymbol{r}_k\|_2/\|\boldsymbol{b}\|_2 \le \varepsilon$ **do**
$\alpha_k = \dfrac{(\boldsymbol{r}_k, z_k)}{(\boldsymbol{p}_k, A\boldsymbol{p}_k)}$
$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$
$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k A\boldsymbol{p}_k$
Roughly solve $Az_{k+1} = \boldsymbol{r}_{k+1}$ using some iterative method
$\beta_k = \dfrac{(\boldsymbol{r}_{k+1}, z_{k+1})}{(\boldsymbol{r}_k, z_k)}$
$\boldsymbol{p}_{k+1} = z_{k+1} + \beta_k \boldsymbol{p}_k$
end for

Figure 3: The algorithm of variable preconditioned conjugate gradient (VPCG) method.

3. In the preconditioning procedure (inner-loop), various Krylov subspace method was adopted because Krylov subspace method is easy to parallelize than stationary iterative method such as SOR method. As the results, the communication time of parallelized inner products increase.

3. k−skip Conjugate Gradient Method [6]

The communication avoiding technique is one of a settlement for communication bottle neck issue for the parallelization CG. Although the algorithm of Krylov subspace method with the communication avoiding technique is different from original method, the theoretical methodology is equivalent. In the technique, inner products is rewritten to the recurrence formula using the basis of Krylov subspace, and the inner products are transcribed into scalar calculation. As the results, communications between PUs can be gathered in only one time. As is obvious the communication time for the parallelization CG can be reduced.

Let us derive the k−skip conjugate gradient method [6]. The inner product $(\boldsymbol{r}_{i+1}, \boldsymbol{r}_{i+1})$ can be rewritten by using $\alpha_i = (\boldsymbol{r}_i, \boldsymbol{r}_i)/(\boldsymbol{r}_i, A\boldsymbol{p}_i)$ as follows.

$$(\boldsymbol{r}_{i+1}, \boldsymbol{r}_{i+1}) = \alpha_i^2 (A\boldsymbol{p}_i, A\boldsymbol{p}_i) - (\boldsymbol{r}_i, \boldsymbol{r}_i). \tag{2}$$

Other inner products can be also rewritten as follows.

$$\begin{aligned}
(\boldsymbol{p}_{i+1}, A^j\boldsymbol{p}_{i+1}) &= (\boldsymbol{r}_{i+1}, A^j\boldsymbol{p}_{i+1}) + \beta_i(\boldsymbol{p}_i, A^j\boldsymbol{r}_i) \\
&+ \beta_i^2(\boldsymbol{p}_i, A^j\boldsymbol{p}_i) - \alpha_i\beta_i(\boldsymbol{p}_i, A^{j+1}\boldsymbol{p}_i), \quad (3)\\
(\boldsymbol{r}_{i+1}, A^j\boldsymbol{p}_{i+1}) &= (\boldsymbol{r}_{i+1}, A^j\boldsymbol{r}_{i+1}) \\
&+ (\boldsymbol{r}_i, A^j\boldsymbol{p}_i) - \alpha_i\beta_i(\boldsymbol{p}_i, A^{j+1}\boldsymbol{p}_i), \quad (4)\\
(\boldsymbol{r}_{i+1}, A^j\boldsymbol{r}_{i+1}) &= (\boldsymbol{r}_i, A^j\boldsymbol{r}_i) \\
&- 2\alpha_i(\boldsymbol{r}_i, A^{j+1}\boldsymbol{p}_i) + \alpha_i^2(\boldsymbol{p}_i, A^{j+2}\boldsymbol{p}_i). \quad (5)
\end{aligned}$$

The values of left hand side of (2), (3), (4) and (5) can be calculated by using i−th step values α_i, β_i,

Let $\boldsymbol{x}_0$ be an initial guess.
Set $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$
Set $\boldsymbol{p}_0 = \boldsymbol{z}_0$
$n = k$
for while $|\gamma_k|/\|\boldsymbol{b}\|_2 \leq \varepsilon$ **do**
  calculate $\boldsymbol{r}_n, A\boldsymbol{r}_n, A^2\boldsymbol{r}_n, \cdots, A^k\boldsymbol{r}_n$
  calculate $\boldsymbol{p}_n, A\boldsymbol{p}_n, A^2\boldsymbol{p}_n, \cdots, A^k\boldsymbol{p}_n, A^{k+1}\boldsymbol{p}_n$
  calculate γ_n
  calculate $\delta_{n,1}, \cdots, \delta_{n,2k}$
  calculate $\eta_{n,1}, \cdots, \eta_{n,2k}, \eta_{n,2k+1}$
  calculate $\zeta_{n,1}, \cdots, \zeta_{n,2k}, \zeta_{n,2k+1}, \zeta_{n,2k+2}$
  for $i = n, 1, \cdots, n+k$ **do**
    $\alpha_i = \gamma_i/\zeta_{i,1}$
    $\beta_i = \alpha_i\, \zeta_{i,2}/\zeta_{i,1} - 1$
    $\gamma_{i+1} = \beta_i\, \gamma_i$
    for $j = 1, \cdots, 2k - 2(i-n)$ **do**
      $\delta_{i+1,j} = \delta_{i,j} - 2\alpha_i\, \zeta_{i,j+1} + \alpha_i^2 \zeta_{i,j+2}$
      $\eta_{i+1,j} = \delta_{i+1,j} + \beta_i\, \eta_{i,j+1} - \alpha_i\, \beta_i \zeta_{i,j+1}$
      $\zeta_{i+1,j} = \eta_{i+1,j} + \beta_i\, \eta_{i,j} + \beta_i^2\, \zeta_{i,j} - \alpha_i\, \beta_i \zeta_{i,j+1}$
    end for
    $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{p}_i$
    $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i A\boldsymbol{p}_i$
    $\boldsymbol{p}_{i+1} = \boldsymbol{r}_{i+1} + \beta_i \boldsymbol{p}_i$
  end for
  $n = n + k + 1$
end for

Figure 4: The algorithm of $k-$skip conjugate gradient method.

$(\boldsymbol{p}_i, A^j\boldsymbol{p}_i)$, $(\boldsymbol{p}_i, A^{j+1}\boldsymbol{p}_i)$, $(\boldsymbol{p}_i, A^{j+2}\boldsymbol{p}_i)$, $(\boldsymbol{r}_i, A^j\boldsymbol{p}_i)$, $(\boldsymbol{r}_i, A^{j+1}\boldsymbol{p}_i)$ and $(\boldsymbol{r}_i, A^j\boldsymbol{r}_i)$. Especially, in CG method, $(i+1)-$th step values $(\boldsymbol{r}_{i+1}, \boldsymbol{r}_{i+1})$, α_{i+1} and β_{i+1} can be calculated using $(\boldsymbol{p}_{i+1}, A\boldsymbol{p}_{i+1})$ and $(\boldsymbol{p}_{i+1}, A^2\boldsymbol{p}_{i+1})$, and the step can be forwarded one step without inner product calculation. By using following notations, the algorithm of $k-$skip CG method can be derived as shown in Fig. 4.

$$\begin{aligned}
\gamma_i &= (\boldsymbol{r}_i, \boldsymbol{r}_i),\\
\delta_{i,j} &= (\boldsymbol{r}_i, A^j\boldsymbol{r}_i),\\
\eta_{i,j} &= (\boldsymbol{r}_i, A^j\boldsymbol{p}_i),\\
\zeta_{i,j} &= (\boldsymbol{p}_i, A^j\boldsymbol{p}_i).
\end{aligned}$$

From Fig. 1 and Fig. 4, we can derive the number of operation of the methods as shown in Table 1. We can see from Fig. 4 and Table 1 that the number of operation of $k-$skip CG is higher than that of CG. However, inner product communication occurs only one time in unit iteration [6].

4. Numerical Results

In this section, $k-$skip CG method is evaluated by using two types of tridiagonal matrices. The values of subdiagonals are set to -1, and main diagonals are set to 25 (Type 1) and 2.5 (Type 2). The condition number of the matrices become 1.17 (Type 1) and 8.97 (Type 2), respectively. That is to say, we can control the illposedness of the linear system by changing the value of main diagonals. Moreover, elements of right hand side vector are set to 1, and termination condition is set to 10^{-10}.

Table 1: Number of operations of CG and $k-$skip CG in $k+1$ iterations. Here, N_{MV}, N_{in}, N_{comm}, N_{s} and N_{v} denote number of matrix vector multiplication, number of inner product, number of communication for inner product, number of scalar operation and number of addition of vector, respectively.

Method	N_{MV}	N_{in}	N_{comm}	N_{s}	N_{v}
CG	$k+1$	$2k+2$	$2k+2$	$O(k)$	$3k+3$
$k-$skip	$3k+2$	$3k+3$	1	$O(k^2)$	$3k+3$

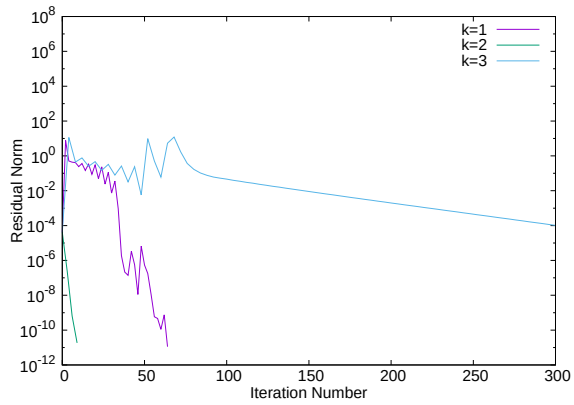

Figure 5: The residual histories of $k-$skip CG method. The values main diagonals of the coefficient matrix are 25.

It is known that (2) causes the calculation instability because the equation includes subtraction. To avoid the truncation error, following procedure is adopted for γ_i calculation [1, 6].

$$\begin{aligned}
\mathtt{tmp}_0 &= \gamma_i - \alpha_i\, \eta_{i,1},\\
\mathtt{tmp}_1 &= \eta_{i,1} - \alpha_i\, \zeta_{i,2},\\
\gamma_{i+1} &= \mathtt{tmp}_0 - \alpha_i\, \mathtt{tmp}_1.
\end{aligned}$$

The residual histories of $k-$skip CG method are shown in Fig. 5 (Type 1) and Fig. 6 (Type 2). In case of Type 1, the linear system can be solved by $k = 1$ and $k = 2$. However, the residual norm behaves unstable as increase the value of k. This is because that the orthogonality of residual vector vanishing on $k-$skip account. This tendency is significantly enhanced in case of Type 2 because of the high condition number (see Fig. 6). As the results, the iteration number increase as the value of k increases. On the other hand, the number of communication decreases as the value of k increases. Thus, the effective parallelization should be expected.

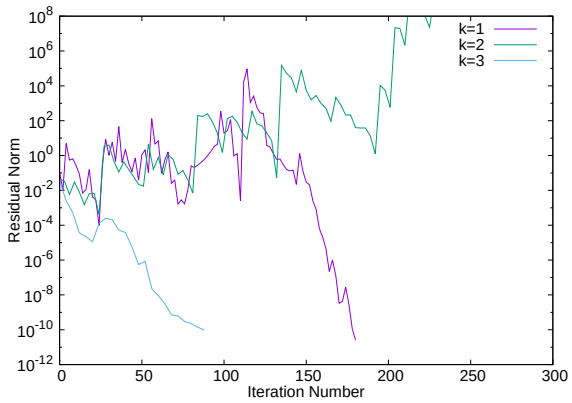

Figure 6: The residual histories of $k-$skip CG method. The values main diagonals of the coefficient matrix are 2.5.

The detail results of the parallelization will be presented at the NOLTA2015.

5. Conclusions

In the present study, we implemented CG method on GPU and evaluated the communication time of the method. Furthermore, numerical character of $k-$skip CG method have been evaluated.

Conclusions obtained in this study are summarized as follows.

- In case of standard CG method, about 55 % to 60 % of total computation time spent for the communication time of transferring data between CPU and GPU global memory. That is to say, the communication time is bottle neck for the effective parallelization.
- Although the linear system can be solved by $k-$skip CG method with $k = 1$ and $k = 2$, the residual norm behaves unstable as increase the value of k. This is because that the orthogonality of residual vector vanishing on $k-$skip account.
- The number of communication decreases as the value of k increases. Thus, the effective parallelization should be expected.

In the future work, $k-$skip CG method is implemented on variable preconditioning Krylov subspace method to reduce the communication time of the method. Moreover, stabilization technique will be implemented on the method as well.

Acknowledgment

This work was supported in part by Japan Society for the Promotion of Science under a Grant-in-Aid for Scientific Research (C) No. 26390135.

References

[1] SAAD, Youcef, "Practical use of polynomial preconditionings for the conjugate gradient method", *SIAM Journal on Scientific and Statistical Computing*, 1985, 6.4: 865-881.

[2] MEURANT, Gerard, "Multitasking the conjugate gradient method on the CRAY X-MP/48", *Parallel Computing*, 1987, 5.3: 267-280.

[3] K. Abe, S. L. Zhang, "A variable preconditioning using the SOR method for GCR-like methods," *Int. J. Numer. Anal. Model.* 2, No.2, pp. 137-151, 2005.

[4] S. Ikuno, Y. Kawaguchi, 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 Trans. Magn.*, **48**, No. 2 (2012) 467-470.

[5] Mark Hoemmen, "Communication-Avoiding Krylov Subspace Methods,", The doctoral dissertation of University of California at Berkeley, Berkeley, CA, USA, 2010.

[6] Toru Motoya and Reiji Suda, "k-skip Conjugate Gradient Methods: Communication Avoiding Iterative Symmetric Positive Definite Sparse Linear Solver For Large Scale Parallel Computings", *IPSJ SIG Tech. Rep.*, 2012, Vol.2012-HPC-133 No.30 (Japanese).

[7] DAzevedo, E. F., V. Eijkhout, and C. H. Romine, "Lapack Working Note 56 Conjugate Gradient Algorithms with Reduced Synchronization Overhead on Distributed Memory Multiprocessors", Technical Report, Mathematical Sciences Section, Oak Ridge National Laboratory, 1999.

[8] CHRONOPOULOS, A. T., GEAR, Charles William, "s-Step iterative methods for symmetric linear systems," Journal of Computational and Applied Mathematics, 1989, 25.2: 153-168.

[9] The University of Florida Sparse Matrix Collection, T. A. Davis and Y. Hu, "ACM Transactions on Mathematical Software," Vol. 38, Issue 1, 2011, pp 1:1 - 1:25.