jh250043

# Innovative Multigrid Methods III

Akihiro Fujii (Kogakuin University)

Abstract

Multigrid methods play a critical role in enabling large-scale simulations on modern supercomputers. 'Innovative Multigrid Methods III' was launched in FY2023 as a three-year project. In the Final year of the project, we made significant progress in mixed precision Parallel-in-Space-Time solver, advanced research in performance analysis of number of threads in Geometric Multigrid, and also performace optimization of SA-AMG solver on Miyabi Supercomputer (GH200 Grace Hopper).

## 1 Basic Information

### 1.1 Collaborating JHPCN centers

- The University of Tokyo
- Nagoya University
- Kyoto University
- Kyushu University

### 1.2 Theme Area

- Large-scale computational science area

### 1.3 Project Members and Their Roles

Akihiro Fujii[3] : (1)
Kengo Nakajima[2,4] :(CoPI) (3),(2),(1)
Matthias Bolten[8] :(CoPI) (1),(2)
Takeshi Iwashita[1] : (1),(2)
Akihiro Ida[10] : (2)
Masatoshi Kawai[6] : (2)
Satoshi Ohshima[5] : (1),(3)
Tetsuya Hoshino[6] : (2),(3)
Toshihiro Hanawa[2] : (2),(3)
Gerhard Wellein[9] : (3)
Kenji Ono[5] : (1)
Ryo Yoda[8] : (1),(3)
Yasuhito Takahashi[7] : (1)
Yen-Chen Chen[11] : (1)
Martin Schreiber[12] : (3)
Christie Alappat[9] : (3)
Teruo Tanaka[3] : (1)
Georg Hager[9] : (3)
Ayesha Afzal[9] : (3)
Bole Ma[9] : (3)
Kazuya Yamazaki[2]: (3)
Shunsei Ito[3] : (1)
Kei Teshigawara[3] : (1)
Taketeru Kondo [3] : (1)
Sena Msaui[3] : (1)
Jin Kukita[3] : (1)

1: Kyoto U., 2: U. Tokyo, 3: Kogakuin U., 4: RIKEN R-CCS, 5: Kyushu U., 6: Nagoya U., 7: Doshisha U. 8: U. Wuppertal*, 9: FAU*, 10: JAMSTEC, 11: KIT* 12: the Université Grenoble Alpes, *:Germany

## 2 Purpose and Significance of the Research

We are planning to conduct research and development on the following three items:

(1) Research and development of fundamental algorithms in multigrid methods
(2) Parallel reordering methods
(3) Performance evaluation models for parallel multigrid procedures

(1) includes stabilization method for AMG solvers by specifying error component to be transferred to the coarse levels, and parallel time integration (PinT) algorithms. In (2), we study reordering technique on unstructured problem domain. (3) deals with performance evaluation models for multigrid solvers on supercomputers.

This year is the final year of a three-year plan. The goal is to increase the number of research papers and, at the same time, to concretize the development details for the future release of the code. From this year, we also have started researches on Miyabi (GH200 Grace Hopper) at Joint Center for Advanced High Performance Computing (JCAHPC).

## 3 Significance as JHPCN Joint Research Project

Multigrid method is scalable and used in many fields. It is known as one of the most efficient linear solvers. It can also be applied to parallel time integration problems, which exploits parallelism in time dimension. Our research project has original codes and algorithms. Therefore, research papers and codes from the project will enhance the efficiency of the multigrid solver, and will help many researchers exploit parallelism in time direction. Our research focuses on hierarchical algorithms and their performance on supercomputers. Thus, availability of supercomputers with different kinds of architectures helps us verify the codes we are developing. In addition, a JHPCN joint research project offers collaborative research opportunity with JHPCN members who have expertise knowledge in various application fields. Our project members include international experts in Germany, U.S., France and Japan on multigrid methods and PinT. We are sure that this JHPCN joint research project promotes the international collaborative activity with JHPCN members.

## 4 Outline of Research Achievements until FY2024 (Only for continuous projects)

We conducted our research project 'Innovative Multigrid II' from FY2020 to FY2022. While the project yielded substantial progress, several key research challenges remained unresolved. To address these issues more systematically, we restructured the research themes into three focused areas as described in Section 2, and launched a new three-year initiative, 'Innovative Multigrid III' starting in FY2023.

Despite a significant output of research papers covering MG, AMG, and PinT solvers, there was still a noticeable gap in performance analysis and GPU-related studies.

## 5 Details of FY2025 Research Achievements

Our research output for FY2025 consists of two journal papers, two peer-reviewed conference proceedings, and nine research presentations. This section details three representative projects conducted during this fiscal year:

- Effects of "Unused" Cores in Multigrid Procedure
- Effects of mixed-precision approach for block epsilon-circulant preconditioned solvers
- SA-AMG solver on GH200

**Effects of "Unused" Cores in Multigrid Procedure**

In the multigrid method, the levels are defined such that the finest grid is at level-1, and the level number increases as the grid becomes coarser. When dealing with large-scale problem sizes per CPU/node, the benefits of parallelization using OpenMP and MPI are significant. However, for smaller-scale ones, parallelization can sometimes lead to a decrease in computing performance. In such cases, strategies to maintain computational efficiency include reducing the number of threads, or even disabling parallelization for OpenMP. In the present work [1], we evaluate the effects of thread number for each level of multigrid procedure. If we reduce the number of cores to be used for computing, we have several unused (or idle) cores. One of the motivations for this type of research is to utilize such unused cores for other workloads. While it may be difficult to utilize several cores during the multi-grid procedures for other purposes concurrently, we try to evaluate the performance of the multigrid procedures for different number of cores/threads. In [2], number of used/unused cores was changed for do-loop's at each level of multigrid procedure. Even if there are unused cores, the program is implemented so that load balancing between used cores is maintained at each level of multigrid. Generally, effects of unused cores were not clear in [2], where all the cores are active, but no data is assigned to unused cores. In the present work, number of used/unused cores is fixed at all levels of the multigrid procedure, and effects of unused cores were evaluated. Effects of unused cores were evaluated using 16 CPU's on Wisteria/BDEC-01/Odyssey where HB $12 \times 4$ (12-threds for each MPI process, 4 MPI processes on each CPU). We show the results (elapsed computation time for MGCG solver) in Fig.1 when the number of threads during execution is reduced in each case. The computation time is normalized by the computation time when all cores are used. When the number of threads is reduced, the computation time increases when the problem size is large (l), but decreases when the it is small (s). This tendency has already been observed in our previous work [3]. When the problem size is small, reducing the number of threads may actually improve performance of multi-threaded workload. "BEST" in the figure is the sum of the shortest computation times for IC(0) smoothers at each level of multigrid, coarse

grid solver, and others (conjugate gradient method, restriction etc.). This shows that even when the problem scale is large, the computation time can be reduced by reducing the number of threads. Speed-up rates for the BEST cases are 5-15%. Fig.2 provides detailed breakdown for each procedure in MGCG solver for s12-man-b on in Fig.1. The "BEST" case combines the configuration with the shortest computation time for each of Smoothers (lev-1 lev-5), Coarse Grid Solver, and Others, where the number of used cores is between 4 and 12. The number of used cores with the highest performance for each procedure is as follows: lev-1:12, lev-2:6, lev-3:4, lev-4:4, lev-5:4, Coarse Grid:4, Others:7. Generally, fewer threads result in better performance for small problem sizes. Further performance improvements may be obtained in some procedures, such as lev-3, lev-4, and lev-5, by reducing the number of threads below 4. In the present work, "BEST" cases are ideal cases. It is possible to change the number of used/unused cures for each procedure within the program. But, the computation time of each procedure is short in the MGCG solver of the present work, while the overhead of switching the number of threads could be very large.

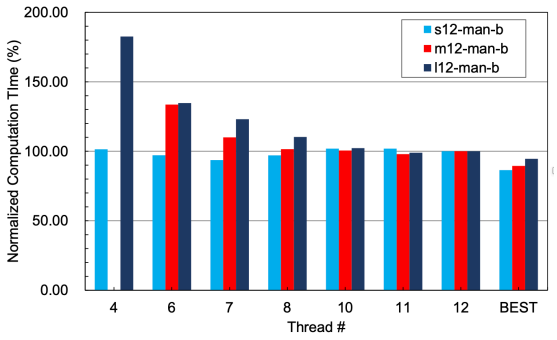


Fig. 1 Effects of Unused Cores, Elapsed Computation Time of MGCG, normalized by the computation time when all cores are used

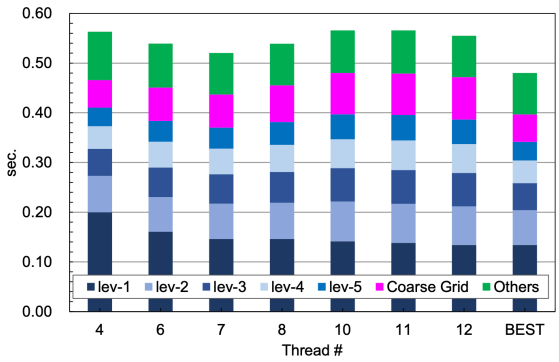


Fig. 2 Effects of Unused Cores on ODY, Elapsed Computation Time for MGCG Solver (s12-man-b), Breakdown for Each Procedure in MGCG Solver.

**Effects of mixed-precision approach for block epsilon-circulant preconditioned solvers**

For block-Toeplitz systems arising from all-at-once discretizations of time-dependent partial differential equations, block epsilon-circulant (BEC) preconditioning provides an effective parallel-in-time approach. In our works [4, 5, 6], we have proposed a mixed-precision BEC-preconditioned GMRES implementation for CPU–GPU environments. Although mixed-precision arithmetic can reduce the cost of the preconditioning stage, we observed a loss of robustness for large-scale problems, where the complex-valued diagonal blocks in the BEC preconditioner become increasingly ill-conditioned.

To address this issue, our work [7] intro-

duce two improvements: mass-matrix scaling based on a lumped mass matrix for the single-precision BEC preconditioner, and a $2 \times 2$ block CRS implementation based on an equivalent real-valued formulation of the complex-valued systems. The former suppresses the growth of the condition number with low computational overhead, whereas the latter avoids explicit complex arithmetic and enables block-wise sparse matrix operations with GPU-oriented memory access.

Figure 3 shows the effect of mass-matrix scaling. Without scaling, the condition number reaches $10^5$–$10^6$, whereas after scaling it is generally reduced to $10^2$ or less. This improvement is expected to mitigate the convergence degradation observed in the FP32 implementation. Figure 4 presents the strong-scaling results. The FP32-BEC-preconditioned GMRES solver exhibits good strong scaling up to 64 GPUs and achieves shorter runtime than both FP64-BEC-preconditioned GMRES and MGRIT, an established parallel-in-time method. In addition, the reduced memory footprint of the FP32 implementation enables computations for configurations where the FP64 implementation is difficult to execute.

The runtime breakdown in Figure 5 further clarifies the performance difference between FP64-BEC and FP32-BEC. Switching the BEC preconditioning stage to FP32 reduces the runtime of the three dominant preconditioning components. As a result, the overall solver achieves a speedup of approximately 1.4×. These results demonstrate the effectiveness of a mixed-precision strategy in which the computationally intensive preconditioning stage is accelerated in FP32 while the outer GMRES iteration is kept in FP64 for numerical robustness.

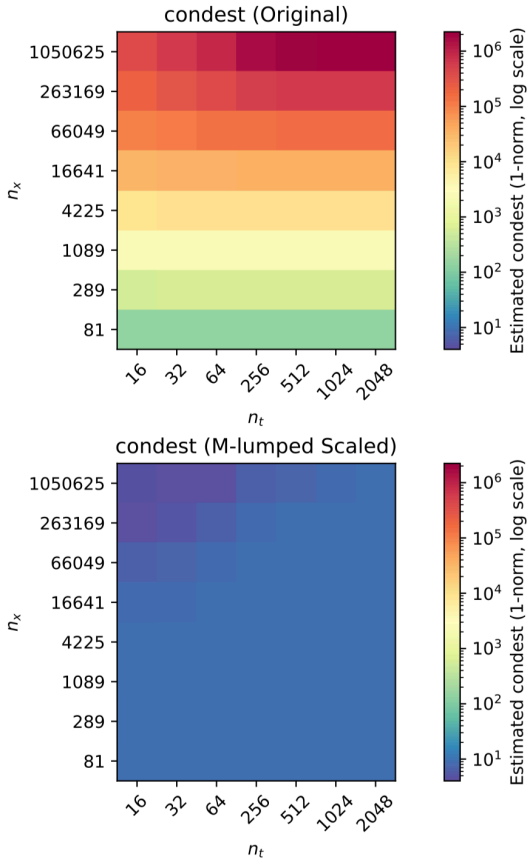


Fig. 3 Estimated condition numbers

**SA-AMG solver on GH200**[8]

The NVIDIA GH200 Grace Hopper Superchip represents a novel architecture that integrates CPU and GPU cores onto a single module. This tight integration significantly reduces communication overhead between the CPU and GPU. In this study, we conducted preliminary experiments using our multi-process implementation of the SA-AMG (Smoothed Aggregation Algebraic

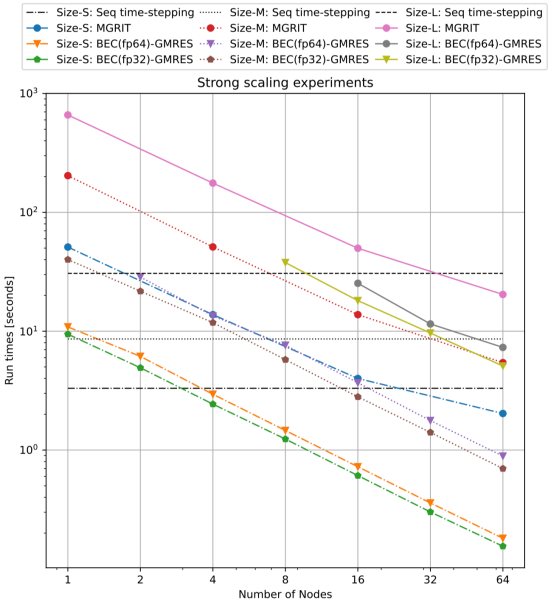


Fig. 4 Strong scaling experiments

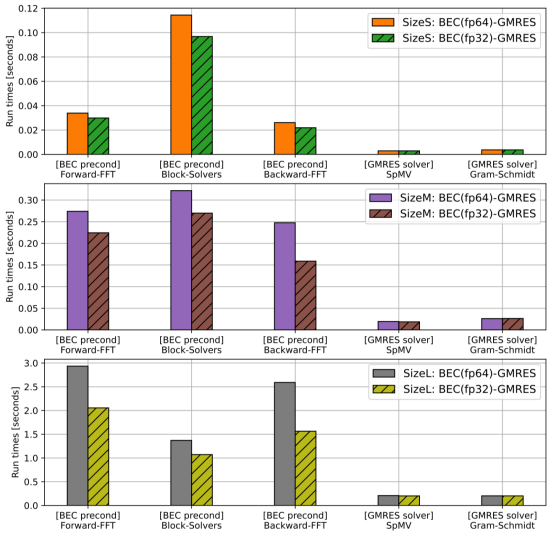


Fig. 5 Breakdown at 64 nodes

Multigrid) solver on the "Miyabi" supercomputer at JCAHPC, where each compute node is equipped with a GH200. Given the high-performance capabilities of the Grace CPU cores, the setup phase of the AMG solver is executed on the CPU. Subsequently, the iterative solver phase, which demands high throughput, is offloaded to the GPU cores. Figure 6 illustrates the execution time breakdown for a linear system with 6 million degrees of freedom (DOF) derived from fluid dynamics simulations. The left and right graphs display the results without and with NVIDIA MPS (Multi-Process Service), respectively. The vertical axis represents the execution time, while the horizontal axis indicates the number of processes. The results demonstrate that the setup phase (blue bar) scales well with the number of processes, with the thread count fixed at one. In contrast, the iterative phase initially suffered from performance degradation when multiple processes accessed the GPU simultaneously (as seen in the left graph). However, as shown in the right graph, the introduction of MPS successfully mitigates this contention, allowing our solver to maintain high performance. Since the setup phase is responsible for constructing the coarse-grid hierarchy, it is a critical factor for ensuring convergence. By leveraging the CPU for an efficient setup, our solver can construct high-quality hierarchies, which leads us to expect robust performance even when tackling highly complex problems.

## References

[1] Kengo Nakajima, `https://link.springer.com/article/10.1007/s13160-025-00732-3`, 2025.

[2] Kengo Nakajima, IPSJ SIG Technical Report, 2024-HPC-197/2024-ARC-251, No.23, 2024.

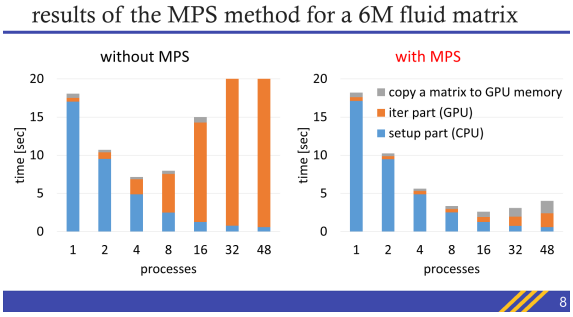


Fig. 6 Effect of MPS

[3] Kengo Nakajima, https://doi.org/10.1007/978-3-642-38718-0_39, 2013.

[4] Ryo Yoda, Matthias Bolten, "Mixed-precision implementation of block $\epsilon$-circulant preconditioner on modern CPU-GPU environments", Japan Society for Industrial and Applied Mathematics (JSIAM) Letters, JSIAML-25R017

[5] Ryo Yoda, Matthias Bolten, "Block Epsilon-Circulant Preconditioning with GPU-Accelerated Spatial Solvers for Linear Time-Dependent PDEs", 2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Milan, Italy, June 3, (2025)

[6] Ryo Yoda, Matthias Bolten, "Toward efficient solvers using block-epsilon circulant preconditioning on modern integrated CPU-GPU systems", PinT 2025: the 14th Workshop on Parallel-in-time Integration, Parallel-in-time algorithms for exascale applications, ICMS, Bayes Centre, Edinburgh, UK, July 7–12, 2025.

[7] Ryo Yoda, Matthias Bolten, "Enhancing Stability and Optimizing Implementation of Mixed-Precision Block epsilon-Circulant Preconditioned Solvers for Parallelization-in-Time", In Proceedings of the Supercomputing Asia and International Conference on High Performance Computing in Asia Pacific Region (SCA/HPCAsia '26). Association for Computing Machinery, New York, NY, USA, 177–185. (2026)

[8] Taketeru Kondo, AKihiro Fujii, Teruo Tanaka, "Mixed precision AMG solver on GH200"(in Japanese), National Convention of IPSJ, March 8 (2026)

## 6 Self-review of Current Progress and Future Prospects

We have made significant progress on PinST solvers and performance analysis of thread count adjustment of geometric multigrid solver. As for the SA-AMG solver, we have made preliminary performance test on Miyabi supercomputer (GH200). In near future, we will publish codes with our project research results.