Communication-Hiding Pipelined BiCGStar-Plus Method and Its Application to GPU-based Numerical Simulation of Blood Flow

HUYNH Quang Huy Viet & SUITO Hiroshi

Advanced Institute for Materials Research, Tohoku University

{hqhviet,hiroshi.suito}@tohoku.ac.jp

1. Introduction

One of the widely used methods for solving linear equation systems is the BiCGStab 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 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, this method can gain higher scalability property than the standard BiCGStab method. Among generalized algorithms of the BiCGStab method such as GPBiCG [4], BiCGSafe, BiCGStar-plus [2, 3], BiCGStar-plus has good convergence behavior. In this poster, similar to the work of Cools and Vanroose, we propose a variant of BiCGStar-plus named Pipelined BiCGStar-plus that hides communication latency. To verify the effectiveness of the proposed algorithm for real problems, we apply it to blood flow simulation.

2. The Proposed Pipelined BiCGStar-plus Algorithm

Algorithm 1 BiCGStar-plus

1: Let $\boldsymbol{x}_0$ is an initial guess,
2: Choose $\boldsymbol{r}_0^*$ such that $(\boldsymbol{r}_0^*, \boldsymbol{r}_0) \neq 0$, e.g., $\boldsymbol{r}_0^* = \boldsymbol{r}_0$,
3: Set $\boldsymbol{p}_0 = \boldsymbol{r}_0$, $\boldsymbol{Ap}_0 = \boldsymbol{Ar}_0, \boldsymbol{y}_0 = \boldsymbol{0}$,
4: **for** $i = 0, 1, ...$ **do**
5: **if** $||\boldsymbol{r}_i||/||\boldsymbol{r}_0|| \leq \epsilon$ **stop**,
6: Compute $\boldsymbol{Ar}_i$,
7: Define $\rho_i := (\boldsymbol{r}_0^*, \boldsymbol{r}_i)$, $\sigma_i := (\boldsymbol{r}_0^*, \boldsymbol{Ar}_i)$, $\tau_i := (\boldsymbol{r}_0^*, \boldsymbol{Aw}_{i-1})$,
8: Define $\mu_i := (\boldsymbol{Ar}_i, \boldsymbol{Ar}_i)$, $\nu_i := (\boldsymbol{Ar}_i, \boldsymbol{r}_i)$, $\lambda_i := (\boldsymbol{y}_i, \boldsymbol{y}_i)$,
9: Define $\omega_i := (\boldsymbol{Ar}_i, \boldsymbol{y}_i)$, $\xi_i := (\boldsymbol{y}_i, \boldsymbol{r}_i)$,
10: **if** $i = 0$ **then**
11: $\beta_i = 0$, $\alpha_i = \rho_i/\sigma_i$,
12: $\zeta_i = \nu_i/\mu_i$, $\eta_i = 0$,
13: **else**
14: $\beta_i = (\alpha_{i-1}/\zeta_{i-1})(\rho_i/\rho_{i-1})$,
15: $\alpha_i = \rho_i/(\sigma_i + \beta_i\tau_i)$,
16: $\zeta_i = (\lambda_i\nu_i - \omega_i\xi_i)/(\mu_i\lambda_i - \omega_i^2)$,
17: $\eta_i = (\mu_i\xi_i - \omega_i\nu_i)/(\mu_i\lambda_i - \omega_i^2)$,
18: **end if**
19: $\boldsymbol{s}_i = \boldsymbol{y}_i + \beta_i\boldsymbol{c}_{i-1}$,
20: $\boldsymbol{p}_i = \boldsymbol{r}_i + \beta_i\boldsymbol{w}_{i-1}$,
21: $\boldsymbol{Ap}_i = \boldsymbol{Ar}_i + \beta_i\boldsymbol{Aw}_{i-1}$,
22: $\boldsymbol{v}_i = \zeta_i\boldsymbol{r}_i + \eta_i\boldsymbol{t}_i$,
23: $\boldsymbol{z}_i = \zeta_i\boldsymbol{Ar}_i + \eta_i\boldsymbol{y}_i$,
24: $\boldsymbol{c}_i = \zeta_i\boldsymbol{Ap}_i + \eta_i\boldsymbol{s}_i$,
25: Compute $\boldsymbol{Ac}_i$,
26: $\boldsymbol{w}_i = \boldsymbol{p}_i - \boldsymbol{c}_i$,
27: $\boldsymbol{Aw}_i = \boldsymbol{Ap}_i - \boldsymbol{Ac}_i$,
28: $\boldsymbol{t}_{i+1} = \boldsymbol{v}_i - \alpha_i\boldsymbol{c}_i$,
29: $\boldsymbol{y}_{i+1} = \boldsymbol{z}_i - \alpha_i\boldsymbol{Ac}_i$,
30: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{v}_i + \alpha_i\boldsymbol{w}_i$,
31: $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \boldsymbol{z}_i - \alpha_i\boldsymbol{Aw}_i$,
32: **end for**

The proposed Algorithm 2 is mathematically equivalent to the original Algorithm 1 [3] 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 11 through 19) can be performed simultaneously or in a manner that overlaps with the matrix-vector computation (line 20). A detailed derivation of Algorithm 2 will be presented in a future paper.

Algorithm 2 Pipelined BiCGStar-plus

1: Let $\boldsymbol{x}_0$ is an initial guess,
2: Compute $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{Ax}_0$,
3: Choose $\boldsymbol{r}_0^*$ such that $(\boldsymbol{r}_0^*, \boldsymbol{r}_0) \neq 0$, e.g., $\boldsymbol{r}_0^* = \boldsymbol{r}_0$,
4: Initialize $\boldsymbol{e}_0 = \boldsymbol{Ar}_0^*$,
5: Initialize $\boldsymbol{y}_0 = \boldsymbol{g}_0 = \boldsymbol{q}_0 = \boldsymbol{o}_0 = \boldsymbol{0}$,
6: **for** $i = 0, 1, ...$ **do**
7: **if** $||\boldsymbol{r}_i||/||\boldsymbol{r}_0|| \leq \epsilon$ **stop**,
8: Define $\rho_i := (\boldsymbol{r}_0^*, \boldsymbol{r}_i)$, $\sigma_i := (\boldsymbol{r}_0^*, \boldsymbol{e}_i)$, $\tau_i := (\boldsymbol{r}_0^*, \boldsymbol{u}_{i-1})$,
9: Define $\mu_i := (\boldsymbol{e}_i, \boldsymbol{e}_i)$, $\nu_i := (\boldsymbol{e}_i, \boldsymbol{r}_i)$, $\lambda_i := (\boldsymbol{y}_i, \boldsymbol{y}_i)$,
10: Define $\omega_i := (\boldsymbol{e}_i, \boldsymbol{y}_i)$, $\xi_i := (\boldsymbol{y}_i, \boldsymbol{r}_i)$,
11: **if** $i = 0$ **then**
12: $\beta_i = 0$, $\alpha_i = \rho_i/\sigma_i$,
13: $\zeta_i = \nu_i/\mu_i$, $\eta_i = 0$,
14: **else**
15: $\beta_i = (\alpha_{i-1}/\zeta_{i-1})(\rho_i/\rho_{i-1})$,
16: $\alpha_i = \rho_i/(\sigma_i + \beta_i\tau_i)$,
17: $\zeta_i = (\lambda_i\nu_i - \omega_i\xi_i)/(\mu_i\lambda_i - \omega_i^2)$,
18: $\eta_i = (\mu_i\xi_i - \omega_i\nu_i)/(\mu_i\lambda_i - \omega_i^2)$,
19: **end if**
20: Compute $\boldsymbol{Ae}_i$,
21: $\boldsymbol{s}_i = \boldsymbol{y}_i + \beta_i\boldsymbol{c}_{i-1}$,
22: $\boldsymbol{f}_i = \boldsymbol{g}_i + \beta_i\boldsymbol{d}_{i-1}$,
23: $\boldsymbol{p}_i = \boldsymbol{r}_i + \beta_i\boldsymbol{w}_{i-1}$,
24: $\boldsymbol{q}_i = \boldsymbol{e}_i + \beta_i\boldsymbol{u}_{i-1}$,
25: $\boldsymbol{o}_i = \boldsymbol{Ae}_i + \beta_i\boldsymbol{l}_{i-1}$,
26: $\boldsymbol{v}_i = \zeta_i\boldsymbol{r}_i + \eta_i\boldsymbol{t}_i$,
27: $\boldsymbol{z}_i = \zeta_i\boldsymbol{e}_i + \eta_i\boldsymbol{y}_i$,
28: $\boldsymbol{h}_i = \zeta_i\boldsymbol{Ae}_i + \eta_i\boldsymbol{g}_i$,
29: $\boldsymbol{c}_i = \zeta_i\boldsymbol{q}_i + \eta_i\boldsymbol{s}_i$,
30: $\boldsymbol{d}_i = \zeta_i\boldsymbol{o}_i + \eta_i\boldsymbol{f}_i$,
31: Compute $\boldsymbol{Ad}_i$,
32: $\boldsymbol{w}_i = \boldsymbol{p}_i - \boldsymbol{c}_i$,
33: $\boldsymbol{u}_i = \boldsymbol{q}_i - \boldsymbol{d}_i$,
34: $\boldsymbol{l}_i = \boldsymbol{o}_i - \boldsymbol{Ad}_i$,
35: $\boldsymbol{t}_{i+1} = \boldsymbol{v}_i - \alpha_i\boldsymbol{c}_i$,
36: $\boldsymbol{y}_{i+1} = \boldsymbol{z}_i - \alpha_i\boldsymbol{d}_i$,
37: $\boldsymbol{g}_{i+1} = \boldsymbol{h}_i - \alpha_i\boldsymbol{Ad}_i$,
38: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{v}_i + \alpha_i\boldsymbol{w}_i$,
39: $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \boldsymbol{z}_i - \alpha_i\boldsymbol{u}_i$,
40: $\boldsymbol{e}_{i+1} = \boldsymbol{e}_i - \boldsymbol{h}_i - \alpha_i\boldsymbol{l}_i$,
41: **end for**

3. Simulation of Blood Flow

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 $\Omega^e, e = 1, 2..., n_{el}$. 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 [5]: Find $\mathbf{u} \in S_\mathbf{u}$ and $p \in S_p$ such that $\forall \mathbf{w} \in \mathcal{V}_\mathbf{u}$ and $\forall q \in \mathcal{V}_p$:

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

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

where τ is the SUPG/PSPG stabilization parameter [5].

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 Pipelined BiCGStar-plus algorithm. To implement the proposed algorithm on the GPU platform, we used the Nvidia's GPU linear algebra libraries cuSPARSE and cuBLAS 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).

4. Performance Results

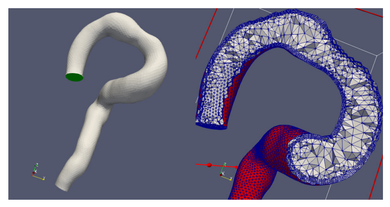

Figure 2: Triangle surface and tetrahedral volume meshes

The computational conditions are as follows:

- Intel(R) Xeon(R) CPU E5-2640 v4, 2.40GHz
- GPU NVIDIA Tesla P100,
- Cuda 10.1,
- Double precision,
- Stopping criteria: $||r_i||/||b|| < 10^{-9}$.

Inlet speed is set to 0.5 m/s. The time step is set to a time interval of 0.002 seconds for a cycle of 50 steps.

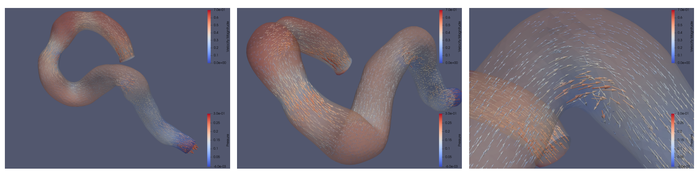

Figure 3: The result flow pattern

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.

Table 1: Execution time

Mesh Size		GPU	CPU
#Node	#Element	Time	Time
32242	173439	572 s	1158 s

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 BiCGStar-plus* 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.

Acknowledgment

This work was supported by JST CREST Grant Number JPMJCR15D1, Japan. The lead author would like to thank Professor Emeritus Seiji Fujino at Kyushu University for the support and guidance on the BiCGStar-plus method.

References

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

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

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

[4] 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).

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