A New Matrix Reordering Method for GPU Acceleration of an ILU Preconditioner

Kengo Suzuki
Hokkaido University
Graduate School of Information Science and Technology
Sapporo, Japan
kengo.suzuki@eis.hokudai.ac.jp

Takeshi Fukaya
Hokkaido University
Information Initiative Center
Sapporo, Japan
fukaya@iic.hokudai.ac.jp

Takeshi Iwashita
Kyoto University
Academic Center for Computing and Media Studies
Kyoto, Japan
iwashita@i.kyoto-u.ac.jp

1 INTRODUCTION

We are interested in solving the following large sparse linear system using the Krylov subspace method on a GPU:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \text{ where } \boldsymbol{A} = \{a_{i,j}\} \in \mathbb{R}^{n\times n},\ \boldsymbol{b}, \boldsymbol{x} \in \mathbb{R}^{n}. \tag{1}$$

Due to the current trend in computer hardware architecture, there is an increasing demand for Krylov subspace solvers (and preconditioners) that can utilize accelerators such as GPUs.

The ILU(0) preconditioner has been widely used on CPUs, but its straightforward implementation cannot maximize GPU utilization because of its inherent data dependency in the substitution process. Thus, we focus on matrix reordering to enhance the fine-grained concurrency of ILU(0). Specifically, we extend our hierarchical block multi-color (HBMC) ordering [2] to GPU implementation and also propose a new variant of HBMC called multi-stage multi-color (MMC) ordering.

Recently, as another approach to improving ILU(0), there has been an interest in replacing the substitutions with an iterative method such as Jacobi [1]. Hence, we confirm the effectiveness of our technique by comparing it with this type of preconditioner.

2 HBMC ORDERING

The HBMC ordering is a variant of the standard multi-color (MC) ordering that can remedy the convergence rate deterioration while maintaining the concurrency via a block structure.

HBMC comprises three steps: blocking, coloring, and reordering, and uses a graph whose adjacency matrix is $\boldsymbol{A} + \boldsymbol{A}^{\top}$. First, the node set is divided into small subsets (blocks) of size n_b. Then, the blocks are labeled (colored) such that no dependent blocks have the same color; the dependence between two blocks, B and C, is defined when the following expression is true:

$$\exists i \in \{1, \ldots, n_b\} : \exists j \in \{1, \ldots, n_b\},\ a_{B_i, C_j} \neq 0, \tag{2}$$

where B_i and C_i denote the indexes of the i-th nodes in B and C, respectively. Finally, the nodes (and the corresponding adjacency matrix) are reordered based on the color attribute. After ordering the blocks of the same color in a sequence, in each color, i-th ($i = 1, \ldots, n_b$) nodes in the blocks are gathered to generate concurrency.

Although we developed HBMC for utilizing CPU SIMD, it can exploit GPU parallelism if a sufficient number of blocks are labeled with the same color.

3 MMC ORDERING

We propose the MMC method for further increasing the concurrency generated by HBMC. In MMC, the dependence between two blocks is redefined by

$$\exists i \in \{1, \ldots, n_b\} : a_{B_i, C_i} \neq 0. \tag{3}$$

This definition is of a stricter form of expression (2), and thus, more blocks can be labeled in the same color, increasing the fine-grained concurrency in the substitution process.

4 NUMERICAL RESULTS

We conducted numerical tests on a computing node with NVIDIA V100 GPUs. The test matrices were from the SuiteSparse matrix collection, and the right-hand side vectors were random. We developed four ILU(0) preconditioners used with FGMRES(50); the three were combined with MC, HBMC, and MMC, respectively, and the other was with the asynchronous triangular solver in Ref. [1]. We refer to these four as MC-, HBMC-, MMC-, and Async-ILU(0).

Figure 1 shows the speedup over MC-ILU(0). The proposed MMC-ILU(0) was the fastest among the three ordering-based methods in 7 out of 9 test cases. Furthermore, MMC-ILU(0) outperformed Async-ILU(0) in 7 out of 9 cases, including that MMC-ILU(0) could perform well in the tests that Async-ILU(0) required long time to converge, such as G3_circuit and ldoor. It was confirmed that in the best case, MMC-ILU(0) can improve the execution speed by up to almost 50% over MC-ILU(0).

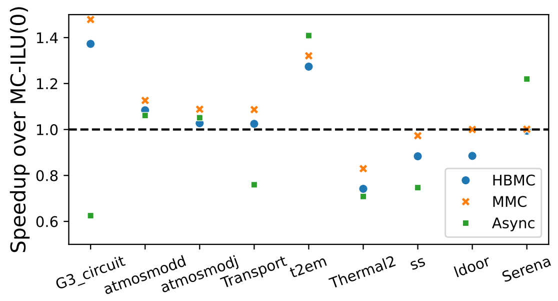

Figure 1: Speedup over MC-ILU(0)

REFERENCES

[1] Hartwig Anzt, Edmond Chow, and Jack Dongarra. 2015. Iterative sparse triangular solves for preconditioning. In *Euro-Par 2015: Parallel Processing: 21st International Conference on Parallel and Distributed Computing, Vienna, Austria, August 24-28, 2015, Proceedings 21*. Springer, 650–661.

[2] Takeshi Iwashita, Senxi Li, and Takeshi Fukaya. 2020. Hierarchical block multi-color ordering: A new parallel ordering method for vectorization and parallelization of the sparse triangular solver in the ICCG method. *CCF Transactions on High Performance Computing* 2 (2020), 84–97.