Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures
Final Report for JHPCN Joint Research of FY 2019

jh190041-NAHI

Innovative Multigrid Methods

Kengo Nakajima（Information Technology Center, The University of Tokyo, Japan）

Abstract
In the present work, we are developing robust and efficient GMG and AMG methods, where we are focusing on development of algorithms for (1) efficient and robust smoother, (2) parallel global reordering/aggregation methods for robustness, (3) utilization of near-kernel vectors for robustness, and (4) hierarchical methods for scalability. Moreover, we develop new algorithms for Parallel-in-Space/Time (PinST).

1. Basic Information

(1) Collaborating JHPCN Centers
- The University of Tokyo (Oakforest-PACS)
- Hokkaido University (New System A)
- Kyushu University (ITO Subsystem-A)

(2) Research Areas
- ■ Very large-scale numerical computation
- □ Very large-scale data processing
- □ Very large capacity network technology
- □ Very large-scale information systems

(3) Roles of Project Members (*: Student)
- Kengo Nakajima (The University of Tokyo) (Co-PI) Administration, Applications, GMG, AMG, PinST, Hierarchical Methods
- Matthias Bolten (University of Wuppertal) (Co-PI) GMG, AMG, Smoother, Near-Kernel Vectors
- Takeshi Iwashita (Hokkaido University) PinST
- Yasuhiro Takahashi (Doshisha University) PinST
- Akihiro Fujii (Kogakuin University) AMG, PinST, Near-Kernel Vectors
- Osni Marques (Lawrence Berkeley National Laboratory) GMG, AMG, Smoother, Near-Kernel Vectors
- Ryo Yoda* (Kogakuin University) PinST
- Akihiro Ida (The University of Tokyo) GMG, AMG
- Masatoshi Kawai (RIKEN R-CCS⇒The University of Tokyo) GMG, AMG, Smoother, Global Reordering
- Naoya Nomura* (The University of Tokyo) AMG, Near-Kernel Vectors
- Yen-Chen Chen* (The University of Tokyo) PinST
- Satoshi Ohshima (Kyushu University⇒Nagoya University) Code Parallelization, Profiling & Optimization
- Tetsuya Hoshino (The University of Tokyo) Code Parallelization, Profiling & Optimization, SELL-C-σ
- Toshihiro Hanawa (The University of Tokyo) Code Parallelization, Profiling & Optimization, Mesh Generation
- Gerhard Wellein (Friedrich-Alexander-University (FAU) of Erlangen-Nürnberg) SELL-C-σ
- Lisa Claus* (University of Wuppertal) GMG, AMG, Smoother, Near-Kernel Vectors

2. Purpose and Significance of the Research

A multigrid is a scalable multilevel method for solving linear equations and preconditioning Krylov iterative linear solvers, and is especially suitable for large-scale problems because of its scalable feature. The parallel multigrid method is expected to be one of the most powerful tools on exa-scale systems. There are two approaches in the multigrid method, where one is a geometrical multigrid (GMG) with explicit hierarchical meshes, and the other is an algebraic one (AMG). Although multigrid methods have been applied to rather well-conditioned problems for a long time, many sophisticated methods for robustness of multigrid have

been developed for ill-conditioned problems derived from real-world scientific and engineering applications. In the present work, we are developing robust and efficient GMG and AMG methods, where we are focusing on development of algorithms for **(1) efficient and robust smoothers, (2) parallel global reordering/aggregation methods for robustness, (3) utilization of near-kernel vectors for robustness, and (4) hierarchical methods for scalability**.

Although parallel computations in science and engineering have been focusing on domain-decomposition approach in space direction, a new approach with parallel computation in time direction (PinST, parallel-in-space/time) has been introduced recently. MGRIT (Multigrid-Reduction-in-Time) [Falgout, R.D. et al., SIAM SISC, 2014] is one of the most well-known PinST methods, and it introduces the idea of multigrid in time domain. MGRIT was very successful in various types of applications including nonlinear ones, but it suffers from instability at coarse-level grid in time. Recently, we proposed a new method (TSC, Time Segment Correction), where no coarse time step at coarse levels in time domain is needed [Kaneko, S., Fujito, Y., Fujii, A., Tanaka, T., Iwashita, T., IPSJ SIG Technical Report, 2017-HPC-161-7, 2017 (in Japanese)]. Because preliminary results show better performance than MGRIT, we will develop efficient and robust TSC method for large-scale problems (PinST-TSC). PinST has been mainly applied to implicit problems due to constraint of time step. But, it is really needed in explicit methods, where many time steps are needed. In the present work, we will develop a new method for PinST with explicit time marching (PinST-Exp). Developed methods (GMG, AMG, PinST-TSC, PinST-Exp) are parallelized by OpenMP/MPI hybrid parallel programming model, and optimized for multicore/manycore clusters in JHPCN by SELL-C-σ matrix storage format [Kreutzer, M., Hager, G., Wellein, G. et al., SIAM SISC, 2014], where Professor G. Wellein (Friedrich-Alexander-University (FAU) of Erlangen-Nürnberg) is one of the original developers of SELL-C-σ. Performance and robustness will be evaluated by 3D FEM/FDM applications on these systems. Finally, developed programs and libraries will be deployed on the supercomputer systems in each center (Hokkaido, Tokyo and Kyushu), and released to the public.

Multigrid method is a promising approach for large-scale computing in exa-scale era. We develop robust and efficient parallel multigrid methods for both of GMG and AMG, focusing on robust and efficient smoothers, parallel reordering, utilization of near-kernel vectors, and hierarchical methods which are proposed and developed by ourselves. Both of PinST-TSC and PinST-Exp are also our original method. Developed methods will be implemented as a numerical library and it will be applied to various types of applications. This is one of the first practical library of multigrid including PinST, especially PinST-Exp. It is expected to provide outstanding performance for large-scale real-world applications.

3. Significance as a JHPCN Joint Research Project

JHPCN provides a variety of supercomputer systems. We can develop and optimize our programs on each platform very easily under collaboration with members of the JHPCN centers. We can deploy our programs and libraries on the supercomputer systems in each center, and release to the public. Improvement of such released programs are accelerated if they are used by users of supercomputer systems for practical scientific and engineering applications. This is an international joint proposal by Germany-Japan-USA including experts of multigrid method and HPC in each country. MOU (memorandum of understanding) for collaborative research has been exchanged between Lawrence Berkeley National Laboratory (LBNL) and Information Technology Center, The University of Tokyo

(ITC/U.Tokyo) since September 2009, and between University of Wuppertal and ITC/U.Tokyo since July 2017. FAU and ITC/U.Tokyo have been collaborating in ESSEX-II Project of German SPPEXA Program by DFG/JST since 2016. Moreover, this proposal includes most of the experts in multigrid method in Japan.

4. Outline of the Research Achievements up to FY 2018

This is a 2-year project in FY.2018 and FY.2019, where we do fundamental research & development, code optimization and preliminary evaluation of performance and robustness in the first year, and performance evaluation using real-world large-scale applications in the second year. In FY.2018, we had many excellent developments which were not planned in the beginning of the project, such as (1) Parallel Aggregation, (2) Iterative Solver with MGRIT Preconditioning, (3) PinST-Exp/Imp Method, (4) Evaluation of Efficiency and Accuracy by Computations with Single Precision. We have also done the preliminary works on AM-*h*CGA, which were originally planned in FY.2019.

① GMG

Target program of GMG is pGW3D-FVM for 3D Groundwater Flow through Heterogenous Porous Media by FVM (Finite Volume Method) in Fig.1.

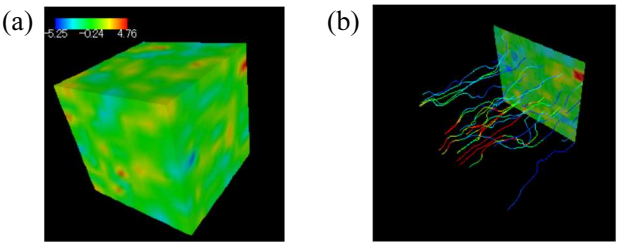


Fig.1 Example of groundwater flow through heterogeneous porous media. (a) Distribution of water conductivity; (b) Streamlines

In the 1st Year (FY. 2018), we have conducted, (1) Optimization of pGW3D-FVM on OFP by CM-RCM and SELL-C-σ, (2) Improvement of Convergence and Efficiency by New Smoother (MS-BMC-GS, Multiplicative-Schwartz type Block Multicolor Gauss-Seidel), (3) Optimization of Mesh Generation by IME (Fast File Cache on the Oakforest-PACS (OFP)) on OFP, (4) Preliminary Study of Adaptive Multilevel *h*CGA (AM-*h*CGA) and (5) Evaluation by 3D GW code on the supercomputers.

We evaluated the performance of pGW3D-FVM code with *h*CGA on the Oakforest-PACS (OFP) system at JCAHPC/The University of Tokyo. Fig. 2 describes the results of evaluation up to 8,192 nodes (524,288 cores, because 64 of 68 cores were used on each node). *h*CGA was also very effective on OFP. The prototype of AM-*h*CGA (Adaptive Multilevel *h*CGA) was also developed, although that was originally planned to be developed in FY.2019. Original pGW3D-FVM adopts RCM reordering for thread parallel computation, but it is not suitable for manycore architectures. We have applied CM-RCM to the code, and CM-RCM with 2-colors is the most efficient on OFP with Intel Xeon Phi (Knights Landing, KNL).

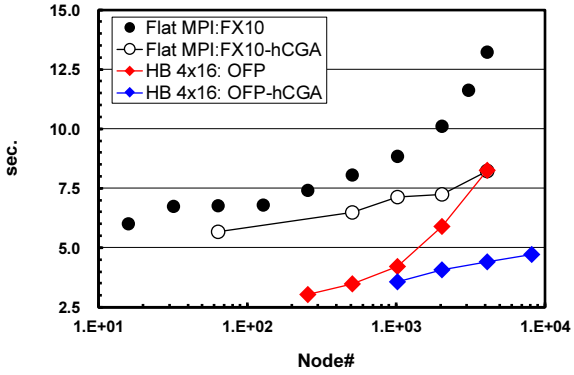


Fig.2 Performance of MGCG solver of pGW3D-FVM on Fujitsu FX10 using up to 4,096 nodes (65,536 cores), and OFP up to 8,192 nodes (524,288 cores), weak scaling: max. total problem size: 17B meshes on FX10, and 35B on OFP, RCM reordering [4]

② AMG

Target program of GMG is GeoFEM SA-AMG for 3D Solid Mechanics by FEM. In the 1st Year (FY.2018), we have conducted, (1) Parallelization of GeoFEM SA-AMG by OpenMP/MPI Hybrid, (2) Improvement of

Convergence and Efficiency by MS-BMC-GS, (3) Improvement of Convergence by Parallel Reordering/Aggregation, (4) Development and Implementation of New Strategies for Extraction of Near-Kernel Vectors, and (5) Evaluation by 3D FEM on the Supercomputers.

As the first step, we investigate the execution time and parallel efficiency by applying Hybrid parallelization. The relaxation of the solution part was performed twice per level by Symmetric Gauss-Seidel. Moreover, CM-RCM was applied at each level.

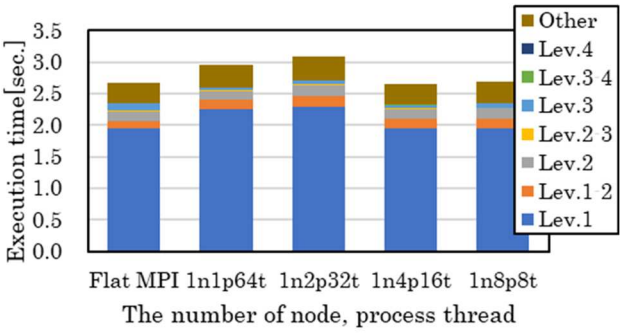


Fig.3 execution time on OFP (The ratio of V-cycle)

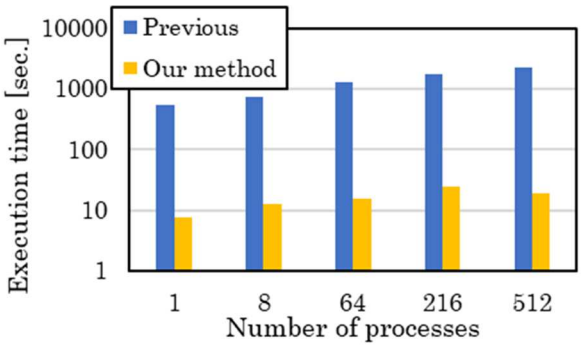


Fig.4 The result of time to verify the optimum number of near-kernel vectors

Fig. 3 is the result of execution time on OFP. Hybrid (1n4p16) is almost the same as Flat MPI. Moreover, the calculation time of the V-cycle, especially smoother, accounts for the majority of whole execution time. On the other hand, the communication cost is small. This is because the number of nodes is small in this experiment. We will investigate this issue in highly parallel environment. In SA-AMG, the way how to set the near-kernel vector is very important to achieve good convergence and scalability. Because our previous method presented in SC17 was very expensive, we proposed a more efficient method in the present work. Fig.4 shows the verification time to set the optimum number. The verification was time significantly reduced by the prediction method. From these results, our method can predict the optimum number of near-kernel vectors easily.

③ PinST-TSC

Target program of PinST-TSC is pHEAT-3D for 3D Nonlinear Heat Transfer by FEM. In the 1st Year (FY.2018), we have conducted (1) Implementation of PinST-TSC to pHEAT-3D and (2) Evaluation of the Performance on Supercomputers.

④ PinST-Exp

In the 1st Year (FY.2018), we developed (1) Parallel Iterative Solver Preconditioned with MGRIT, (2) New PinST for Applications with Explicit Time-Marching (PinST-Exp/Imp) and are conducting (3) Implementation of PinST-Exp/Imp to pHEAT-3D.

1) MGRIT-based Preconditioning

We investigated parallelization in time direction especially for explicit methods. We took up the MGRIT method for parallelization of explicit time evolution, and proposed to use MGRIT as a preconditioner of Krylov subspace method.

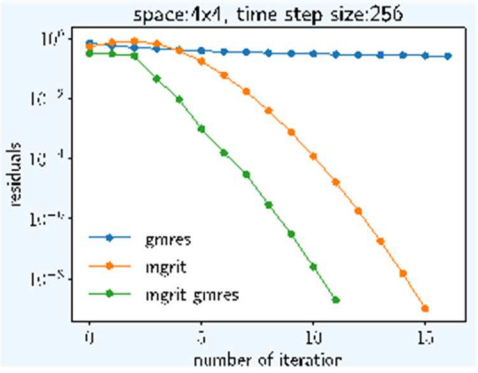


Fig.5 History of Convergence using OFP

Fig.5 shows the residual history of the proposed

method, MGRIT, and non-preconditioned GMRES using a single node of OFP. MGRIT preconditioned GMRES reached convergence faster than the other 2 methods, and it exemplified that the coarse grid problems' instability can be soothed by using MGRIT as a preconditioner.

2) Hybrid Explicit/Implicit Method

If PinST is applied to applications with explicit time-marching, stability problem occurs at coarser levels in time direction. In order to avoid this situation, we proposed a new method PinST-Exp/Imp, where explicit time marching is applied to the finest level in time direction, and implicit ones for other coarser levels. Fig. 6 shows the results (strong scaling) for 2D heat transfer problem using 4 nodes of OFP. The orange line (Exp/Imp) provides the faster computation time than Exp/Exp and Imp/Imp. This is one of the first examples for PinST computations with explicit time-marching.

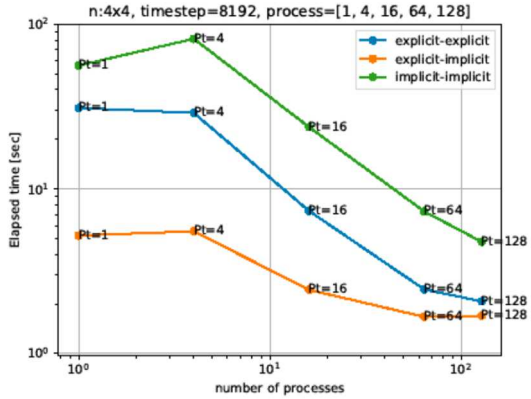


Fig.6 Effect of PinSt-Exp/Imp, Strong Scaling up to 4 Nodes of OFP

⑤ Parallel Aggregation

We proposed a parallelization method with multi-coloring for parallel aggregation in the AMG. The performance and convergence of the AMG strongly depend on the results of aggregation. We proposed a parallelization method of the aggregation process with the multi-coloring method. By this approach, compared with the parallel and sequential aggregation, we achieved constant convergence. Fig.7 shows the comparison between the sequential/decoupling aggregation and the proposed method (multi-coloring) for Parabolic FEM problem in "SuiteSparse Matrix Collection". If we apply the decoupling aggregation, the convergence varied according to the degree of parallelisms. As we expected, the parallel aggregation with multi-coloring kept almost constant convergence even if the degree of parallelism changed. In the future, we will evaluate the performance of the AMG with the parallel aggregation and MS-BMC-GS smoother on the massively parallel systems.

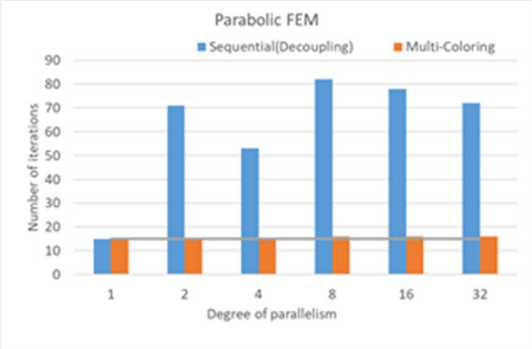


Fig.7 Effect of Parallel Aggregation with Multi-Coloring on Parabolic FEM Problem (Iterations of AMG Solver)

5. Details of FY 2019 Research Achievements

① Overview

In FY.2019, our plan for development in this project is as follows:

- Geometric Multigrid (GMG) for 3D Goundwater Flow
  - ➢ Further Optimization of the pGW3D-FVM on OFP (CM-RCM, SELL-C-σ, MS-BMC-GS) (Hoshino, Hanawa, Nakajima, Wellein, Kawai, Ida, Bolten, Claus)
  - ➢ Improvement of Convergence by Parallel Reordering (Kawai, Nakajima)
  - ➢ Improvement of Convergence by Utilization of Near-Kernel Vectors (Nomura, Bolten, Claus, Nakajima),
  - ➢ Further Optimization of Mesh Generation by IME on OFP (Hanawa, Nakajima)

  - Improvement of Scalability by Adaptive Multilevel *h*CGA (AM-*h*CGA) (Nakajima)
  - Evaluation by 3D GW Code on the Supercomputers (Nakajima).
- Algebraic Multigrid (AMG, SA-AMG) for 3D Solid Mechanics
  - Optimization of GeoFEM SA-AMG on OFP using ELL, SELL-C-σ (Hoshino, Hanawa, Nakajima, Nomura, Wellein)
  - Further Improvement of the Algorithm by MS-BMC-GS (Kawai, Ida, Nomura, Marques, Bolten, Claus)
  - Further Improvement of the Convergence by Parallel Reordering/Aggregation (Kawai, Marques, Bolten, Claus, Nomura)
  - Implementation of *h*CGA and MA-*h*CGA for Scalability (Nakajima, Nomura)
  - Evaluation by Supercomputers (Nomura, Nakajima).
- PinST-TSC
  - Further optimization of the pHEAT-3D (Fujii, Iwashita, Takahashi, Nakajima, Hoshino)
  - Implementation of Multigrid Solver (SA-AMG) in Space Domain for Scalable Computation (Nomura, Fujii)
  - Final Evaluations on the Supercomputers (Fujii, Iwashita).
- PinST-Explicit
  - Further Optimization of the MGRIT Preconditioner (Yoda, Fujii)
  - Further Optimization of pHEAT-3D with PinST Exp/Inp (Yoda, Fujii, Nakajima)
  - Development of hybNS for 3D Compressible Navier-Stokes Flow by FVM with PinST-Exp (Chen, Nakajima)
  - Final Evaluations on the Supercomputers (Yoda, Fujii)
- Parallel Algorithms
  - MS-BMC-GS
  - Parallel Reordering/Coloring
  - Parallel Aggregation

In the following part, we will describe the activities on these issues.

② AM-*h*CGA with IHK/McKernel [2,7,8,9,10,11,14,17]

The parallel multigrid method is expected to play an important role in large-scale scientific computing on exa-scale supercomputer systems. Previously we proposed Hierarchical Coarse Grid Aggregation (*h*CGA, Fig.8), which dramatically improved the performance of the parallel multigrid solver when the number of MPI processes was O(10^4) or more.

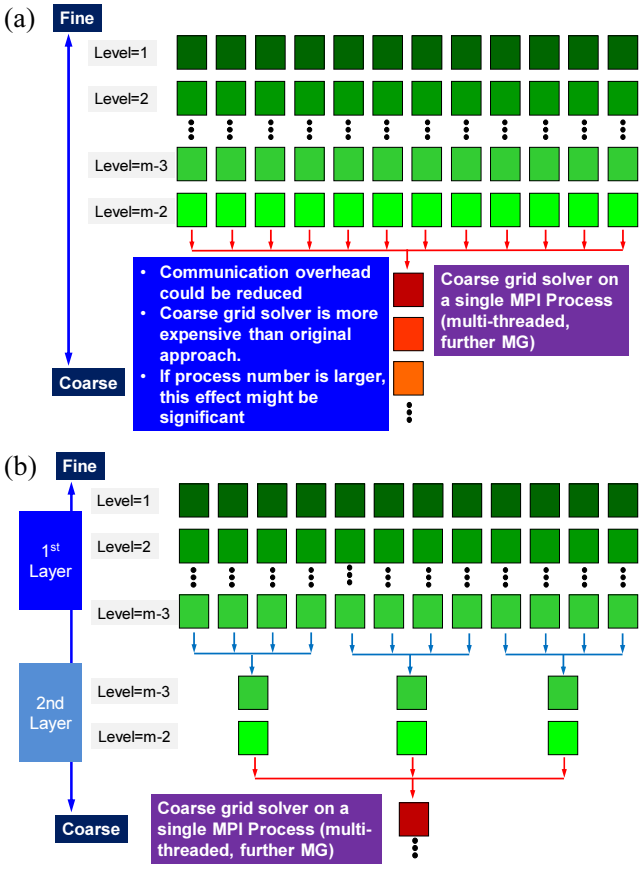


Fig.8 Procedures for Parallel Multigrid (a) *Coarse Grid Aggregation* (CGA), where information of each MPI process is gathered in a single MPI process for computation at *level=m-2*, (b) *Hierarchical* CGA (*h*CGA) [4]

Because *h*CGA can handle only two layers of parallel

hierarchical levels, the computation overhead due to coarse grid solver may become significant when the number of MPI processes reaches O(10^5)- O(10^6) or more. In the present work, we propose AM-*h*CGA (Adaptive Multilevel *h*CGA) that can take into account multiple layers of three or more levels, and show preliminary results using the Oakforest-PACS (OFP) system by JCAHPC. Additionally, we also examine the impact of a lightweight multi-kernel operating system, called IHK/McKernel, for parallel multigrid solvers running on OFP.

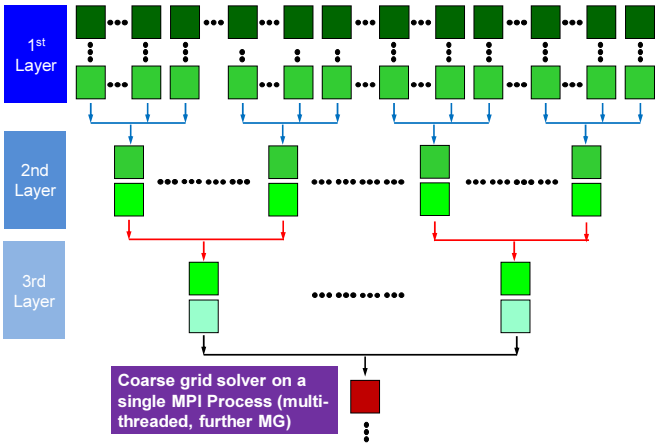


Fig.9 Overview of AM-*h*CGA (Adaptive Multilevel-*h*CGA) with 3 hierarchical layers

The previous work by the authors [Nakajima, K., 2014 IEEE ICPADS] showed that the *h*CGA can avoid the overhead of coarse grid solver even when the number of MPI processes increases. If the number of MPI processes is more than O(10^4), the *h*CGA method is considered effective. However, when the number of processes is on the order of O (10^5) -O (10^6), the two layers of *h*CGA shown in Fig. 8(b) are not enough, because the number of MPI processes at the second level of *h*CGA could be O(10^4). Therefore, it is necessary to increase the number of layers of *parallel hierarchical levels* to 3 or more in order to reduce the overhead by coarse grid solver if the number of MPI processes is O (10^5), or more. In this study, we propose an Adaptive Multilevel *h*CGA (AM-*h*CGA) method with an increased number of hierarchical levels as shown in Fig. 9.

We performed weak scaling experiments on pGW3D-FVM (Fig.1) using up to 2,048 nodes (131,072 cores) of the OFP. The summary of the computation is as follows:

① Elapsed computation time of MGCG solver was evaluated using CGA, *h*CGA, and AM-*h*CGA (Fig.12) on OFP. The number of nodes was 128, 256, 512, 1,024, and 2,048.

② The following two types of problem sizes were implemented:

- **Medium (m)**: 64×32×32 (65,536) per core, 4,194,304 per node, maximum 8,589,934,592 for 2,048 nodes
- **Small (s)**: 32×16×16 (8,192) per core,524,288 per node, max 1,072,741,824 for 2,048 nodes
- **Tiny (t)**: 16×8×8 (1,024) per core, 65,536 per node, max 134,217,728 for 2,048 nodes

③ Flat MPI is applied for all cases (with -qopenmp option).

④ Measurements are done 5 times for each case, and the best timing was adopted.

Fig.10 shows relative fluctuation between maximum and minimum measured time. Although the behavior is generally random, the relative fluctuation is increasing as core number, and it is 20% -70% at 2,048 nodes.

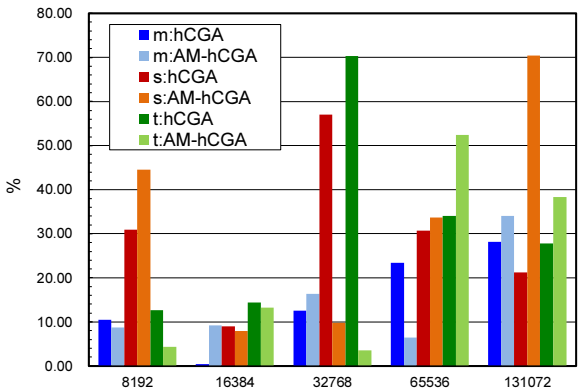


Fig. 10 Relative time fluctuation for 5 measurements, 100×(Max.Time-MinTime)/Min.Time

The relative fluctuation of computation time in Fig.10 is presumably due to various types of OS noise, such as job scheduler and file system activities. On the OFP system, the *period of timer interrupt* of the first two cores (0^{th} and 1^{st}) on the first *tiles* are set to 1kHz, while timer interrupts on the other 66 cores are disabled. Therefore, interference by the OS can be reduced, if 0^{th} and 1^{st} cores are not used for the application, as we do in the present work. Although major processes related to OS and job scheduler are pinned to 0^{th} and 1^{st} cores, other Linux kernel components (e.g., certain kernel threads) are not necessarily bound to the OS cores. Generally, the effect of such noise becomes larger, as the number of cores or MPI processes increases, and the problem size per core becomes smaller. The relative fluctuation sometimes reaches more than 50%, as shown in Fig.10. Under these conditions, proper evaluation of the performance of AM-*h*CGA at 2,048 nodes is difficult.

IHK/McKernel is a lightweight multi-kernel operating system designed for high-end supercomputing, and is developed by RIKEN R-CCS. There are two main components of the software stack, a low-level software infrastructure, called Interface for Heterogeneous Kernels (IHK) and a lightweight co-kernel called McKernel. McKernel is a lightweight co-kernel developed on top of IHK. It is designed explicitly for high-performance computing workloads, but it retains a Linux compatible application binary interface (ABI) so that it can execute unmodified Linux binaries. There is no need for recompiling applications or for any McKernel specific libraries. McKernel implements only a small set of performance sensitive system calls and the rest of the OS services are delegated to Linux. Specifically, McKernel provides its own memory management, it supports processes and multi-threading, it has a simple round-robin co-operative (tickless) scheduler, and it implements standard POSIX signaling. It also implements inter-process memory mappings and it offers interfaces for accessing hardware performance counters. Applications are executed on the lightweight kernel, and only performance insensitive operations are offloaded to Linux, therefore OS noise is significantly reduced. In particular, the performance of collective communication by IHK/McKernel is better than that of Linux on OFP. Finally, many applications attained efficient and robust performance on OFP with may nodes.

Fig.11 shows the relative time fluctuation for 5 measurements of *h*CGA and AM-*h*CGA with IHK/McKernel. Although large values are still measured in several cases, relative time fluctuation is much smaller than the cases without IHK/McKernel in Fig.16. Especially, it is less than 3.20% at 131,072 cores (2,048 nodes). Therefore, we can expect proper evaluation of performance of *h*CGA and AM-*h*CGA.

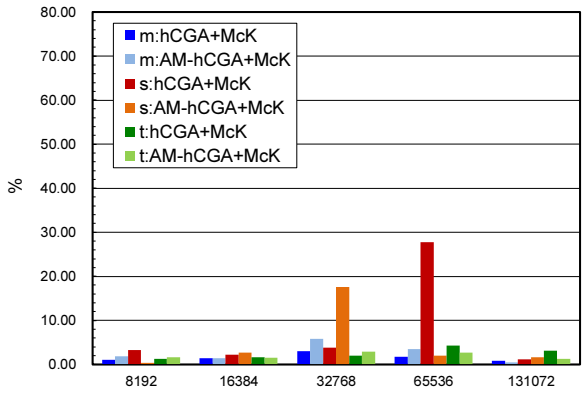


Fig. 12 Relative time fluctuation for 5 measurements, with IHK/McKernel: 100×(Max.Time-MinTime)/Min.Time

Each plate of Fig.12 (a)-(c) shows computation time for weak scaling of optimum cases for *h*CGA, and AM-*h*CGA, with and without IHK/Mckernel. Computation time for MGCG is normalized by the time of optimum case of *h*CGA without IHK/McKernel at each core number. Therefore, the values of Y-axis (*Ratio*) are always equal to 1.00 for *h*CGA without IHK/McKernel. Effects of IHK/McKernel for improvement of the performance of *h*CGA at 131,072 cores (2,048 nodes) is

approximately 4% for Medium (m), 17% for Small (s), and 22% for Tiny (t). If the results of *h*CGA+Mck (○), and AM-*h*CGA+McK (◆) in Fig.12(a)-(c) are considered, improvement is very slight for Medium (1.50%), and Small (0.34%) at 131,072 cores. But the improvement of performance by AM-*h*CGA over *h*CGA for Tiny case at 131,072 cores is 7.81%.

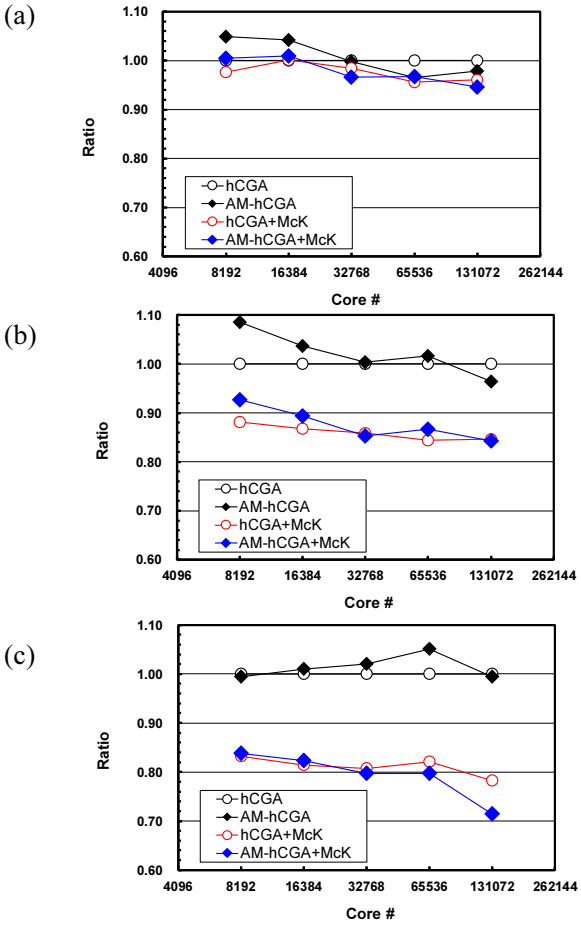


Fig.12 Computation time for MGCG solvers up to 2,048 nodes (131,072 cores) of OFP, normalized by computation time for *h*CGA at each core number, *h*CGA/AM-*h*CGA: without IHK/McKernel, *h*CGA/AM-*h*CGA+McK: with IHK/McKernel, (a) Medium (m), (b) Small (s), (c)Tiny (t)

In this work we have proposed AM-*h*CGA, which can take into account multiple layers of three or more parallel hierarchical levels for large-scale computation with $O(10^5)$-$O(10^6)$ or more MPI processes. We have implemented the proposed method to the MGCG solvers of pGW3D-FVM, and evaluated the efficiency and robustness of the developed method on up to 2,048 nodes (131,072 cores) of the OFP system. On OFP, fluctuation of computation time due to various types of OS noise on Linux, occurs. The relative ratio of fluctuation was more than 50% in this study. Under these conditions, proper evaluation of the performance of the proposed method (AM-*h*CGA) at 2,048 nodes is difficult. We deployed IHK/McKernel, a lightweight multi-kernel OS developed at RIKEN R-CCS, which provides efficient and scalable execution environment on large-scale systems by reducing OS noise and communication overhead. IHK/McKernel reduced the fluctuation of computation time to less than 10% of the original case using Linux. The maximum ratio of performance improvement by IHK/McKernel is more than 20% for MGCG solver by *h*CGA using 2,048 nodes of OFP. IHK/McKernel is especially effective for reduction of overhead by MPI communications, for which the maximum ratio of improvement was 35%. Finally, it was proved that AM-*h*CGA is faster than *h*CGA if the number of MPI processes is large, and problem size per process is small, and the maximum improvement of performance by AM-*h*CGA over *h*CGA is 7.81% at 2,048 nodes of OFP using IHK/McKernel. We demonstrated that IHK/McKernel is a powerful software for performance evaluation on large-scale supercomputer systems. Currently, vectorization is not enough. Further optimization using more sophisticated method for vectorization, such as SELL-C-σ, will be investigated.

③ AMG for 3D Solid Mechanics: Near Kernel Vectors [1,19,22]

In the smoothed aggregation algebraic multigrid (SA-AMG) method, the way how to set the near-kernel vector is very important things to achieve good convergence and scalability. In the present work, we propose a new extraction method more efficiently.

SA-AMG is efficient and scalable for solving large-scale linear equations. SA-AMG achieves good

convergence and scalability by damping various wavelength components efficiently. To achieve this damping, SA-AMG creates multi-level matrices which are hierarchically coarser than the original matrix. Moreover, the convergence of the method can be further improved by setting near-kernel vectors defined as nonzero vector $\boldsymbol{p}$ satisfying $A\boldsymbol{p} \approx 0$.

There are several existing works on near-kernel vector extraction method. In our previous work, we proposed a method that extracts near-kernel vectors and add them at each level, and evaluated the robustness and efficiency [1]. By using our method, the convergence and execution time is improved. However, the extraction cost is expensive compared to the solution time of linear equations.

To reduce this cost, in the present work, we propose a new extraction method that extract near-kernel vectors at coarse level using eigenvalue algorithm. Moreover, to set the near-kernel vectors at each level, this method interpolates extracted vectors to finer levels.

```
1  Given: L̂, k_e
2  For l = L̂ to 1
3    if l == L̂ then
4      W_L = {w_l^κ | A_l w_l^κ = λ_l^κ w_l^κ, κ = 1,.., k_e}
5    else
6      w_l^κ = P_{l+1}^l w_{l+1}^κ, λ_l^κ = λ_{l+1}^κ, κ = 1, ..., k_e
7      for κ = 1 to k_e do
8        Relax on A_l w_l^κ = 0
9      End for
10   End if
11 End for
12 Output W_1 as the near_kernel vectors
```

- $\hat{L}$: The level number to apply eigensolver
- k_e: Number of extracted near-kernel vectors
- A_l: Matrix A of level l
- P_{l+1}^{l}: A matrix to interpolate level $l+1$ to l

Fig.13 The outline of the extraction method

```
1  Given: L̂, W_1, ..., W_L̂
2  For l = 1 to L̂
3    Multilevel_creation(W_l)
4    Ŵ_{l+1} = R_l^{l+1} W_l
5  End for
6  Output Ŵ_2, ..., Ŵ_L̂ as the near_kernel vectors
```

- R_l^{l+1}: A matrix to interpolate level $l+1$ to l
- $Multilevel\_creation(W_l)$: Recreation of the interpolate matrices and coarser level matrices by using W_l.

Fig.14 The outline of the way to set the near-kernel vectors to coarser levels for proposed method

Fig.13 shows the outline of proposed extraction method of only level 1. This method needs to input the level number $\hat{L}$ ($\hat{L}$ is set to 2 in this study). Eigenvalue solver is applied at $\hat{L}$ (Line 4). After that, calculated eigenvectors close to 0 eigenvalues is interpolated by using P matrix to fine level (Line 6). To set coarser level near-kernel vectors, Fig.14 is applied after Fig.13.

In this study, we investigate the effectiveness of proposed method by comparing previous method in 3d elastic problem. Table 1 shows the target of comparison. From Fig.15, the extraction cost of proposed method can be reduced compared to previous method. Moreover, From Fig.16 and 17, proposed method is the same good convergence as previous methods. As above, our proposed method can extract near-kernel vectors at low cost compared to previous method.

Table1 The target of comparison in this experiment

Candidate	Details
Previous 1	Proposed at [Nomura, N. et al., VECPAR2016, 2016]
Previous 2	Proposed at [Nomura, N. et al., ACS65, 2019] [1]
Proposed	Using Fig.13 and Fig.14 method

Moreover, to investigate the effectiveness for other problem settings, we apply to various problems from SuiteSparse Matrix Collection (https://sparse.tamu.edu /). Table2 shows the list of problems. Table3 and Table4

show the result of extraction time and iterations, respectively. From these tables, proposed method can also realize the low-cost extraction and good convergence.

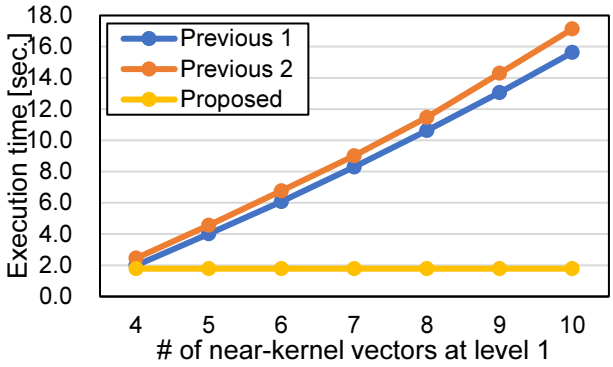


Fig.15 The extraction time at single process case

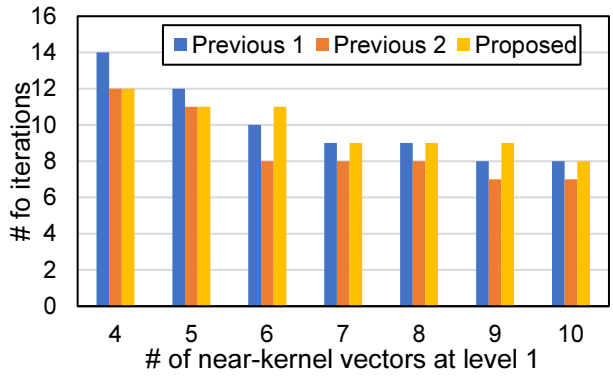


Fig.16 The number of iterations at single process case

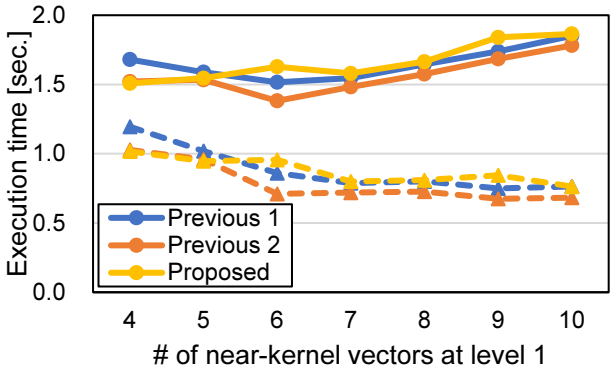


Fig.17 The execution time at single process (The solid lines (round marker) show the whole execution time (setup + solution part), and the dashed lines (triangle marker) show the execution time of only solution part.

From these results, in this experiment, proposed method is effective way to extract near-kernel vectors at low cost. Proposed method has some input parameter (such as $\hat{L}, k_e$). Because these parameters influence the performance, we will investigate the way to set the suitable parameter. Moreover, our goal is development of robust and efficient solver for any problems. Therefore, we must improve the extraction method and investigate at other ill-conditioned problems.

Table2 The list of problem matrix information

Name	# of rows	# of NNZ	Condition Number
ex10	2,410	54,840	9.10e+11
bcsstk28	4,410	219,024	9.45e+08
s1rmq4m1	5,489	262,411	1.81e+06
af_shell	504,855	17,562,051	—

Table3 The extraction time at various problem test

Name	Previous 2	Proposed
ex10	9.08e-01	4.44e-02
bcsstk28	2.85e+00	5.17e-02
s1rmq4m1	4.37e+00	6.67e-02
af_shell	2.87e+02	1.31e+03

Table4 The number of iteration at various problem test (SGS: Symmetric Gauss-Seidel preconditioned CG, “af_shell” - “SGS” is not converged)

Name	SGS	Previous 2	Proposed
ex10	369	36	48
bcsstk28	1543	355	475
s1rmq4m1	282	29	72
af_shell	—	86	157

④ PinST-TSC [2]

We have continued development of PinST algorithms for nonlinear problems. In order to accelerate the convergence, many parallel time integration algorithms such as parareal and MGRIT use “coarse grid” simulation with enlarged time step width to propagate prior time step information to later time steps at a fast pace but with low accuracy. However, problems with enlarged time step width tend to cause instability, which leads to difficulty with the time integration. On the other hand, the parallel time integration method, parallel TP-EEC method (Time-Periodic Explicit Error Correction), that does not introduce re-discretization with enlarged time step width was also known for time-periodic

nonlinear magnetic field problems. It uses the Jacobian matrix information to calculate rough solution correction.

We extended the correction scheme of the TP-EEC method to deal with multi-level structure more than one coarse level, and applied it to an ordinary nonlinear time integral simulation problem. We call it TSC (Time Segment Correction) method. In addition, we studied its implementation method and checked the effectiveness in comparison with the MGRIT method.

Our implementation method is described in Fig.18. Fine level corresponds to usual nonlinear time integration problem, and the coarse level is linear problem calculated from Jacobian matrices over all time steps. Here, the coarse level matrix becomes block lower triangular form, and is easily solved by forward substitution. TSC-loop3 corresponds to nearly 3 iterations of TSP-loop1, and coarse level corrections of a TSC-loop3 iteration can be calculated in pipelined manner, when time steps are block distribution to processes.

Fig.19 shows the execution time of each method when the problem size corresponding to the number of time steps is enlarged. The problem is 2-dimensional heat diffusion equation with nonlinear diffusion coefficient. It shows that MGRIT shortens the execution time in comparison with the step-by-step method for problems with larger than 8,192 time-steps in this numerical test setting. The TSC methods reached convergence faster than the step-by-step method for all problems including small sized problems. In comparison with the MGRIT method, TSC methods are faster than the MGRIT method for almost all problems. The TSC loop3 was the fastest.

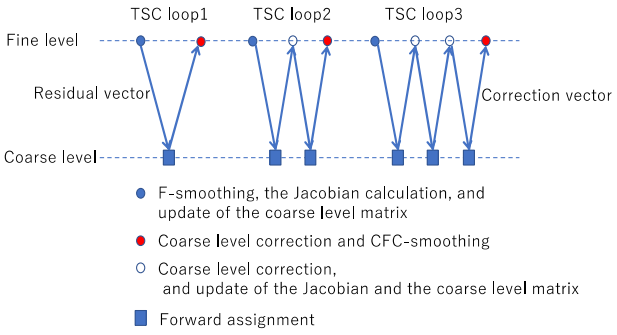


Fig. 18 Cycle shapes: TSC loop 1,2 and 3

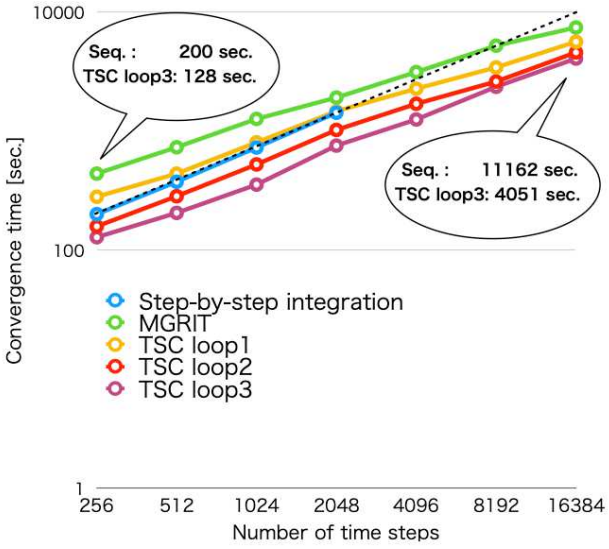


Fig.19 Execution time with different number of time steps

As for applicability to existing application codes, the TSC method uses only Jacobian matrices for calculation of the coarse level problem. Therefore, the TSC method becomes “nonintrusive” method for applications that can output Jacobian matrices. We will increase the evaluation cases in various simulation fields.

⑤ PinST-Explicit

1) MGRIT Preconditioning [3]

We proposed the MGRIT preconditioning to improve the convergence and stability of PinST solvers in 2018. In 2019, we applied the MGRIT preconditioning to a one-dimensional linear mass-damper system and evaluated it on Oakforest-PACS. The Newmark-beta method was used to discretize the governing equations. Fig.18 shows the execution time of the PinST solvers

with 2^{20} (about one-million) time steps and 48 mass points, using the acceleration data of Tokai earthquake observed at the Kogakuin University. We compared the time-marching method, MGRIT, and MGRIT preconditioned GMRES (MGRIT-GMRES). This experiment is performed with weak scaling. The time-step width is fixed and the time-range increases with the number of time steps. The labels in Fig.20 indicate the number of iterations for each solver. As the number of time steps increased, the number of iterations of the conventional method, MGRIT, significantly increased, exceeding the execution time of the time-marching method. This result may be due to the instability caused by the enlarged time-step with. In contrast, our proposed method keeps a constant number of iterations and reduces the execution time by about 5.98 times at the most. We confirmed that the MGRIT preconditioning improves the convergence of PinST solvers and is also effective in terms of execution time.

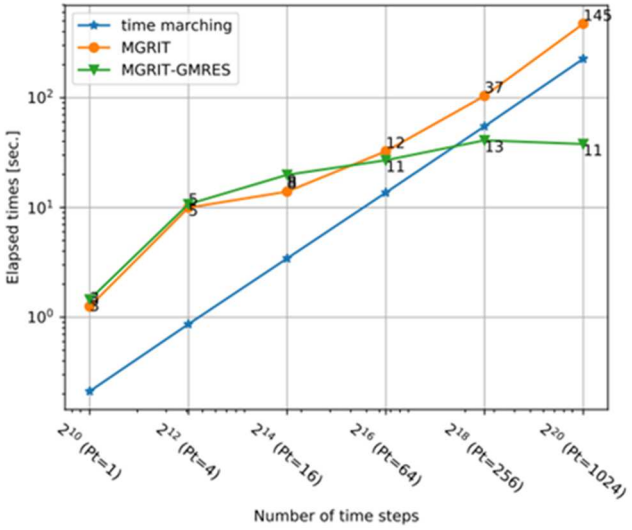


Fig.20 Elapsed times in a weak scaling

2) PinST-Exp/Imp

In 2018, we proposed the PinST-Exp/Imp, which discretizes the fine-level by an explicit method and the coarse-level by an implicit method. We observed that the convergence deteriorates and the number of iterations increases as the fine-level parameter approaches the CFL condition in the linear diffusion problem. We tried to improve the convergence by using MGRIT of PinST-Exp/Imp as a preconditioner. Table 5 shows the number of iterations in a two-dimensional linear diffusion problem with a fixed analysis region by varying the time-step width and spatial mesh width. Since the CFL condition $c=\frac{\Delta t}{\Delta x^2}$ for this problem is 0.25, it can be seen that the number of iterations of MGRIT increases as the ratio c approaches 0.25. In contrast, MGRIT preconditioning reduces the number of iterations from 402 to 329 at most. We confirmed that the MGRIT preconditioning is also effective in PinST-Exp/Imp in terms of the number of iterations.

Table 5 Number of iterations with various diffusion coefficients.

$N_x = 16$	N_t	256	128	120	116	114
	$c = \Delta t/\Delta x^2$	0.110	0.221	0.236	0.245	0.249
	MGRIT	12	13	23	38	49
	MGRIT-GMRES	12	13	22	35	44
$N_x = 32$	N_t	1024	768	512	484	482
	$c = \Delta t/\Delta x^2$	0.117	0.157	0.241	0.2487	0.2497
	MGRIT	12	12	39	98	137
	MGRIT-GMRES	12	12	29	87	120
$N_x = 64$	N_t	4096	2048	2000	1988	1986
	$c = \Delta t/\Delta x^2$	0.121	0.242	0.248	0.2496	0.2499
	MGRIT	12	36	115	292	402
	MGRIT-GMRES	12	33	110	244	329

3) PinST-Explicit for 1D Advection [20]

We propose a multilevel parareal PinST (Parallel-in-Space/Time) method that achieve scalability better than that of the spatial parallelization method. The multilevel parareal method was proven to achieve reasonable approximation results for the one-dimensional advection problem faster than that of a regular spatial parallelization method with more than 8 processors.

Our target problem is to develop a PinST method for explicit time-stepping schemes. There are two main challenges to develop a PinST method for explicit time-stepping schemes. First, explicit schemes have to satisfy the CFL (Courant-Friedrichs-Lewy) condition in order to converge. Secondly, since spatial parallelization is very efficient with explicit schemes, it is difficult to leverage more or even similar scalability with PinST

methods.

The proposed multilevel parareal method is based on the parareal method with the hierarchy of the MGRIT (Multigrid Reduction-in-Time) method. The idea of the multilevel parareal method is to achieve good initial values from coarser levels and reduce the iteration number of the parareal method in each level.

First, we construct multiple level similar to the MGRIT method. Each level has a different time step size for explicit time-stepping. The coarser girds have larger time step sizes and the finer grids have smaller ones. In order to satisfy the CFL condition, we also coarsen the space grid with the same ratio of the time step size changes.

Since we are coarsening the space grid, we have to define the restriction and prolongation operators. We define the restriction by picking up coarse points directly, similar to the multigrid method, and we define the prolongation by interpolating the update values of neighbor coarse points.

Start from the coarsest two levels, we perform the parareal method iterations until convergence. Then we move to a finer level, perform the parareal iteration again until convergence. Similarly until the parareal iteration in the finest grid is converged.

Algorithm 2: Multilevel parareal (MPI parallelized)

Explicit time-marching on the coarsest level L.
for *level* $l = L\text{-}1$ *to* 1 **do**
  for *iterate until residual tolerance* **do**
    On current processor:
    Solve on the current level $y_f = \mathcal{F}_l(y_j, t_j, t_{j+1})$.
    Solve on the coarsest level $y_c = \mathcal{G}(y_j, t_j, t_{j+1})$.
    for *Processor* $p = 1$ *to* P **do**
      Solve on the coarsest level
      $y_{j+1}^{k+1} = \mathcal{G}(y_j^{k+1}, t_j, t_{j+1}) + \mathcal{F}_l(y_j^k, t_j, t_{j+1}) - \mathcal{G}(y_j^k, t_j, t_{j+1})$
    end
  end
end

Fig.21 The algorithm of the proposed multilevel parareal method. The red functions are the low precision solvers and the blue ones are the high precision solvers of the parareal algorithm.

In order to further reduce the computation time, we set smaller tolerance for the parareal iteration at coarser grids, where the computation is cheaper.

Even with small iteration numbers, the proposed multilevel parareal method has larger computation complexity, and slightly larger communication cost. However, with many time steps, the multilevel parareal method has much fewer number of synchronizations to perform compare to that of the spatial parallelization method.

We apply the multilevel parareal method to the one-dimensional advection example with the Lax-Wendroff method as the explicit time-marching schemes. From the numerical experiment we found that we could derive decent results with not so small tolerances (average tolerance 0.3) the error mainly occurs at the discontinuous points, which is of our least interest.

We implement the code using C++ with MPI parallelization. We conduct runtime experiments on Oakbridge-CX supercomputer in the Information Technology Center, the University of Tokyo. We compare runtime results of our PinST method and the spatial parallelization method with similar number of space points and number of time steps.

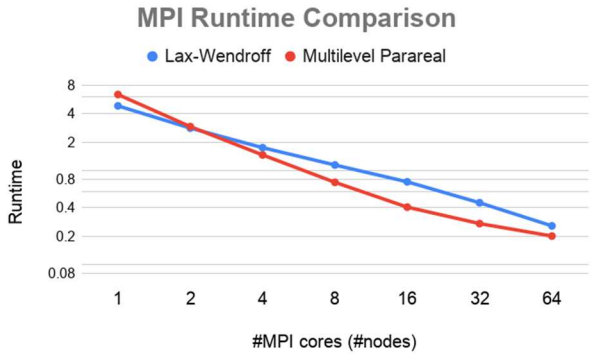


Fig.22 MPI scability comparison between sequential Lax-Wendroff method (parallelized in space domain) and multilevel parareal method (parallelized in time domain).

Our method prove that it is possible to leverage similar or even better scalability with PinST methods. However, our current method could serve only as a fast approximation. To achieve high precision as other PinST methods, smaller tolerance and longer runtime is expected. For future work, we would like to further improve the robustness and the convergence of our

method.

4) PinST-Explicit for 2D Compressible CFD [20]

We are working on applying the proposed multilevel parareal PinST method on two-dimension compressible CFD examples with explicit time-marching schemes. We expect that two-dimension examples would perform better as spatial parallelization gets more complicated than that of the one-dimension examples.

As a first example, we are solving a compressible flow around a circle. We solve the Navier-Stokes equations (the density equation, the momentum equation and the energy equation) with Lax-Wendroff method as explicit time-marching scheme. We apply time-stepping on each node values considering a dual mesh around each node.

We have currently completed the code implementation of the sequential solver and the MPI parallelization of the sequential solver. We are moving on to the PinST implementation for the two-dimensional CFD example.

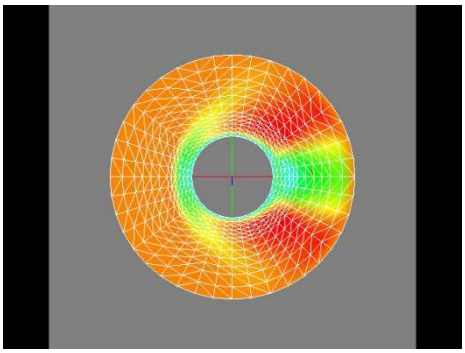


Fig.23 Results of a supersonic simulation with 8,346 nodes and 38,400 meshes, Distribution of Mach number, M_{inf}= 1.40, Re=10^3

⑥ MS-BMC-GS [5]

We propose a multiplicative Schwartz type block multi-color Gauss-Seidel (MS-BMC-GS) [5] smoother for the AMG method. The MS-BMC-GS smoother improves the convergence rate and data-locality of the AMG method. The smoother in AMG determines the convergence rate and computational time of the AMG method. A hybrid approach, which is using Gauss-Seidel and weighted-Jacobi, and multi-colored Gauss-Seidel is widely used as the smoother. However, these smoother shows lower convergence rate or data-locality compared with sequential Gauss-Seidel smoother. A block multi-color Gauss-Seidel (Fig.24) is one of the effective smoother that solve the problems of existing smoother, and it shows the same convergence rate and data-locality compared with sequential Gauss-Seidel. MS-BMC-GS is a modified version of the BMC-GS, and it further improves the convergence rate and data locality. Although the proposed approach increases the amount of computation, the effect of this negative factor is minimized by an improved data locality.

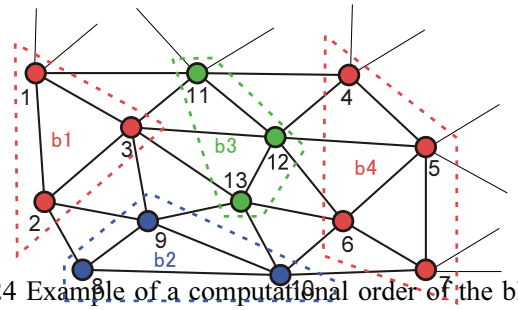


Fig.24 Example of a computational order of the block multi-color Gauss-Seidel smoother

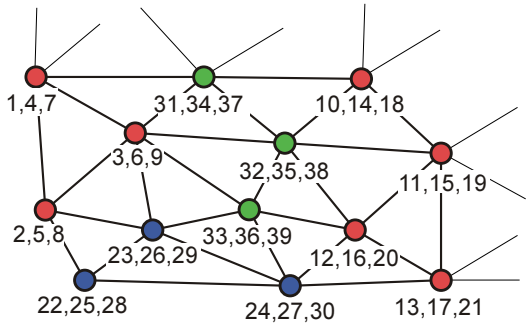


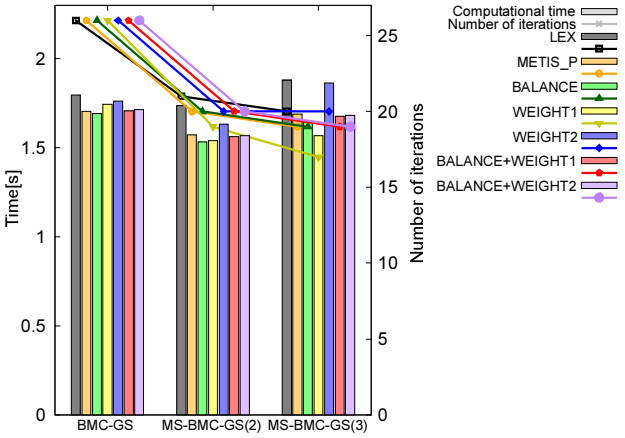
Fig.25 Example of a computational order of the MS-BMC-GS(3) smoother

In detail, with MS-BMC-GS(α), we apply the α times Gauss-Seidel (GS) iterations continuously to each block that was applied a block multi-color ordering (Fig. 25). This approach improves the convergence rate efficiently. Also, thanks to the size of the blocks as smaller than the cache size, the continuous Gauss-Seidel iterations are enough faster. In this study, we also

discuss the impact of the strategy for constructing blocks. As we mentioned above, we apply the multiple Gauss-Seidel iterations. Therefore, load-balancing among the blocks and relationships among elements in each block have strong impacts on the convergence rate and the effect of parallelization. Then, we create a weighted graph from the coefficient matrices that represents fine grid or coarse grid, and using a METIS library for partitioning.

Fig.26 shows the effects of MS-BMC-GS and the methodology for constructing blocks. For the numerical evaluations, we used one node of a local system, which has Xeon silver 4110. Then, we solved “G3 circuit” and “Parabolic FEM” that are published on the Florida sparse matrix collection. In this figure, lines and bars show the number of iterations and computational time, simultaneously. A "LEX" shows the result of the partitioning with lexicographic order. "METIS_P" and "BALANCE" denote the result of the graph partitioning using the METIS library with the non-weighted or the vertex-weighted graph. Also, "WEIGH1" and "WEIGHT2" shows the graph partitioning with the edge-weighted graph. The difference between them is how to calculate the weight of the edge. On the "WEIGH1", we calculated the weight of the edge from the non-zero elements of the coefficient matrix that present a fine grid. On the "WEIGHT2", we calculated the weight of the edge from a coarse grid. As shown in Fig.24, we confirm that the MS-BMC-GS shows better performance than the BMC-GS. We also find that the methodology for constructing blocks has a strong impact on the performance of MS-BMC-GS. In the best case, MS-BMC-GS shows 18% faster than the BMC-GS.

(a) Result with the G3 circuit



(b) Result with the Parabolic FEM

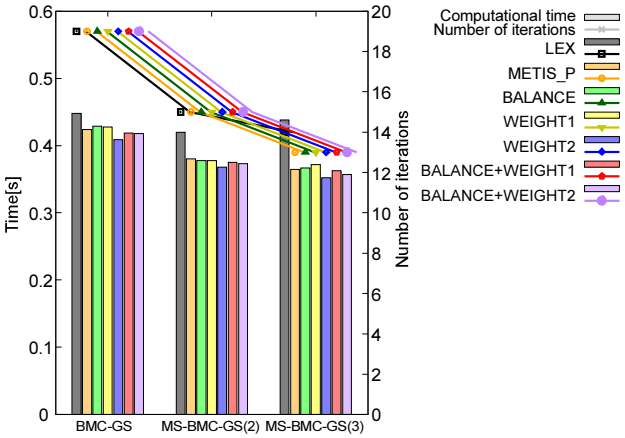


Fig. 26 Computational time and number of iterations of AMG with the MS-BMC-GS smoother. (Bars show the computational time and lines show the number of iterations.)

6. Progress of FY 2019 and Future Prospects

This is a 2-year project in FY.2018 and FY. 2019, where we do fundamental research and development, code optimization and preliminary evaluation of performance and robustness in FY.2018, and performance evaluation using real-world large-scale applications in FY.2019.

This project was accomplished almost as originally planned in the following areas:

- Geometric Multigrid (GMG) for 3D Goundwater Flow
- Algebraic Multigrid (AMG, SA-AMG) for 3D Solid Mechanics
- PinST-TSC

- PinST-Explicit
- Parallel Algorithms (MS-BMC-GS, Parallel Reordering/Coloring, Parallel Aggregation)

Although some of the works, such as implementation of SELL-C-σ, were not done yet, the works will continue in the future.

We published 2 journal papers [1,2], and 4 conference papers with peer review [3,4,5,6]. Moreover, 1 journal paper [3] is under 2nd round review. Trip to SC19 in Denver, CO for presentation of [4] was supported by JHPCN.

Because this is an international project, international collaboration is also an important part of the project. Researchers from Japan, Germany and USA have been working closely on certain topics, such as SELL-C-σ, MS-BMC-GS and Nera Kernel Vectors. We have been holdings meetings for discussions during international conferences, such as ISC-HPC 2019 (Frankfurt, Germany, 2019.6), SPPEXA Final Workshop (Dresden, Germany, 2019.10), 3rd French-Japanese-German Workshop on Programming and Computing for Exascale and Beyond (Tokyo, Japan, 2019.11), SC19 (Denver, CO, USA, 2019.11), SIAM PP20 (Seattle, WA, 2020.2). Members of ITC/U.Tokyo also visited Lawrence Berkeley National Laboratory in September, 2019, and discussed on further research collaboration including this topic.

Ryo Yoda (Kogakuin University) visited Prof. Matthias Bolten (BUW) at Wuppertal, Germany for two weeks in February 2020. This trip was supported by JHPCN. They worked together for research and development on PinST, and for revising the paper [3].

Although the project has completed successfully, we also submitted a proposal for a new project from FY.2020 to FY.2022, “Innovative Multigrd Methods II (Leading-PI:: Akihiro Fujii (Kogakuin University), Co-PI’s: Matthias Bolten (BUW) , Kengo Nakajima (The University of Tokyo). The proposal was accepted for the JHPCN project in FY.2020.

The new project basically takes over activities and results in the first phase, and more focuses on PinST.

We aim to extend the results of our research project as a library that will be available from JHPCN supercomputers. Our target computers are Post-K (Fugaku) supercomputer and BDEC system at the University of Tokyo (operation starts at May 2021).

Therefore, we set the new project as 3-year one, where we do fundamental research & development in the first year, apply our methods to real world applications in second year, and will publish the codes and the results for other researchers to apply our methods to their applications in the third year.

- Efficiency and Robustness for GMG and AMG
  - 1st year: Each item in (a)-(d) in the research purpose will be studied and developed by the members.
  - 2nd year: we will integrate the items (a)-(d) and apply them to various applications
  - 3rd year: Organize and publish the codes as library.
- PinST:
  - 1st year: (a) We will continue studying and enhancing our methods. (b) At the same time, we will try to apply our approaches to pHeat-3D. We will survey other explicit time marching applications for increasing case studies.
  - 2nd year: we will increase the test cases.
  - 3rd year: Organize and publish the codes as library.

7. List of Publications and Presentations

(1) Journal Papers (Refereed)

[1] 野村直也，中島研吾，藤井昭宏，高スケーラブル・安定的なSA-AMG法に向けたニアカーネルベクトル自動抽出手法に関する研究，情報処理学会論文誌コンピューティングシステム（ACS）12-3，46-63, 2019 （in Japanese）

[2] Fujii, A., Kaneko, S., Tanaka, T., Iwashita, T., Time Segment Correction Method for Parallel Time Integration, Journal of Information Processing, 2019, Volume 27, Pages 822-830, Released December 15, 2019, Online ISSN 1882-6652

[3] Yoda, R., Fujii, A., Tanaka, T, Bolten, M., Nakajima, K., Multigrid Reduction in Time Preconditioned Krylov Solvers for Linear Time Integration Problems, Numerical Linear Algebra with Applications (under 2nd Round Review)

(2) Proceedings of International Conferences (Refereed)

[4] Nakajima, K., Gerofi, B., Ishikawa, Y., Horikoshi, M., Parallel Multigrid Methods on Manycore Clusters with IHK/McKernel, IEEE Proceedings of 10th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems in conjunction with SC19 (The International Conference for High Performance Computing, Networking, Storage, and Analysis), 2019

[5] Kawai, M., Ida, A., Matsuba, H., Nakajima, K., Bolten, M., Multiplicative Schwartz-Type Block Multi-Color Gauss-Seidel Smoother for Algebraic Multigrid Methods, ACM Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region, (HPC Asia 2020), 2020

[6] Sakamoto, R., Kondo, M., Fujita, K., Ichimura, T., Nakajima, K., The Effectiveness of Low-Precision Floating Arithmetic on Numerical Codes: A Case Study on Power Consumption, ACM Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region, (HPC Asia 2020), 2020

[7] Nakajima, K., Parallel Multigrid Method on Multicore/Manycore Clusters, IXPUG (Intel Extreme Performance Users Group) HPC Asia 2020, 2020

(3) International Conferences Papers (Non-refereed)

[8] Nakajima, K., Parallel Multigrid with Adaptive Multilevel hCGA on Manycore Clusters, Exascale Computing Technology for Future CFD, 2019 KSIAM Spring Conference, Seoul, Korea, 2019

[9] Nakajima, K., Parallel Multigrid with Adaptive Multilevel hCGA on Manycore Clusters, Extreme-Scale/Exascale Applications China, Japan, World, ISC High Performance 2019, Frankfurt, Germany, 2019 (**Invited Talk**)

[10] Nakajima, K., Parallel Multigrid with Adaptive Multilevel hCGA on Manycore Clusters, MS12: Recent achievements on numerical algorithms and performance optimization for large-scale scientific and engineering computing, IMG 2019: International Multigrid Conference, Kunming, China, 2019

[11] Nakajima, K., Innovative Methods for Scientific Computing in the Exascale Era by Integrations of (Simulation+Data+ Learning) (S+D+L): Supercomputing in “Society 5.0”, Jornada Universritaria de Supercomputo 2019 (El Supercmoputo en la Tranformacion Digital) (Guadalajara, Mexico) (**Keynote Talk**)

[12] Nakajima, K., An Innovative Method for Integration of Simulation /Data/Learning in the Exascale/Post-Moore Era, APCOM 2019: Asian Pacific Congress on Computational Mechanics (December 18-21, 2020, Taipei, Taiwan) (**Semi-Plenary Talk**)

[13] Nakajima, K., Parallel Multigrid Methods with Adaptive Multilevel hCGA on Manycore Clusters, MS1603: Parallel Programming Models, Algorithms and Frameworks for Extreme Computing & Big Data, APCOM 2019: Asian Pacific Congress on Computational Mechanics (December 18-21, 2020, Taipei, Taiwan)

[14] Iwashita, T, Nakajima, K., Shimokawabe, T., Nagao, H., Ogita, T., Katagiri, T., Yashiro, H., Matsuba, H., h3-Open-BDEC: Innovative Software Platform for Scientific Computing in the Exascale Era by Integrations of (Simulation + Data + Learning) , HPC Asia 2020 (poster) (Fukuoka, January 2020)

[15] Nakajima, K., Innovative Methods for Scientific Computing in the Exascale Era by Integrations of (Simulation+Data+ Learning), SIAM Conference on Parallel Processing for Scientific Computing (PP20), MS25/36: Progress and Challenges in Extreme Scale Computing and Big Data (Seattle, WA, USA, February 12-15, 2020)

[16] Bolten, M., Multigrid for Shifted Systems Appearing in Parallel-in-Time Integration, SIAM Conference on Parallel Processing for Scientific Computing (PP20), MS47/58: Parallel-in-Time Integration Methods (Seattle, WA, USA, February 12-15, 2020)

(4) Presentations at Domestic Conferences (Non-refereed)

[17] 中島研吾，高性能・変動精度・高信頼性数値解析手法とその応用，第24回日本計算工学会講演会（大宮，2019年5月30日）（in Japanese）

[18] 中島研吾，堀越将司，Balazs Gerofi，石川裕，AM-hCGA法による並列多重格子法，情報処理学会研究報告（2019-HPC-170-19），北見，北海道，2019（in Japanese）

[19] 野村直也，中島研吾，藤井昭宏，SA-AMG法における収束性安定化のための効率的なニアカーネルベクトル抽出手法に向けた研究，情報処理学会研究報告（2019-HPC-170-20），北見，北海道，2019（in Japanese）

[20] Chen, Y.C., Nakajima, K., Parallel-in-Space/Time Method for Explicit Time-Marching Scheme, IPSJ SIG Technical Report, 2019-HPC-170-37, Kitami, Hokkaido, Japan, 2019

[21] 中島研吾，メニィコアクラスタ向け並列多重格子法，日本応用数理学会 2019 年度年会，東京，駒場，2019

[22] 野村直也，中島研吾，藤井昭宏，ロバストな SA-AMG 法に向けたニアカーネルベクトル抽出手法に関する研究，情報処理学会研究報告（2020-HPC-173-2），札幌，北海道，2020（in Japanese）（発表は中止）

[23] 中島研吾，坂本龍一，星野哲也，有間英志，塙敏博，近藤正章，低精度演算とアプリケーション性能，情報処理学会研究報告（2020-HPC-174-5）（第 174 回 HPC 研究会）（オンライン，2020 年 5 月 13 日）（in Japanese）（in press）

(5) Others (Patents, Press Release, Books etc.)

該当無し