A proposal of communication-hiding pipelined BiCGSafe algorithm and its application to GPU-based numerical simulation of blood flow

◯ HUYNH Quang Huy Viet
SUITO Hiroshi
Advanced Institute for Materials Research, Tohoku University
E-mail: {hqhviet, hiroshi.suito}@tohoku.ac.jp

Recently, the new variants of the BiCGStab algorithm with hiding communication latency of computing inner products by overlapping inner-product computations with a matrix-vector computation have been proposed by Cools and Vanroose. On parallel computers, this method can gain higher scalability property than the standard BiCGStab method. Among generalized algorithms of the BiCGStab method such as GPBiCG, BiCGSafe, BiCGStar-plus, GPBiCG variant 1, 2, 3, 4, BiCGSafe is an algorithm with good convergence behaviors. In this paper, similar to the work of Cools and Vanroose, we propose a variant of BiCGSafe named *Pipelined BiCGSafe* that hides communication latency. To verify the effectiveness of the proposed algorithm for real problems, we apply it to blood flow simulation.

1. Introduction

Simulation of blood flow by using numerical methods has become an emerging research area. The understanding of flow patterns of blood flow is useful in the prevention, diagnosis, and treatment of cardiovascular disease - a class of diseases involving narrowed or blocked blood vessels that can cause a range of heart diseases such as check pain, heart attack. In the numerical simulation of blood flow, the discretization of the Navier-Stokes equations (NSE) by finite element methods leads to a large scale system of linear equations. Solving systems of linear equations is also a central problem for many algorithms in scientific and engineering computations. One of the widely used methods for solving linear equation systems is the BiCGStab [7] iterative algorithm, which uses an initial solution and creates a sequence of improved approximate solutions. The BiCGStab algorithm is built from three basic operations: inner-product, linear combination (the addition of a scalar multiple of one vector to another vector), and matrix-vector product. In real-life applications, it is essential to speed up the solution process of large-scale linear equation systems. In the parallel implementation of the BiCGStab algorithm, one main problem that causes delays to the whole process is the inner product operation, which requires a global synchronization phrase for one global communication operation to collect the scalar partial sums in each processor to one processor, and one global communication operation for distributing the result to all processors. Time for inner product computation will dominate the time of the whole algorithm as the number of processors increases. Recently, the new variants of the BiCGStab algorithm with hiding communication latency of computing inner products by overlapping inner-product computations with a matrix-vector computation have been proposed by Cools and Vanroose [1]. On parallel computers, these methods can gain higher scalability property than the standard BiCGStab algorithm. Among generalized algorithms of BiCGStab method such as GPBiCG [5], BiCGSafe [2][3], BiCGStar-Plus [4], GPBiCG variant 1, 2, 3, 4 [6], BiCGSafe is the algorithm with good convergence behaviors. In this paper, similar to the work of Cools and Vanroose, we propose a variant of BiCGSafe named *Pipelined BiCGSafe* that hides communication latency. GPU (graphics processing units) computing has recently been recognized as a powerful platform to achieve high performance. For the simulation of blood flow, we have developed a GPU-based NSE solver [9] founded on the stabilized finite element method [8]. To verify the effectiveness of the proposed algorithm for real problems, we apply it to a GPU-based simulation of blood flow in the aorta of the human body.

2. Pipelined BiCGSafe Algorithm

Algorithm 1 ssBiCGSafe2 [3]

Let $\boldsymbol{x}_0$ is an initial guess,
Compute $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$,
Choose $\boldsymbol{r}_0^*$ such that $(\boldsymbol{r}_0^*, \boldsymbol{r}_0) \neq 0$, e.g., $\boldsymbol{r}_0^* = \boldsymbol{r}_0$,
for $i = 0, 1, ...$ **do**
  if $||\boldsymbol{r}_i||/||\boldsymbol{r}_0|| \leq \epsilon$ **stop**,
  Compute $\boldsymbol{A}\boldsymbol{r}_i$,
  Define the scalar variables for the inner products:
  $a_i \coloneqq (\boldsymbol{A}\boldsymbol{r}_i, \boldsymbol{A}\boldsymbol{r}_i)$, $b_i \coloneqq (\boldsymbol{y}_i, \boldsymbol{y}_i)$, $c_i \coloneqq (\boldsymbol{A}\boldsymbol{r}_i, \boldsymbol{y}_i)$,
  $d_i \coloneqq (\boldsymbol{A}\boldsymbol{r}_i, \boldsymbol{r}_i)$, $e_i \coloneqq (\boldsymbol{y}_i, \boldsymbol{r}_i)$, $f_i \coloneqq (\boldsymbol{r}_0^*, \boldsymbol{r}_i)$,
  $g_i \coloneqq (\boldsymbol{r}_0^*, \boldsymbol{A}\boldsymbol{r}_i)$, $h_i \coloneqq (\boldsymbol{r}_0^*, \boldsymbol{t}_{i-1})$,
  if $i = 0$ **then**
    $\beta_i = 0$,
    $\alpha_i = f_i/g_i$,
    $\zeta_i = d_i/a_i$,
    $\eta_i = 0$,
  else
    $\beta_i = (\alpha_{i-1} f_i)/(\zeta_{i-1} f_{i-1})$,
    $\alpha_i = f_i/(g_i + \beta_i h_i)$,
    $\zeta_i = (b_i d_i - c_i e_i)/(a_i b_i - c_i^2)$,
    $\eta_i = (a_i e_i - c_i d_i)/(a_i b_i - c_i^2)$,
  end if
  $\boldsymbol{p}_i = \boldsymbol{r}_i + \beta_i(\boldsymbol{p}_{i-1} - \boldsymbol{u}_{i-1})$,
  $\boldsymbol{A}\boldsymbol{p}_i = \boldsymbol{A}\boldsymbol{r}_i + \beta_i \boldsymbol{t}_{i-1}$,
  $\boldsymbol{u}_i = \zeta_i \boldsymbol{A}\boldsymbol{p}_i + \eta_i(\boldsymbol{y}_i + \beta_i \boldsymbol{u}_{i-1})$,
  Compute $\boldsymbol{A}\boldsymbol{u}_i$,
  $\boldsymbol{t}_i = \boldsymbol{A}\boldsymbol{p}_i - \boldsymbol{A}\boldsymbol{u}_i$,
  $\boldsymbol{z}_i = \zeta_i \boldsymbol{r}_i + \eta_i \boldsymbol{z}_{i-1} - \alpha_i \boldsymbol{u}_i$,
  $\boldsymbol{y}_{i+1} = \zeta_i \boldsymbol{A}\boldsymbol{r}_i + \eta_i \boldsymbol{y}_i - \alpha_i \boldsymbol{A}\boldsymbol{u}_i$,
  $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{p}_i + \boldsymbol{z}_i$,
  $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i \boldsymbol{A}\boldsymbol{p}_i - \boldsymbol{y}_{i+1}$,
end for

GPBiCG and BiCGSafe are two typical algorithms that can be considered as generalized algorithms of BiCGStab. Based on the strategy of constructing associated residuals, BiCGSafe has better convergence properties. There are two improved variants of BiCGSafe with single synchronization: ssBiCGSafe1 and ssBiCGSafe2 [3]. The main difference between the two is the use of a transposed matrix: ssBiCGSafe1 uses, but ssBiCGSafe2 does not. Based on ssBiCGSafe2, we have developed a variant named *Pipelined BiCGSafe* that hides communication latency as follows.

Algorithm 2 Pipelined BiCGSafe

1: Let $\boldsymbol{x}_0$ is an initial guess,
2: Compute $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$,
3: Choose $\boldsymbol{r}_0^*$ such that $(\boldsymbol{r}_0^*, \boldsymbol{r}_0) \neq 0$, e.g., $\boldsymbol{r}_0^* = \boldsymbol{r}_0$,
4: **for** $i = 0, 1, ...$ **do**
5: **if** $||\boldsymbol{r}_i||/||\boldsymbol{r}_0|| \leq \epsilon$ **stop**,
6: Define $a_i \coloneqq (\boldsymbol{s}_i, \boldsymbol{s}_i)$, $b_i \coloneqq (\boldsymbol{y}_i, \boldsymbol{y}_i)$, $c_i \coloneqq (\boldsymbol{s}_i, \boldsymbol{y}_i)$, $d_i \coloneqq (\boldsymbol{s}_i, \boldsymbol{r}_i)$,
7: Define $e_i \coloneqq (\boldsymbol{y}_i, \boldsymbol{r}_i)$, $f_i \coloneqq (\boldsymbol{r}_0^*, \boldsymbol{r}_i)$, $g_i \coloneqq (\boldsymbol{r}_0^*, \boldsymbol{s}_i)$, $h_i \coloneqq (\boldsymbol{r}_0^*, \boldsymbol{t}_{i-1})$,
8: **if** $i = 0$ **then**
9: $\beta_i = 0$,
10: $\alpha_i = f_i/g_i$,
11: $\zeta_i = d_i/a_i$,
12: $\eta_i = 0$,
13: **else**
14: $\beta_i = (\alpha_{i-1} f_i)/(\zeta_{i-1} f_{i-1})$,
15: $\alpha_i = f_i/(g_i + \beta_i h_i)$,
16: $\zeta_i = (b_i d_i - c_i e_i)/(a_i b_i - c_i^2)$,
17: $\eta_i = (a_i e_i - c_i d_i)/(a_i b_i - c_i^2)$,
18: **end if**
19: Compute $\boldsymbol{A}\boldsymbol{s}_i$,
20: $\boldsymbol{p}_i = \boldsymbol{r}_i + \beta_i(\boldsymbol{p}_{i-1} - \boldsymbol{u}_{i-1})$,
21: $\boldsymbol{o}_i = \boldsymbol{s}_i + \beta_i \boldsymbol{t}_{i-1}$,
22: $\boldsymbol{q}_i = \boldsymbol{A}\boldsymbol{s}_i + \beta_i \boldsymbol{v}_{i-1}$,
23: $\boldsymbol{u}_i = \zeta_i \boldsymbol{o}_i + \eta_i(\boldsymbol{y}_i + \beta_i \boldsymbol{u}_{i-1})$,
24: $\boldsymbol{w}_i = \zeta_i \boldsymbol{q}_i + \eta_i(\boldsymbol{g}_i + \beta_i \boldsymbol{w}_{i-1})$,
25: Compute $\boldsymbol{A}\boldsymbol{w}_i$,
26: $\boldsymbol{t}_i = \boldsymbol{o}_i - \boldsymbol{w}_i$,
27: $\boldsymbol{v}_i = \boldsymbol{q}_i - \boldsymbol{A}\boldsymbol{w}_i$,
28: $\boldsymbol{z}_i = \zeta_i \boldsymbol{r}_i + \eta_i \boldsymbol{z}_{i-1} - \alpha_i \boldsymbol{u}_i$,
29: $\boldsymbol{y}_{i+1} = \zeta_i \boldsymbol{s}_i + \eta_i \boldsymbol{y}_i - \alpha_i \boldsymbol{w}_i$,
30: $\boldsymbol{g}_{i+1} = \zeta_i \boldsymbol{A}\boldsymbol{s}_i + \eta_i \boldsymbol{g}_i - \alpha_i \boldsymbol{A}\boldsymbol{w}_i$,
31: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{p}_i + \boldsymbol{z}_i$,
32: $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i \boldsymbol{o}_i - \boldsymbol{y}_{i+1}$,
33: $\boldsymbol{s}_{i+1} = \boldsymbol{s}_i - \alpha_i \boldsymbol{q}_i - \boldsymbol{g}_{i+1}$,
34: **end for**

Algorithm 2 is mathematically equivalent to Algorithm 1 in the exact arithmetic. However, the strategy of hiding communication can be applied to Algorithm 2, but not to Algorithm 1. In Algorithm 2, the inner product computation (lines 6 through 18) can be performed simultaneously or in a manner that overlaps with the matrix-vector computation (line 19). A detailed derivation of Algorithm 2 will be presented in a future paper.

3. Stabilized Finite Element Method

We consider the following dimensionless form of the Navier-Stokes equations in a spatial domain $\Omega \subset \mathbf{R}^3$:

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{1}{Re}\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \quad in \quad \Omega, \tag{1}$$

$$\frac{\partial u_i}{\partial x_i} = 0 \quad in \quad \Omega. \tag{2}$$

Here, we adopt the summation convention on repeated indices that have values 1, 2, and 3. The x, y, z axes in the Cartesian coordinate system are designated as $x_i, i = 1, 2, 3$. Here, u_i represents the component of the velocity vector field $\mathbf{u}$ in the i^{th} dimension, p stands for the scalar pressure field, and Re denotes the Reynolds number.

Let us discretize the spatial domain Ω by elements Ω^e, $e = 1, 2..., n_{el}$. Let $\mathcal{S}_\mathbf{u}, \mathcal{V}_\mathbf{u}$ be the trial and test function spaces for velocity and $\mathcal{S}_p, \mathcal{V}_p$ ($\mathcal{V}_p = \mathcal{S}_p$) be trial and test function spaces for pressure. The stabilized finite element formulation of the equations (1)-(2) with the SUPG/PSPG stabilization terms can be expressed as follows [8]: Find $\mathbf{u} \in S_\mathbf{u}$ and $p \in S_p$ such that $\forall \mathbf{w} \in \mathcal{V}_\mathbf{u}$ and $\forall q \in \mathcal{V}_p$:

$$\begin{aligned}\int_\Omega w_i\left(\frac{\partial u_i}{\partial t} + \bar{u}_j\frac{\partial u_i}{\partial x_j}\right)d\Omega - \int_\Omega \frac{\partial w_i}{\partial x_i} p d\Omega \\ + \int_\Omega \frac{1}{Re}\frac{\partial w_i}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)d\Omega \\ + \sum_{e=1}^{n_{el}} \int_{\Omega_e} \tau \bar{u}_k \frac{\partial w_i}{\partial x_k}\left(\frac{\partial u_i}{\partial t} + \bar{u}_j\frac{\partial u_i}{\partial x_j} + \frac{\partial p}{\partial x_i}\right)d\Omega = 0,\end{aligned} \tag{3}$$

$$\int_\Omega q\frac{\partial u_i}{\partial x_i}d\Omega + \sum_{e=1}^{n_{el}} \int_{\Omega_e} \tau \frac{\partial q}{\partial x_i}\left(\frac{\partial u_i}{\partial t} + \bar{u}_j\frac{\partial u_i}{\partial x_j} + \frac{\partial p}{\partial x_i}\right)d\Omega = 0, \tag{4}$$

where τ is the SUPG/PSPG stabilization parameter. The formula for calculating the coefficient τ is detailed in the paper [8].

4. Numerical Results

The discretization of the Navier-Stokes equation by stabilized finite element method leads to a large and sparse non-symmetric system of linear equations. This linear equation system is solved by using by the proposed BiCGSafe algorithm in which the inner-product computations are hidden.

To implement the proposed algorithm on the GPU platform, we used the Nvidia's GPU linear algebra libraries cuSPARSE [10] and cuBLAS [11] to implement four basic vector operations of the algorithm: SpMV (sparse matrix-vector product,), DOT (inner product), AXPY (add a multiple of one vector to another),and SCAL (scaling a vector by a constant).

The computational conditions are as follows:

- Intel(R) Xeon(R) Gold 6130 CPU, 2.10GHz
- GPU NVIDIA Tesla P4, 48 GB System Memory.
- NVIDIA's CUDA Compiler (NVCC), GCC 5.4,

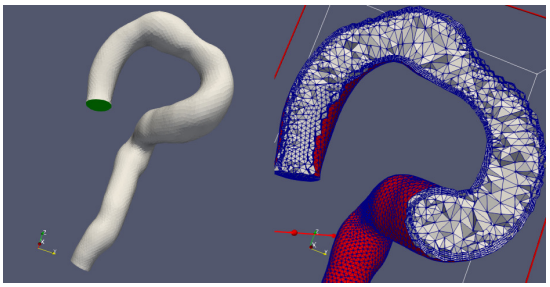

Fig. 1: Triangle surface and tetrahedral volume meshes

- Cuda 9.1, double precision,
- Compiler options: -O3 -std=c++14
- Stopping criteria: $||r_n||/||b|| < 10^{-9}$.

The open-source software Gmsh [12] is used to generate the 3D finite element mesh shown in Fig. 1. The inlet velocity boundary condition is time-dependent as shown in Fig. 2. The time step is set to start at 0s and finish at 0.5s with a time interval of 0.002s.

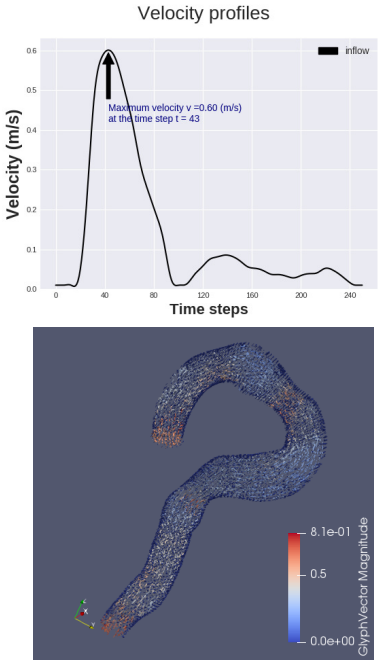

Fig. 2: Inlet velocity profile and the result flow pattern

The computation was performed successfully on GPU and CPU. Table 1 shows a speed-up ratio and execution times of GPU and CPU computations. GPU execution is significantly faster than CPU execution.

Tab. 1: Speed-up ratios and execution times

Mesh Size		GPU	CPU	Speed
#Node	#Element	Time	Time	Up
32242	173439	4201s	27729 s	6.6

Although the above numerical results show the effectiveness of the proposed algorithm, the possibility of simultaneous calculation of the inner products and the matrix-vector product has not been considered at the current implementation stage. This will be carried out in the near future.

5. Conclusions

We propose the *Pipelined BiCGSafe* algorithm that can hide the latency of inner product computation by matrix-vector computation, and show its application in GPU-based simulation of blood flow in the aorta of the human body. In future work, we will further improve the current implementation by exploring the possibility of simultaneous computation of the inner products and the matrix-vector product.

6. Acknowledgment

This work was supported by JST CREST Grant Number JPMJCR15D1, Japan.

REFERENCES

(1) S. Cools, W. Vanroose, The communication-hiding pipelined BiCGStab method for the parallel solution of large unsymmetric linear systems. Parallel Comput. 65, 1—20 (2017).

(2) S. Fujino, M. Fujiwara, M. Yoshida, BiCGSafe method based on minimization of associate residual, Transactions of the Japan Society for Computational Engineering and Science, 2005, Volume 2005.

(3) S. Fujino, K. Iwasato, An Estimation of Single-Synchronized Krylov Subspace Methods with Hybrid Parallelization, Proceedings of the World Congress on Engineering 2015 Vol I.

(4) S. Fujino and K. Murakami, A Parallel Variant of BiCGStar-Plus Method Reduced to Single Global Synchronization, AsiaSim, pp.325–332, Springer Verlag, Berlin, 2013.

(5) S.L. Zhang, GPBi-CG, Generalized product-type methods preconditionings based on BiCG for solving nonsymmetric linear systems, SIAM J. Sci. Comput., pp.537–551, 1997.

(6) K. Abe, and G.L.G. Sleijpen, Solving Linear Equations with a Stabilized GPBiCG Method, Applied Numerical Math., 67 (2013) 4–16.

(7) H.A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., 13 (1992), 631–644.

(8) T.E. Tezduyar, Stabilized finite element formulations for incompressible flow computations, Advances in Applied Mechanics 28 (1992) 1–44.

(9) V.Q.H. Huynh, H. Suito, Multi-GPU Implementation of a Parallel Solver for Incompressible Navier-Stokes Equations Discretized by Stabilized Finite Element Formulations, RIMS Kokyuroku, Vol. 2037 (2016), pp. 149-152.

(10) CuSPARSE, `https://developer.nvidia.com/cusparse`.

(11) CuBLAS, `https://developer.nvidia.com/cublas`.

(12) Gmsh, `http://gmsh.info/`