Efficient Parallel Implementation of Bi-Conjugate Gradient Algorithm

Mukul Mehar
Supercomputer Education and Research Centre
Indian Institute of Science, Bangalore, India
mukulmehar@iisc.ac.in

***Abstract*—Parallelization of the BiCG algorithm has been an active area of research to achieve faster convergence and to handle large-scale systems. However, the standard parallel BiCG algorithm requires three global synchronization points for computing dot products, which can become a bottleneck in large-scale parallel computations. In this project, a pipelined version of the BiCG algorithm is proposed that reduces the number of global synchronization points required for dot products from three to one and also overlaps the global communication required for the dot product with the local computation. The experimental results on test matrices show that the proposed algorithm outperforms the standard parallel BiCG algorithm and achieves better scalability with the number of processors.**

I. Introduction

The solution of large sparse linear systems is a fundamental problem in scientific computing, engineering, and other fields. Bi-Conjugate Gradient (BiCG) algorithm is an iterative method, for solving linear systems, that is widely used due to its robustness and simplicity. In the context of parallel computing, the parallel Bi-Conjugate Gradient Stabalized (BiCGStab) algorithm is a well-known variant of BiCG that has been extensively studied.

However, the standard BiCG algorithm features three global reduction steps that require communication among all workers, which can limit the performance of the algorithm on parallel machines. To address this issue, we propose a pipelined version of the BiCG (h-BiCG) algorithm that reduces the number of global synchronization points required for dot products from three to one and overlaps the global communication required for the dot product with the local computation.

In this project report, the implementation of this pipelined algorithm is discussed and demonstrates its effectiveness. The performance of h-BiCG is also compared with the standard BiCG and other pipelined algorithms (two-point BiCGStab [1]).

The results show that the pipelined algorithm can achieve significant speedups over the standard algorithm on large-scale problems.

Overall, the main contributions of this project are:

- Reduction in number of global synchronization points, required for dot products, from three to one.
- Overlapping the global communication with the local computation.
- Comparision and analysis of the proposed algorithm with other parallel BiCG algorithms.

The remainder of this report is organized as follows. In the next section, a review of the few existing parallel BiCG algorithms is discussed. Then, the pipelined algorithm is discussed in detail along with the derivations required to reduce the global synchronization points. In the subsequent section, the experimental results and the performance of the pipelined algorithm with that of other parallel BiCG variants are discussed. Finally, the report is concluded with a short note on future work in this field.

II. Related Work

There have been several recent works on parallel Biconjugate Gradient algorithms that have explored different approaches to parallelization and optimization.

"An improved parallel hybrid bi-conjugate gradient method suitable for distributed parallel computing" by Tong-Xiang Gu et al. [2] proposes an improved parallel Bi- conjugate gradient stabilised (IBiCGSTAB) method for distributed parallel environments. The main idea of IBiCGSTAB is to eliminate data dependency of inner product computations in the original BiCG algorithm, through mathematical derivation, i.e., to cluster all inner products in one or two steps in order to reduce the negative effect of global communication. It increases the total number of inner products relative to the original BiCGSTAB, but the inner products only appear in two steps. After the local inner products in the same step have been computed in each processor they can be packed in a message and sent to each processor which computes the global inner product and broadcasts its value back to each processor. The algorithm analysis is based on a distributed memory parallel machine, which is a mesh-based processor grid with P processors. The instructions are executed in a SIMD manner. If one of the processors needs data from other processors, they are transformed by message passing and communication is carried out through a binary tree.

"The communication-hiding pipelined BiCGstab method for the parallel solution of large unsymmetric linear systems" by S. Cools and W. Vanroose [1], proposes a pipelined framework (p-BiCGStab) to implement BiCGStab, which has two main properties: avoiding communication, which is achieved by reducing the number of global reductions where possible, and hiding communication by overlapping local computational work with the global communication required for the composition of inner products. As a consequence, pipelined methods

Algorithm 1 Standard BiCGStab

```
1:  procedure BiCGStab($A$, $b$, $x_0$)
2:      $r_0 = b - Ax_0$; $p_0 = r_0$
3:      for $i = 0, 1, 2, \ldots$ do
4:          $s_i = Ap_i$
5:          begin reduction $\langle r_0,\ s_i\rangle$ end reduction
6:          $\alpha_i = \langle r_0,\ r_i\rangle / \langle r_0,\ s_i\rangle$
7:          $q_i = r_i - \alpha_i s_i$
8:          computation $y_i = Aq_i$
9:          begin reduction $\langle q_i,\ y_i\rangle$; $\langle y_i,\ y_i\rangle$ end reduction
10:         $\omega_i = \langle q_i,\ y_i\rangle / \langle y_i,\ y_i\rangle$
11:         $x_{i+1} = x_i + \alpha_i p_i + \omega_i q_i$
12:         $r_{i+1} = q_i - \omega_i y_i$
13:         begin reduction $\langle r_0,\ r_{i+1}\rangle$ end reduction
14:         $\beta_i = (\alpha_i/\omega_i)(\langle r_0,\ r_{i+1}\rangle / \langle r_0,\ r_i\rangle)$
15:         $p_{i+1} = r_{i+1} + \beta_i(p_i - \omega_i s_i)$
16:     end for
17: end procedure
```

offer better scalability in the strong scaling limit for computing solutions to large and sparse linear systems on massively parallel hardware. The experimental results point out two minor numerical drawbacks that originate from reordering the BiCGStab algorithm into a pipelined version. In the extremely small residual case, a loss of maximal attainable accuracy can be expected, which is a typical phenomenon related to pipelined (and other communication-avoiding) Krylov subspace methods. Furthermore, due to the introduction of additional axpy ($\mathbf{x} = a\mathbf{x} + \mathbf{y}$) operations by the pipelining framework, the p-BiCGStab algorithm is typically less robust with respect to numerical rounding errors compared to the standard algorithm. It is observed that the p- BiCGStab residuals are not guaranteed to remain at the same level after attaining maximal accuracy, which is a highly unwanted feature. Both of these numerical issues are simultaneously resolved by including a residual replacement strategy in the pipelined method.

III. Pipelined Bi-Conjugate Gradient Algorithm

In this section, the methodologies involved to derive a single-stage pipelined algorithm are presented. The general framework for deriving a two-staged pipelined Bi-Conjugate algorithm is presented.

The standard parallel BiCGStab algorithm, shown in “Algorithm1” [1] features three global reductions and 2 matrix-vector products along with some vector updates. When parallelized, the AXPYs computations utilized for calculating the recurrences are local tasks that do not necessitate any communication among the individual workers. In the conventional BiCGStab algorithm, three global reduction steps are incorporated to determine the dot-products necessary for calculating the scalar variables α_i (line 5-6), ω_i (line 9-10), and β_i (line 13-14). In a parallel environment, these dot-products necessitate global communication amongst all workers to gather the locally computed dot-product fractions and redistribute the final scalar result to all workers.

A. Step 1: Avoiding global communication:

Beginning with the initial BiCGStab algorithm, the first step involves reducing the number of global communication phases. This is achieved by merging the dot-product used for computing α_i (line 5) with the global reduction phase required to calculate β_i (line 13). By employing the recurrence for p_i on line 15, we can rephrase the intermediate spmv $s_i = Ap_i$ as follows:

$$\begin{aligned} s_i = Ap_i &= A(r_i + \beta_{i-1}(p_{i-1} - \omega_{i-1}s_{i-1})) \\ &= w_i + \beta_{i-1}(s_{i-1} - \omega_{i-1}z_{i-1}) \end{aligned} \tag{1}$$

where $w_i = Ar_i$ and $z_i = As_i$. Therefore,

$$\langle r_0,\ s_i\rangle = \langle r_0,\ w_i + \beta_{i-1}(s_{i-1-\omega_{i-1}-z_{i-1}}))\rangle \tag{2}$$

Substituting $\langle r_0,\ s_i\rangle$ into the definition of α

$$\alpha_i = \frac{\langle r_0,\ r_i\rangle}{\langle r_0,\ w_i\rangle + \beta_{i-1}\langle r_0,\ s_{i-1}\rangle - \beta_{i-1}\omega_{i-1}\langle r_0,\ z_{i-1}\rangle} \tag{3}$$

Since α_i is no longer dependent on the intermediate variables s_i and p_{i+1}, it can be shifted upwards, and its global reduction phase can be combined with the global reduction necessary for computing β_i (line 13). Additionally, the spmv $y_i = Aq_i$ (line 8) can be substituted with a recurrence by introducing the new variables $w_i = Ar_i$ and $z_i = As_i$, as follows:

$$y_i = Aq_i = A(r_i - \alpha_i s_i) = w_i - \alpha_i z_i \tag{4}$$

By slightly rearranging the operations, we arrive at the communication-avoiding version of the conventional parallel BiCGStab algorithm.

B. Step 2: Hiding Global Communication

Building upon step 1, our goal is to develop a communication-hiding version of BiCGStab by relocating the spmv operations to occur after the global reduction phases.

Note: In order to overlap communication for the dot products and the local computation, “MPI_IAllreduce” is used so that a process can immediately return after computing the local dot products and perform its computation independently.

$$\begin{aligned} z_i = As_i &= A(w_i + \beta_{i-1}(s_{i-1} - \omega_{i-1}z_{i-1})) \\ &= t_i + \beta_{i-1}(z_{i-1} - \omega_{i-1}v_{i-1}) \end{aligned} \tag{5}$$

where $t_i = Aw_i$ and $v_i = Az_i$.

Similarly

$$\begin{aligned} w_{i+1} = Ar_{i+1} &= A(q_i - \omega_i(w_i - \alpha_i z_i)) \\ &= y_i - \omega(t_i - \alpha_i v_i) \end{aligned} \tag{6}$$

Next, we make sure that the dot products are independent of the corresponding newly defined SPMVs.

“Algorithm2” shows the complete pipelined algorithm which features one global synchronization point less than “Algorithm1”

Algorithm 2 P-BiCGStab

```
procedure P-BiCGStab($A$, $b$, $x_0$)
    $r_0 = b - Ax_0$; $p_0 = r_0$; $w_0 = Ar_0$; $t_0 = Aw_0$; $\alpha_0 = \langle r_0,\ r_0\rangle / \langle r_0,\ w_0\rangle$; $\beta_{-1} = 0$
    for $i = 0, 1, 2, \ldots$ do
        $p_i = r_i + \beta_{i-1}(p_{i-1} - \omega_{i-1}s_{i-1})$
        $s_i = w_i + \beta_{i-1}(s_{i-1} - \omega_{i-1}z_{i-1})$
        $z_i = t_i + \beta_{i-1}(z_{i-1} - \omega_{i-1}v_{i-1})$
        $q_i = r_i - \alpha_i s_i$
        $y_i = w_i - \alpha_i z_i$
        begin reduction $\langle q_i,\ y_i\rangle$; $\langle y_i,\ y_i\rangle$
        computation $v_i = Az_i$
        end reduction
        $\omega_i = \langle q_i,\ y_i\rangle / \langle y_i,\ y_i\rangle$
        $x_{i+1} = x_i + \alpha_i p_i + \omega_i q_i$
        $r_{i+1} = q_i - \omega_i y_i$
        $w_{i+1} = y_i - \omega(t_i - \alpha_i v_i)$
        begin reduction $\langle r_0,\ r_{i+1}\rangle$; $\langle r_0,\ w_{i+1}\rangle$; $\langle r_0,\ s_i\rangle$; $\langle r_0,\ z_i\rangle$
        computation $t_{i+1} = Aw_{i+1}$
        end reduction
        $\beta_i = (\alpha_i/\omega_i)(\langle r_0,\ r_{i+1}\rangle / \langle r_0,\ r_i\rangle)$
        $\alpha_{i+1} = \langle r_0,\ r_{i+1}\rangle / (\langle r_0,\ w_{i+1}\rangle + \beta_i\langle r_0,\ s_i\rangle - \beta_i\omega_i\langle r_0,\ z_i\rangle)$
    end for
end procedure
```

C. Single Synchronization Point BiCG

Following on the same lines as for deriving a two-synchronization point pipelined algorithm ("Algorithm2"), a new single-synchronization point BiCG is introduced, Hybrid BiCG (h-BiCG), which combines the step of avoiding global communication and hiding global communication with computation one more time to reduce the global synchronization points to just one.

Note: In order to avoid an extra SPMY introduced due to the reordering of operation in "Algorithm3", the global reduction is overlapped with only one SPMY. This gives better performance than reordering.

Algorithm 3 Hybrid BiCGStab

```
procedure h-BiCGStab($A$, $b$, $x_0$)
    $r_0 = b - Ax_0$; $u_0 = Ar_0$; $f_0 = A^T r_0$; $q_0 = v_0 = z_0 = \mathbf{0}$;
    $\sigma_{-1} = \pi_0 = \phi_0 = \tau_0 = 0$; $\rho_0 = \alpha_0 = \omega_0 = 1$ ;
    $\sigma_0 = \langle r_0,\ u_0\rangle$
    for $i = 1, 2, 3, \ldots$ do
        $\rho_i = \phi_{i-1} - \omega_{i-1}\sigma_{i-2} + \omega_{i-1}\alpha_{i-1}\pi_{i-1}$
        $\delta_i = \frac{\rho_i}{\rho_{i-1}}\alpha_{i-1}$; $\beta_i = \frac{\delta_i}{\omega_{i-1}}$
        $\tau_i = \sigma_{i-1} + \beta_i\tau_{i-1} - \delta_i\pi_{i-1}$
        $\alpha_i = \rho_i/\tau_i$
        $v_i = u_{i-1} + \beta_i v_{i-1} - \delta_i q_{i-1}$
        $q_i = Av_i$
        $s_i = r_{i-1} - \alpha_i v_i$
        $t_i = u_{i-1} - \alpha_i q_i$
        $z_i = \alpha_i r_{i-1} + \beta_i z_{i-1} - \alpha_i\delta_i v_{i-1}$
        begin reduction $\langle r_0,\ s_i\rangle$; $\langle r_0,\ q_i\rangle$; $\langle f_0,\ s_i\rangle$; $\langle f_0,\ t_i\rangle$; $\langle s_i,\ t_i\rangle$; $\langle t_i,\ t_i\rangle$
        $\phi_i = \langle r_0,\ s_i\rangle$
        $\pi_i = \langle r_0,\ q_i\rangle$
        $\gamma_i = \langle f_0,\ s_i\rangle$
        $\eta_i = \langle f_0,\ t_i\rangle$
        $\theta_i = \langle s_i,\ t_i\rangle$
        $\kappa_i = \langle t_i,\ t_i\rangle$
        $\omega_i = \theta_i/\kappa_i$
        $r_i = s_i - \omega_i t_i$
        $x_i = x_{i-1} + z_i + \omega_i s_i$
        $u_i = Ar_i$
        end reduction
        $\sigma_i = \gamma_i - \omega_i\eta_i$
    end for
end procedure
```

IV. Experiments and Results

A. Experiment Setup

For analysing the results of the proposed algorithm, a random non-singular matrix of size 40000×40000 is used which is distributed across the processors such that each process gets $\frac{40000}{number\_of\_processes}$ rows of the matrix. Similarly, the vectors are distributed across the processors such that each process gets $\frac{40000}{number\_of\_processes}$ elements of the vector.

For optimal parallelization, "MPICH_ASYNC_PROGRESS=1" is set which enables non-blocking reduction.

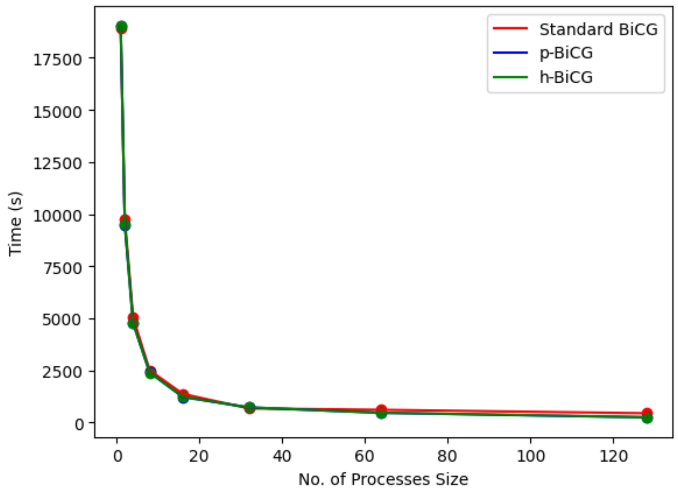
B. Results

For the analysis of speed-up, number of processes are varyied from 1 to 128. "Figure.IV-B" shows the comparision of runtimes of "Algorithm1", "Algorithm2" and "Algorithm3". h-BiCG performs better than p-BiCG and standard parallel BiCG when the number of processes and matrix is large.

Similarly, for the analysis of the accuracy, the algorithm was tested on various problem sizes for testing its convergence behaviour and compared with that of standard BiCG and p-BiCG. The results reveal that h-BiCG performs poorly in terms of accuracy as compared to p-BiCG and standard BiCG. This is because the reordering of the operation causes instability which gets reflected in its final solution. This also holds true for p-BiCG when compared with standard BiCG.

V. Conclusions

In conclusion, the successful implementation of an efficient parallel implementation of the Bi-Conjugate Gradient Algorithm using a pipelined approach that reduces the number of global synchronization points required for dot products from 3, in the standard parallel Bi-Conjugate Gradient Algorithm, to 1 is presented. Furthermore, the implementation overlaps the global communication required for the dot product with the local computation, resulting in significantly reduced communication costs and improved performance. Our experimental results demonstrate that our proposed approach can provide



significant performance improvements for large-scale linear systems, particularly on high-performance computing architectures with a large number of processors. However, as shown in “SectionIV-B” the convergence behaviour of the algorithm and its accuracy is worse than the standard parallel BiCG algorithm.

In future work, I plan to explore additional optimization techniques to further improve the performance of the implemented algorithm. This may include investigating the use of different preconditioning techniques to further accelerate the convergence of the algorithm. Overall, I believe that my approach provides a promising direction for improving the performance of the Bi-Conjugate Gradient Algorithm on high-performance computing architectures, and I look forward to continuing my research in this field.

References

[1] S. Cools and W. Vanroose, “The communication-hiding pipelined bicgstab method for the parallel solution of large unsymmetric linear systems,” *Parallel Computing*, vol. 65, no. 1-20, p. 20, 2017.

[2] T.-X. Gu, X.-Y. Zuo, X.-P. Liu, and P.-L. Li, “An Improved Parallel Hybrid Bi-Conjugate Gradient Method Suitable for Distributed Parallel Computing,” *Journal of Computational and Applied Mathematics*, vol. 226, no. 55-65, p. 11, 2009.