

ppOpen-HPC: Open Source Infrastructure for Development and Execution of Large-Scale Scientific Applications on Post-Peta-Scale Supercomputers with Automatic Tuning (AT)

Kengo Nakajima, Masaki Satoh, Takashi Furumura,
Hiroshi Okuda, Takeshi Iwashita, Hide Sakaguchi, Takahiro Katagiri,
Masaharu Matsumoto, Satoshi Ohshima, Hideyuki Jitsumoto,
Takashi Arakawa, Futoshi Mori, Takeshi Kitayama, Akihiro Ida
and Miki Y. Matsuo

Abstract ppOpen-HPC is an open source infrastructure for development and execution of large-scale scientific applications on post-peta-scale (pp) supercomputers with automatic tuning (AT). ppOpen-HPC focuses on parallel computers based on many-core architectures and consists of various types of libraries covering general procedures for scientific computations. The source code, developed on a PC with a single processor, is linked with these libraries, and the parallel code generated is optimized for post-peta-scale systems. In this article, recent achievements and progress of the ppOpen-HPC project are summarized.

Keywords ppOpen-HPC · Post-peta-scale systems · Automatic tuning · Parallel computing

K. Nakajima (✉) · M. Satoh · T. Furumura · H. Okuda · T. Katagiri · M. Matsumoto ·
S. Ohshima · F. Mori · T. Kitayama
The University of Tokyo, Tokyo, Japan
e-mail: nakajima@cc.u-tokyo.ac.jp

M. Satoh
e-mail: satoh@aori.u-tokyo.ac.jp

T. Furumura
e-mail: furumura@eri.u-tokyo.ac.jp

H. Okuda
e-mail: okuda@k.u-tokyo.ac.jp

T. Katagiri
e-mail: katagiri@cc.u-tokyo.ac.jp

M. Matsumoto
e-mail: matsumoto@cc.u-tokyo.ac.jp

S. Ohshima
e-mail: ohshima@cc.u-tokyo.ac.jp

1 Overview of ppOpen-HPC

Today, high-end parallel computer systems are becoming larger and more complex. It is very difficult for scientists and engineers to develop efficient application codes that make use of the potential performance of these systems.

We propose an open source infrastructure for development and execution of optimized and reliable simulation codes on large-scale parallel computers. This infrastructure is named *ppOpen-HPC* [1, 2], where “pp” stands for “post-peta-scale”, as shown in Fig. 1. The target post-peta-scale system is the Post T2K System, which will be installed and operated by the Joint Center for Advanced High Performance Computing (JCAHPC) [3] under collaboration between the University of Tsukuba and the University of Tokyo. The Post T2K System, which will be installed in FY 2016, is based on many-core architectures, such as the Intel MIC/Xeon Phi. Its peak performance is expected to be more than 30 PFLOPS.

ppOpen-HPC is a five-year project (FY 2011–2015) and a part of the “Development of System Software Technologies for Post-Peta-Scale High Performance Computing” funded by JST/CREST (Japan Science and Technology Agency, Core Research for Evolutional Science and Technology) [4]. ppOpen-HPC is being developed by the University of Tokyo (Information Technology Center, Atmosphere and Ocean Research Institute, Earthquake Research Institute, Graduate School of Frontier Sciences), Kyoto University, Hokkaido University, and Japan Agency for Marine-Earth Science and Technology (JAMSTEC). The expertise of the members covers

F. Mori

e-mail: f-mori@eri.u-tokyo.ac.jp

T. Kitayama

e-mail: kitayama@h.k.u-tokyo.ac.jp

T. Iwashita

Hokkaido University, Hokkaido, Japan

e-mail: iwashita@iic.hokudai.ac.jp

H. Sakaguchi · M.Y. Matsuo

JAMSTEC, Kanagawa, Japan

e-mail: sakaguchih@jamstec.go.jp

M.Y. Matsuo

e-mail: mikiy@jamstec.go.jp

H. Jitumoto

Tokyo Institute of Technology, Tokyo, Japan

e-mail: jitumoto@gsic.titech.ac.jp

T. Arakawa

RIST, Tokyo, Japan

e-mail: arakawa@rist.jp

A. Ida

Kyoto University, Kyoto, Japan

e-mail: ida@media.kyoto-u.ac.jp

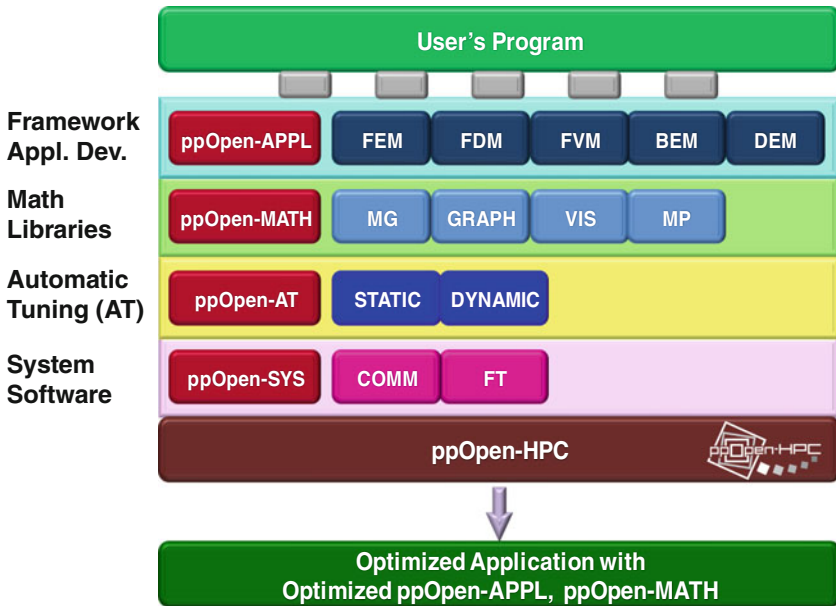


Fig. 1 Overview of ppOpen-HPC

a wide range of disciplines related to scientific computing, such as system software, numerical libraries/algorithms, computational mechanics, and earth sciences. ppOpen-HPC includes the following four components (Fig. 1):

- ppOpen-APPL
- ppOpen-MATH
- ppOpen-AT
- ppOpen-SYS

Libraries in ppOpen-APPL, ppOpen-MATH, and ppOpen-SYS are called from user programs written in Fortran and C/C++ with MPI and OpenMP.

In ppOpen-HPC, we are focusing on five types of discretization methods for scientific computing: FEM, FDM, FVM, BEM, and DEM (Fig. 2). ppOpen-APPL is a set of optimized libraries covering various types of procedures for these five methods. Source code developed on a PC with a single processor is linked with ppOpen-APPL, and the parallel code generated will be optimized for a post-peta-scale systems. Key issue for this type of framework like ppOpen-APPL is well-designed data structure for scientific computing. In previous projects, such as GeoFEM [5] and HEC-MW [6], some of the authors developed such frameworks, where typical procedures for FEM have been optimized for certain types of supercomputers. This type of framework provides dramatic efficiency, portability, and reliability in the development and execution of scientific applications. It reduces both the number of steps in the source code and the duration of time required for parallelization and

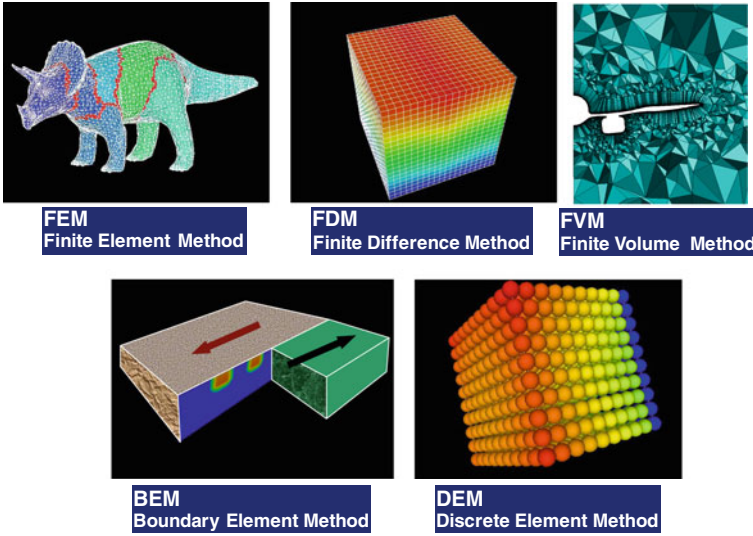


Fig. 2 Target applications of ppOpen-HPC

optimization of legacy code. In ppOpen-HPC, we extend this idea to other four types of methods, and introduce a new feature, *automatic tuning (AT)*. AT enables a smooth and easy shift to further development on future architectures through the use of ppOpen-AT, which generates optimized libraries and applications under various types of environments automatically.

ppOpen-MATH is a set of libraries for multigrid, visualization, loose coupling, etc., while ppOpen-SYS includes system software libraries related to node-to-node communication and fault tolerance. ppOpen-HPC enables more than 2,000 users of the supercomputer system in the University of Tokyo to switch from homogeneous multicore clusters to a post-peta-scale system based on many-core architectures. Although the final target of ppOpen-HPC is the Post T2K system, libraries for multicore clusters, such as K computer, and Fujitsu PRIMEHPC FX10, are also developed.

In the following sections, we describe recent achievements in the development of each component of ppOpen-HPC, as shown in Fig. 1.

2 ppOpen-APPL

ppOpen-APPL is a set of libraries that covers various types of procedures for scientific computations, such as parallel I/O of datasets, matrix formation, linear solvers with practical and scalable preconditioners, visualization, adaptive mesh refinement (AMR), and dynamic load balancing, in various types of models, including FEM, FDM, FVM, BEM, and DEM, shown in Fig. 2.

Each component is based on existing practical application codes. ppOpen-APPL provides common data structures and interfaces that support users with the easy implementation procedures of ppOpen-HPC onto legacy codes.

2.1 Simulation of 3D Seismic Wave Propagation Using ppOpen-APPL/FDM

ppOpen-APPL/FDM is a framework for the development of applications by the finite difference method (FDM), as shown in Fig. 1.

In the recent trend of supercomputer architectures, the *byte-per-flops* ratio (B/F ratio, ratio of memory bandwidth (BYTE/sec, B) and computational performance (FLOPS, F)) has been dropping drastically. For example, it dropped from 4 to 0.5 for the Earth Simulator and the K computer. The B/F ratio of next-generation computers is expected to fall even further. Therefore, an important issue for the future high-performance parallel computing of FDM simulations is the restriction of the memory bandwidth relative to the CPU speed. To overcome this problem, it is necessary to develop a new FDM simulation structure suitable for future many-core and low B/F machines.

We proposed to effectively decrease the required B/F ratio of the FDM simulation of seismic wave propagation. To validate our proposal, we evaluated the performance of a parallel 3D FDM simulation of seismic wave propagation on the Intel Xeon Phi coprocessor [7]. The original FDM simulation code first loaded velocity and stress components from memory to processor, calculated each spatial derivative, and stored them in memory. This requires large B/F ratios to load and store the large number of variables. These derivatives were then used for the next kernels of update stress and update velocity, which also require large B/F ratios of 2.7 and 1.7 for each kernel, respectively. We modified the B/F reduction FDM code to merge the derivative and update calculations, and thereby avoided the need to store and load variables during the calculations. As a result, the required B/F ratios for both kernels dropped dramatically to 0.4.

Figure 3 compares the performance of the original and the modified code in MPI/OpenMP hybrid parallel computing on the Intel Xeon Phi processor. For parallel computing up to 60 physical cores, the modified code is slower than the original code. However, with much larger thread parallel simulation using 240 logical cores, it is confirmed that the speed of the modified code is double the speed of the original code. This B/F reduction code is also suitable for other parallel processors, such as Fujitsu PRIMEHPC FX-10 at the University of Tokyo [8].

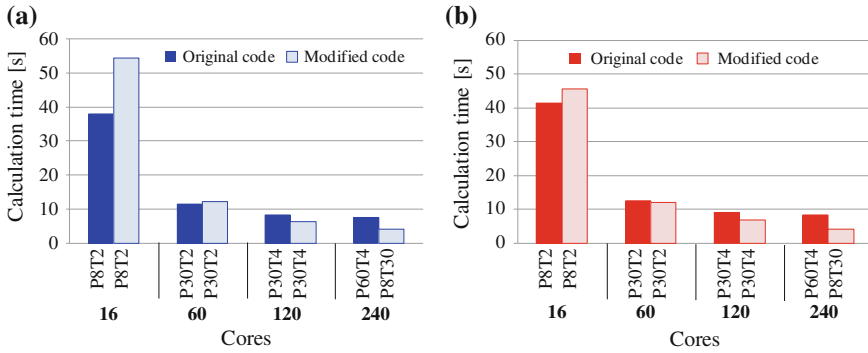


Fig. 3 Comparison of MPI/OpenMP hybrid parallel computing based on the original code and on the modified B/F reduction code for the **a** update velocity and **b** update stress kernels. Indexes P and T indicate number of MPI processes and OpenMP threads, respectively (e.g., P30T4 means hybrid parallel computing using 30 MPI processes and each has 4 OpenMP threads)

2.2 ppOpen-APPL/AMR-FDM with Adaptive Mesh Refinement

We developed an adaptive mesh refinement (AMR) framework for explicit FDM schemes in the ppOpen-APPL/AMR-FDM library [9]. To overcome the problem of load imbalance in parallelized AMR simulations, we implemented a dynamic domain decomposition (DDD) technique, with which the whole computational domain is dynamically re-decomposed into new subdomains so that the computational load on each process becomes nearly the same. Test simulations of a linear advection equation using the AMR framework are shown in Fig. 4. The fine grids are adaptively created where the gradient of the waveform is high.

Figure 5 shows the temporal evolutions of execution time for the simulation of 512 MPI processes in the cases with and without DDD. In the case without DDD, the profile of the execution time has large fluctuations because the waveform propagates across boundaries of the subdomains. On the other hand, in the case with DDD, the average execution time can be reduced. The DDD procedure succeeds in significantly holding down the average execution time. The graph on the right in Fig. 5 shows the execution time focused on a certain iteration count. The wave profile of the zigzag line is attributed to the timing of DDD, which takes time to perform. The DDD performs when the computational costs exceed a load-balance criterion. The time difference between DDD ON and OFF constitutes the overhead of the DDD procedure.

In addition to the above, we also developed the AMR framework for implicit time-marching schemes in which AMR and the multigrid method are used concurrently. In this implementation, each grid layer created by the AMR method corresponds to each layer of the V-cycle used by the multigrid method. We aim for not only explicit but also implicit schemes that can use the framework. Although code optimizations

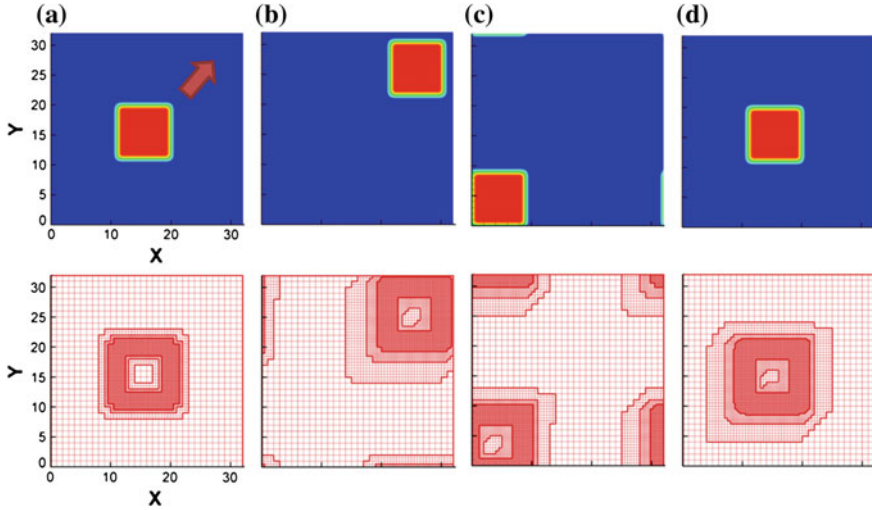


Fig. 4 Example of AMR

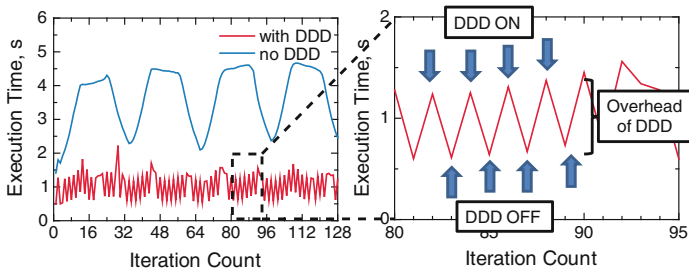


Fig. 5 Effect of DDD procedures

are needed, such an AMR framework shows the value of development for multi-scale simulations on post peta-scale systems.

2.3 *ppOpen-APPL/BEM and HACApK*

The ppOpen-APPL/BEM is a software tool for large-scale parallel boundary element method (BEM) analyses. This tool consists of the BEM-BB (Back-Bone) framework, templates, and the HACApK library. All the components are parallelized based on the hybrid MPI+OpenMP programming model. The BEM-BB framework provides users with parallelized program code for coefficient matrix generation and linear system solvers. Users can easily develop a parallel BEM code for their own applications by adding a user function describing the integral operation onto the framework. For some

Fig. 6 Example of static electric field analysis: The electrical charge on surfaces of the humanoids is calculated

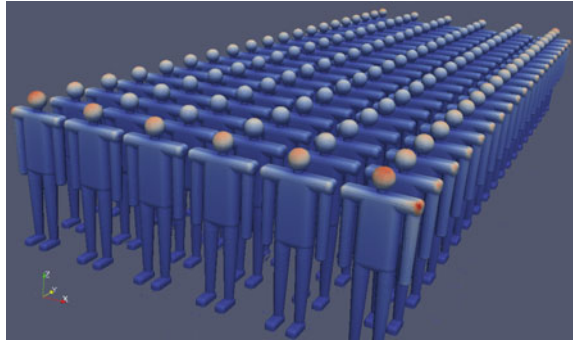
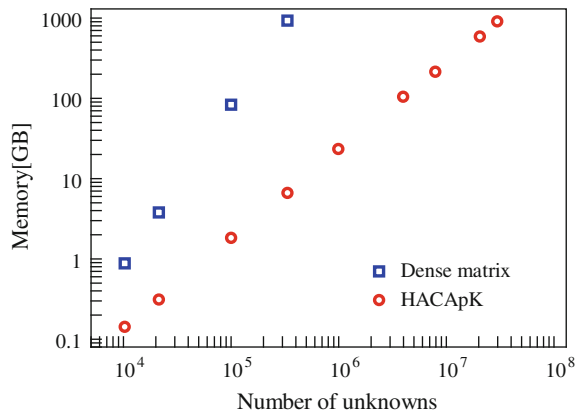


Fig. 7 Memory usage of H-matrices and original dense matrices as a function of the number of unknowns N



specific application domains, template programs for integral operation are provided. By using the BEM-BB framework with a template, the user obtains complete BEM simulation code. Currently, a template for static electric field analysis is available (Fig. 6). Moreover, we have been developing the HACApK library to realize faster and larger-scale BEM analyses.

The HACApK library adopts hierarchical matrices (H-matrices) with adaptive cross approximation (ACA) as the approximation technique for dense matrices occurring from the integral equation method represented by BEM. The method of H-matrices with ACA is based on the idea that submatrices corresponding to remote interactions become numerically low-rank matrices. H-matrices with ACA reduce the complexity from $O(N^2)$ to $O(N \log N)$, where N denotes the number of unknowns (Fig. 7).

For parallelization of H-matrices on symmetric multiprocessor (SMP) cluster systems, we proposed a set of algorithms for constructing H-matrices and performing multiplication of an H-matrix and a vector [10]. The proposed algorithms are implemented by the flat-MPI and hybrid MPI+OpenMP programming models. The performance of these implementations is evaluated by electric field analysis (Fig. 8). In the flat-MPI version, the speedup is limited in hierarchical matrix-vector multiplication.

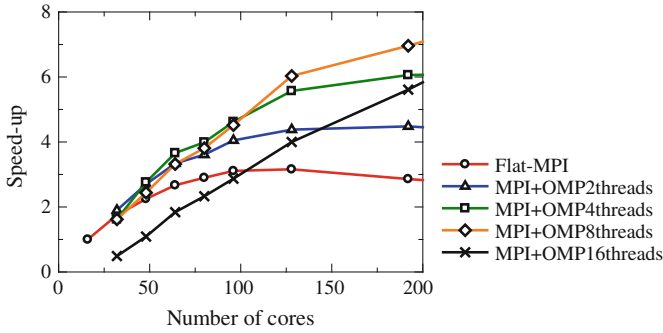


Fig. 8 Parallel scalability when performing a multiplication of an H-matrix and a vector on Fujitsu PRIMEHPC FX10 at the University of Tokyo [8]

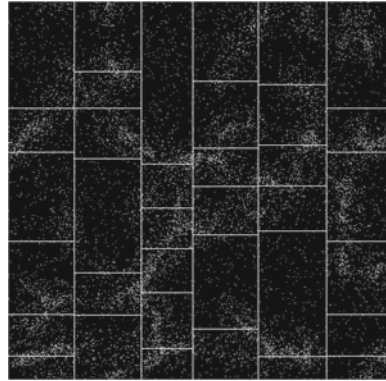
We succeeded in developing a hybrid MPI+OpenMP version to improve the parallel scalability. In numerical experiments, the hybrid version exhibits a better parallel speed-up for the hierarchical matrix-vector multiplication up to 256 cores of Fujitsu PRIMEHPC FX10 at the University of Tokyo [8]. In addition to the above parallel algorithms, we also proposed an improved method for H-matrices with ACA [11]. By using the proposed method, we can avoid the problem that ranks of approximated matrices increase rapidly as the matrix size increases when conventional H-matrices with ACA are employed for an integral equation whose kernel function has high-order singularities. In particular, application of the proposed method enables us to perform large-scale simulations, in which conventional H-matrices with ACA fail to construct appropriate low-rank approximations.

2.4 *ppOpen-APPL/DEM: Open Library for Discrete Element Method*

Particle methods are among the more commonly used approaches for numerical simulations of physical problems. Particle methods, e.g., the discrete element method (DEM), are applied in various fields such as molecular bioscience, material science, civil engineering, oceanography, and astrophysics. However, millions or billions of particles are necessary to simulate physical problems with sufficient accuracy, and this level of refinement is indeed important in applications of industrial problems. Thus, further evolution of both the infrastructure and the performance development of particle simulation is needed. ppOpen-APPL/DEM is a library designed to enable the easy implementation of particle simulation code with the interactions of short-range particles such as DEM for parallel computers.

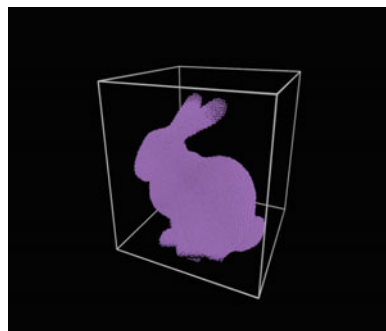
The ppOpen-APPL/DEM main library provides fundamental subroutines and functions that organize the particle simulation coded with OpenMP/MPI hybrid programming [12]. The code using the library performs parallel computation on a PC

Fig. 9 Particle simulation by ppOpen-APPL/DEM: The dynamic load balance is achieved with the slice-grid method



cluster and a supercomputer. Although simulating the motions of interacting particles is easy, moving particles make it difficult to balance the computational load. To realize the dynamic load balance over the computational nodes, we implemented the DDD technique that decomposes the computational domain into several subdomains, each of which is associated with a distinct computational node (Fig. 9), which shows our implementation of the slice-grid method. Other protocols, such as orthogonal recursive bisection, will be implemented in the future. Furthermore, to reduce the difficulty of programming complex initial and boundary conditions, ppOpen-APPL/DEM provides a utility library named ppOpen-APPL/DEM-Util. This utility includes subroutines that combine a stereolithography (STL) data file into particle simulations. For example, input STL data are used to design the initial coordination of particles; the initial coordination enables the user to easily prepare a complex configuration of particle positions (Fig. 10). As another example, the input STL data are used as a boundary condition of the particle simulation. Those implementations are useful for the design of large-scale simulations.

Fig. 10 Particle simulation using ppOpen-APPL/DEM-Util: The utility enables us to use stereolithography (STL) data as the initial condition



3 ppOpen-MATH

ppOpen-MATH consists of common numerical libraries, such as multigrid solvers (ppOpen-MATH/MG) (Fig. 4), parallel graph libraries (ppOpen-MATH/GRAPH), parallel visualization (ppOpen-MATH/VIS), and a library for coupled multi-physics simulations (ppOpen-MATH/MP).

3.1 *ppOpen-MATH/MP and NICAM-COCO Coupling*

ppOpen-MATH/MP is a coupling software applicable to the models employing various discretization methods such as FDM, finite volume method (FVM) and finite element method (FEM) [13]. To demonstrate the applicability of ppOpen-MATH/MP, we used it for an atmospheric model and ocean model coupling. The atmospheric model selected for this purpose is the Nonhydrostatic ICosahedral Atmospheric Model (NICAM), which is a nonhydrostatic global model employing an icosahedral grid system and an FVM discretization method [14]. The CCSR Ocean Component Model (COCO) is used as an ocean model coupled with NICAM. COCO adopts a tri-polar grid, in which the northern polar region grid points do not follow a latitude-longitude grid, and the discretization method is FDM [15]. For realizing wide applicability, ppOpen-MATH/MP is designed so that users can implement their own interpolation code. The interpolation code is based on the first-order conservative remapping scheme in [16]. Physical quantities exchanged from NICAM to COCO are 13 variables, including wind speed, heat flux, and precipitation. The quantities exchanged from COCO to NICAM are 6 variables, including SST and sea ice thickness.

In addition to NICAM-COCO coupling, we implemented NICAM and IO component coupling. The reason for this coupling is that the icosahedral grid employed by NICAM is not suitable for analyzing the results. For example, the calculation of zonal mean values is not straightforward and the visualization tools assume a latitude-longitude grid (lat-lon grid) in many cases. So, we developed an IO program that converts the icosahedral grid to the lat-lon grid and is executed in parallel with NICAM. The implemented conversion schemes are a bilinear interpolation, a control volume weighted average, and the nearest-neighbor method.

Figure 11 is a schematic of the coupling system described above. The coupling system is designed so that NICAM automatically detects the coupling pattern at runtime without any other configuration. For example, when COCO is executed in parallel with NICAM, subroutines for NICAM-COCO coupling are used, and if not, subroutines of the mixed layer ocean model are called. IO is also the same as the case of COCO, in which NICAM automatically sends output data to IO only when the IO component is executed.

For utilizing ppOpen-MATH/MP, a correspondence table of a grid point index between the models is required as input data. Therefore, we developed a calculation

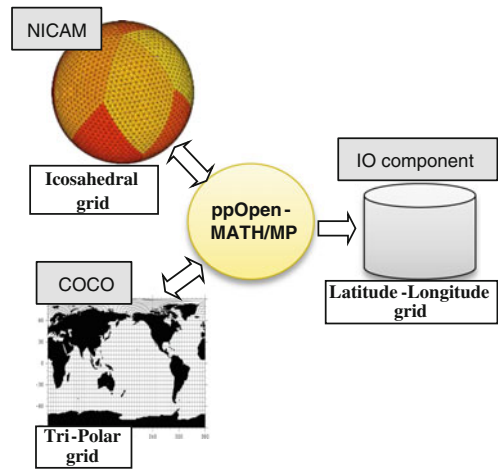


Fig. 11 Schematic of the coupling system

tool named ppOpen-MATH/MP-PP, which targets two-dimensional meshes on a sphere surface. For calculating the correspondence, a new search algorithm, for which efficiency $O(n)$ was developed, requires $O(n^2)$ calculation through the brute-force method. Figure 12 shows the result of the performance measurement of the new algorithm. Here, two mesh types, the NICAM icosahedral grid and the I/O lat-lon grid, were selected as a test case. The numbers of grid points are listed in Table 1. As shown in the figure, execution time versus the number of grid points increases at a rate of $O(n)$, and the effectiveness of the new algorithm is confirmed.

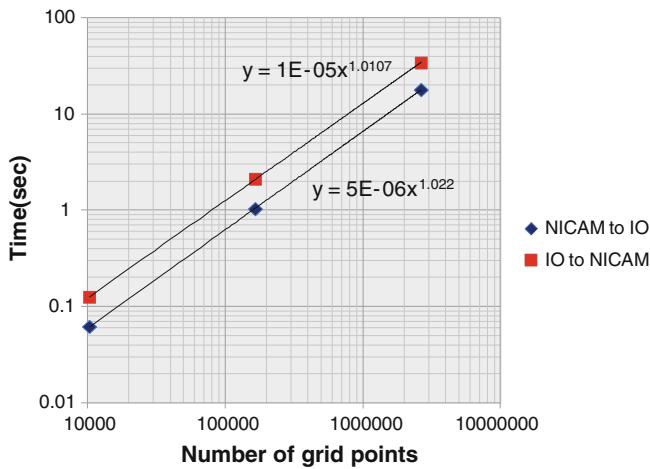


Fig. 12 Search time on NICAM grid versus lat-lon grid

Table 1 The number of grid points

The number of Lat-Lon grid	10,585	166,753	2,657,665
The number of NICAM grid	20,480	327,680	5,242,880

3.2 Integrated Earthquake Simulations Using ppOpen-MATH/MP

Simulations including multi-scale or varied physical phenomena are difficult because such kinds of modeling and implementation by a specific application are difficult. For example, to simulate the earthquake shock coming from earthquake sources and building damage, both a seismic wave that propagates over a wide region several hundred kilometers square and a shaking building that occurs in a small region of several tens of meters square must be resolved concurrently. In this case, if the FDM application using ppOpen-APPL/FDM is suitable for the analysis of elastic (seismic) wave propagations and the FEM application using ppOpen-APPL/FEM is suitable for the analysis of dynamic solid mechanics (the building) can be used in combination, the multi-scale and multi-physics coupling simulation is easy to use. Our final goal is to develop the application coupler (ppOpen-MATH/MP) [11] by using ppOpen-APPL libraries in various combinations. Figure 13 shows an example of a multi-scale and multi-physics coupling simulation by using ppOpen-MATH/MP. In this case, the seismic wave calculated by the FDM model is transferred to the FEM model and is interpolated in the FEM model mesh through ppOpen-MATH/MP. The number of nodes and processes in the FDM model and FEM model, respectively, are different, but they are automatically interpolated and are arranged by the coupler. Application

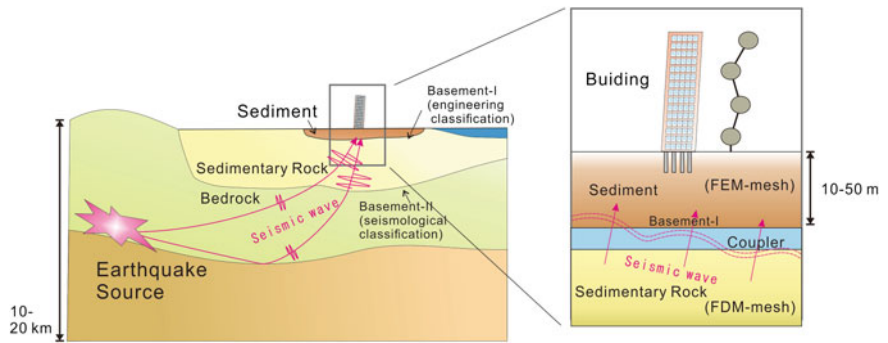


Fig. 13 Example of a multi-scale and multi-physics coupling simulation: seismic wave propagation (ppOpen-APPL/FDM), dynamic solid mechanics (ppOpen-APPL/FEM), and application coupler (ppOpen-MATH/MP)

developers need not consider the structure of the FDM and FEM applications, and they can develop a coupling simulation by using ppOpen-MATH/MP.

A practical simulation is executed on the large-scale computational resources of Fujitsu PRIMEHPC FX10 at the University of Tokyo [8]. The simulation target is the earthquake that occurred at Awaji Island on 13 April 2013. The seismic source was located on the central part of Awaji Island, Hyogo prefecture, Japan. The computational domain of Seism3D+, which is composed by the ppOpen-APPL/FDM library, is 60km square from Awaji Island and the domain of FrontISTR++, which is composed by the ppOpen-APPL/FEM library, is an actual building of the RIKEN Advanced Institute for Computational Science (AICS), Port Island, Kobe, as modeled by an unstructured mesh. This building mesh is placed on two locations of Port Island and the Kobe stadium, where ground conditions differ from each other. The total computational nodes on the FX10 used in the simulation are 4560 nodes (16 cores/1 node): 2560 nodes for Seism3D+ and 1000 nodes/1 place = total 2000 nodes for FrontISTR++. In the simulation, seismic wave propagations (Seism3D+) for the simulation time of 90s were calculated in the computational time of about 6h, and building vibrations originated from the seismic wave (FrontISTR++) for the simulation time of 20s were calculated in the computational time of about 16h. However, it was revealed that the way to allocate memory of the coupler has some problems when such a large-scale simulation is performed. This is because a part of the initialization routine in ppOpen-MATH/MP includes the centralized procedure of all MPI processes. Optimization of the code will be carried out as a future plan.

3.3 ppOpen-MATH/MG: Multigrid Solver

Optimization of both serial and parallel communications is a critical issue for the development of scalable algorithms in next-generation applications. Serial communication is the data transfer through memory hierarchies of each processor, whereas parallel communication is the message passing between computing nodes through the network by MPI. A multigrid is a scalable method for solving linear equations and preconditioning Krylov iterative linear solvers, and is especially suitable for large-scale problems. The parallel multigrid method is expected to be one of the powerful tools on post-peta/exa-scale systems. Recently, the High Performance Conjugate Gradient (HPCG) [17] was proposed as a new benchmark for evaluation of the practical performance of supercomputer systems. HPCG solves sparse matrices derived from finite element applications by using a conjugate gradient (CG) linear solver preconditioned with the multigrid method (MGCG).

The parallel multigrid method and MGCG include both serial and parallel communication processes that are generally expensive. This article summarizes recent efforts of the optimization of serial and parallel communications in parallel MGCG solvers with geometric multigrid procedures using up to 4,096 nodes (65,536 cores) of the Fujitsu PRIMEHPC FX10 at the University of Tokyo [8]. The target application, pGW3D-FVM, is a 3D finite-volume simulation code, which solves groundwater

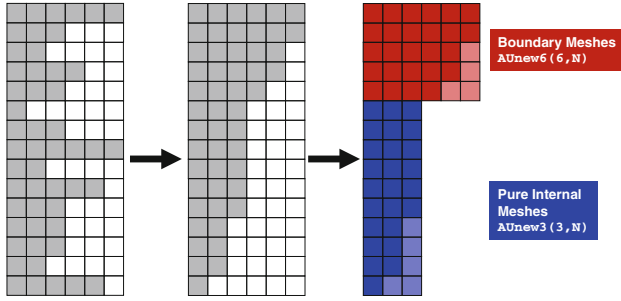


Fig. 14 Idea of sliced ELL format [19] with two slices in the present work

flow problems through heterogeneous porous media by the parallel MGCG method [18]. The performance of both flat MPI and OpenMP/MPI hybrid parallel programming model (HB $M \times N$ (M : number of threads on each MPI process, N : number of MPI processes on each node)) has been evaluated.

In the present work, a new format for sparse matrix storage based on sliced ELL [19] (Fig. 14), which has been well utilized for the optimization of sparse matrix-vector multiplication (SpMV), is proposed for optimization of serial communication on memories. In addition, hierarchical coarse grid aggregation (*hCGA*) (Fig. 15) is introduced for optimization of parallel communication by message passing. The proposed methods are implemented for pGW3D-FVM, and the robustness and performance of the code was evaluated by using up to 4,096 nodes (65,536 cores) of the Fujitsu FX10 system. The parallel MGCG solver using the *sliced* ELL format

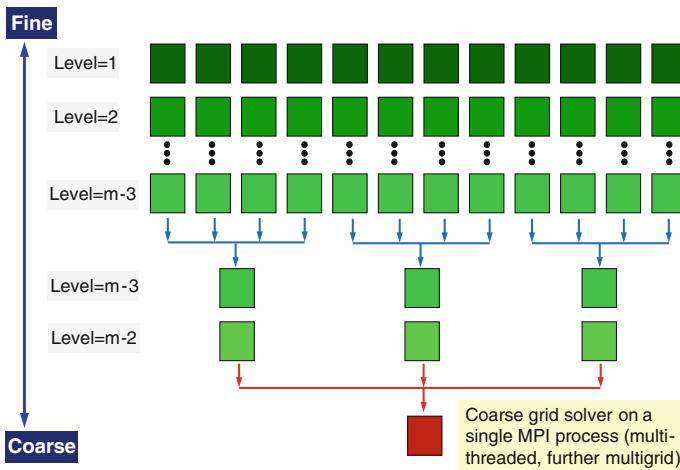


Fig. 15 Procedures of hierarchical CGA (*hCGA*), where the number of MPI processes is reduced before the final coarse grid solver of CGA on a single MPI process

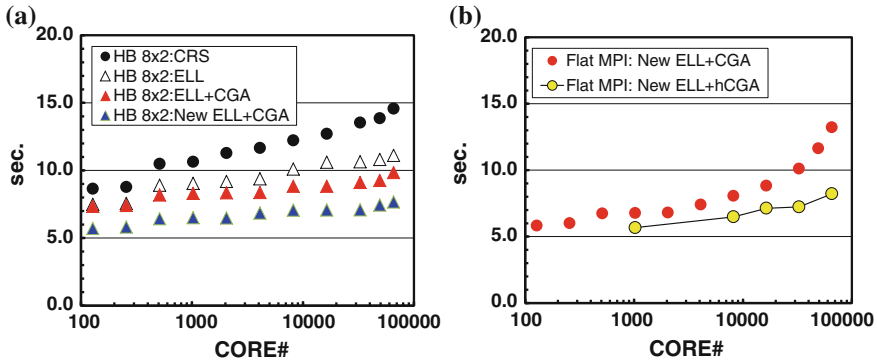


Fig. 16 Performance of MGCG solver on Fujitsu FX10 using up to 4,096 nodes (65,536 cores), weak scaling (elapsed time for MGCG): 262,144 ($=64^3$) meshes/core, max. total problem size: 17,179,869,184 meshes

provided performance improvement in both weak scaling (25–31 %) and strong scaling (9–22 %) compared to the code using the original ELL format. Moreover, *hCGA* provided excellent performance improvement in both weak scaling (1.61 times) and strong scaling (6.27 times) for the flat MPI parallel programming model. *hCGA* was also effective for improvement of parallel communications (Fig. 16a, b).

The effect of sliced ELL on serial communication was significant, while that of *hCGA* on parallel communication was not so impressive except for flat MPI cases. Because *hCGA* proved to be very effective for reducing the overhead of the coarse grid solver, it will also provide a more significant effect on hybrid parallel programming models with a larger number of nodes. The computational amount of the coarse grid solver for each core of flat MPI is 256 ($=16 \times 16$) times as large as that of HB 16×1 . Therefore, *hCGA* is expected to be really effective for HB 16×1 with more than 2.50×10^5 nodes (4.00×10^6 cores) of the Fujitsu FX10, where the peak performance is more than 60 PFLOPS.

CGA and *hCGA* include various types of parameters, and the optimum values of those were derived through empirical studies in the present work. Development of methods for automatic selection of these parameters [20] is also an interesting technical issue for future work.

4 ppOpen-AT

Computer architectures are becoming more and more complex due to non-standardized memory accesses and hierarchical caches. The AT capability is an important and critical technology for further development of new architectures, and maintenance of the overall framework to establish high productivity for performance tuning. Generally speaking, keeping high performance with a legacy code

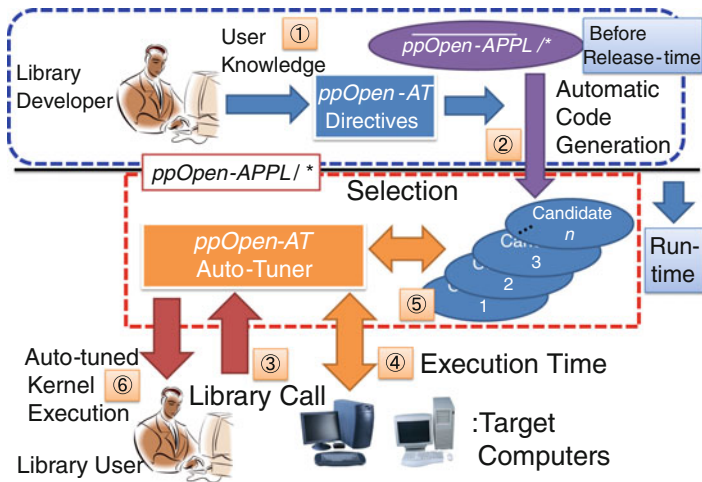


Fig. 17 Procedures for generation of optimized code by using ppOpen-AT

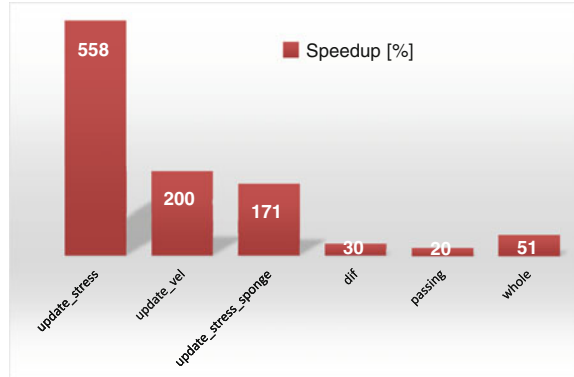
on different computer environments, described as “performance portability”, is a challenging issue. In this section, we show the current results of research on the AT function to establish performance portability.

ppOpen-AT automatically and adaptively generates optimum implementations for efficient memory accesses in the processes of methods for scientific computing in each component of ppOpen-APPL. Example processes are explicit time marching procedures, matrix assembling procedures, and implicit linear solvers. These are achieved under various environmental constraints, such as the architecture of the supercomputer system, available resources, problem size, etc. ppOpen-AT also optimizes widely used open source applications and numerical libraries, such as OpenFOAM and PETSc. With the focus on optimum memory access, directive-based special AT languages for specific procedures in scientific computing are being developed.

Figure 17 describes procedures for the generation of optimized code using ppOpen-AT.

We applied ppOpen-AT to simulation code based on FDM, which was provided as ppOpen-APPL/FDM in ppOpen-HPC. The framework utilizes well-known loop transformation techniques, such as loop fusion and loop split. In addition, we looked at the AT function to support compiler optimizations with a re-ordering of statements. Processes of data packing and unpacking for communication with MPI are targets for AT. The components of AT are carefully designed to minimize the use of software stacks to satisfy the requirements of the many-core architectures currently in operation. The FIBER framework [19] is utilized to implement an auto-tuner and timings of invocation of the AT function. In particular, execution of the AT with dedicated problem sizes and numbers of threads is crucial. The timing called Before execute-time AT on FIBER is used to optimize the target codes of ppOpen-HPC.

Fig. 18 Effect of AT by ppOpen-AT on the performance of ppOpen-APPL/FDM



The results of evaluations conducted using ppOpen-AT indicate that maximum speedup factors greater than 550 % are obtained when ppOpen-AT is applied in eight nodes of the Intel Xeon Phi [21] (Fig. 18). To show performance portability, an additional evaluation with the Intel Ivy Bridge and the Sparc64 IX-fx was performed. Different parameters were set by the AT. For execution with auto-tuned parameters, the Intel Xeon Phi is faster than the other two architectures, while the Xeon Phi is the slowest if we do not apply the AT. This shows that AT is a crucial factor to establish performance portability.

5 ppOpen-SYS/FT: Application-Level Checkpoint/Restart (CP/RS) Framework with Runtime Optimization

The application-level checkpoint/restart technique is periodically makes snapshots of application and stores them for recovery later by application programming. And it is frequently implemented within the application which has time stepping. However, optimizations of some parameters require the runtime information. In this case, the parameters described in the program source tend to depend on the application programmer's ad hoc decision.

For reducing the cost of checkpoint/restart (CP/RS) at the application level, the framework focused on a time-stepping model application and asynchronous coordination on the checkpoint with an applied optimized parameter. The scenario is as follows: (1) Processes send runtime information to the optimizing daemon. (2) The daemon optimizes the runtime with information of the processes and the environment, and returns the result to the processes. (3) The processes make a checkpoint with the newly optimized information. (4) The daemon deletes a waste checkpoint with appropriate information, such as the core-loop stepping count included on the checkpoint ppOpen-SYS/FT framework that has a reference implementation for checkpoint interval optimization.

ppOpen-SYS/FT is the directive-based application-level checkpoint/restart framework and its implementation with runtime optimization [22–24]. The implementation of ppOpen-SYS/FT adapts to large-scale systems managed with a job scheduler. This implementation optimizes the interval with the count of stepping, the execution time of stepping, and checkpointing. The optimization daemon returns the appropriate interval as an offset or stride format. Then, if the process passes the offset, it ignores the offset and the checkpoint on the next timing. This implementation is used with the directive-based method. In addition, we are considering the partial message logging (PML) method by the ppOpen-SYS/FT framework. This method partitions application processes as a group and uses a different checkpoint method on the intra-group and the inter-group. The partitioning depends on the communication amount and the frequency between processes. ppOpen-SYS/FT optimizes this partitioning with the topological method that we also proposed (Fig. 19).

Moreover, we are considering cooperation with ppOpen-MATH/MP. By focusing communication between the coupler and applications that construct coupling software, each application can perform checkpointing individually with our asynchronous coordination. However, because of the coupler keeping state, the consistency of the coupling software is broken. Consequently, ppOpen-SYS/FT supplies an API that informs whether the application will be a checkpoint until the next communication with the coupler. On each communication of coupled applications, the coupler calls the API for the decision of the checkpoint itself (Fig. 20). An additional API is also defined for AMR applications with ppOpen-APPL/FVM. The AMR application changes the checkpoint time drastically by re-partitioning. Then, ppOpen-SYS/FT supplies the optimized checkpoint timing and receives the checkpoint timing from the application decision.

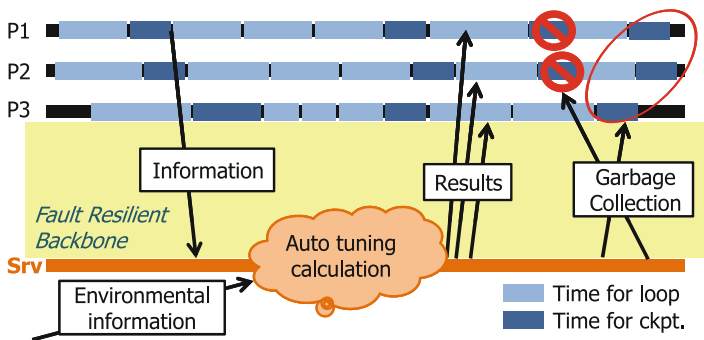


Fig. 19 Framework of ppOpen-SYS/FT

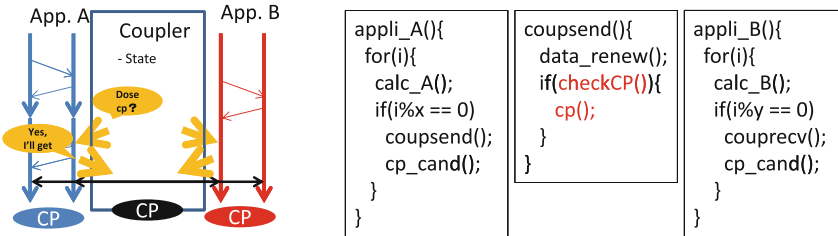


Fig. 20 Procedures for checkpointing of couplers

6 Summary

In this article, recent achievements and progress of the ppOpen-HPC project have been presented. The libraries developed for ppOpen-HPC are open for public use under MIT license and can be downloaded at the website of the project [1]. ppOpen-HPC has been installed on various types of supercomputers, and is utilized for research and development that requires large-scale supercomputer systems. Moreover, ppOpen-HPC is introduced in graduate and undergraduate classes at universities.

Currently, we are focusing on development and optimization of ppOpen-HPC for Intel Xeon/Phi architecture, and preparing for further research and development towards exascale systems.

Acknowledgments This work is supported by Core Research for Evolutional Science and Technology (CREST), the Japan Science and Technology Agency (JST), Japan.

References

1. ppOpen-HPC: <http://ppopenhpc.cc.u-tokyo.ac.jp/>
2. Nakajima, K.: ppOpen-HPC: open source infrastructure for development and execution of large-scale scientific applications on post-peta-scale supercomputers with automatic tuning (AT). In: ATIP '12 Proceedings of the ATIP/A*CRC Workshop on Accelerator Technologies for High-Performance Computing: Does Asia Lead the Way?, ACM Digital Library (ISBN: 978-1-4503-1644-6) (2012)
3. Joint Center for Advanced High Performance Computing (JCAHPC): <http://jcahpc.jp/>
4. Post-Peta CREST: <http://postpeta.jst.go.jp/en/>
5. GeoFEM: <http://geofem.tokyo.rii.or.jp/>
6. HEC-MW: <http://www.multi.k.u-tokyo.ac.jp/FrontISTR/>
7. Mori, F., Matsumoto, M., Furumura, T.: Performance optimization of the 3D FDM simulation of seismic wave propagation on the intel Xeon Phi coprocessor using the ppOpen-APPL/FDM library. In: Lecture Notes in Computer Science (LNCS) (in press)
8. Information Technology Center, The University of Tokyo: <http://www.cc.u-tokyo.ac.jp>
9. Matsumoto, M., Mori, F., Ohshima, S., Jitsumoto, H., Katagiri, T., Nakajima, K.: Implementation and evaluation of an AMR framework for FDM applications. *Procedia Comput. Sci.* **29**, 936–946 (2014)

10. Ida, A., Iwashita, T., Mifune, T., Takahashi, Y.: Parallel hierarchical matrices with adaptive cross approximation on symmetric multiprocessing clusters. *J. Inf. Process.* **22**(4), 642–650 (2014)
11. Ida, A., Iwashita, T., Ohtani, M., Hirahara, K.: Improvement of hierarchical matrices with adaptive cross approximation for large-scale simulation. *IPJS Trans. Adv. Comput. Syst.* **49** (in press)
12. Nishiura, D., Matsuo, M.Y., Sakaguchi, H.: ppohDEM: computational performance for open source code of the discrete element method. *Comput. Phys. Commun.* **185**, 1486–1495 (2014)
13. Arakawa, T., Inoue, T., Satoh, M.: Performance evaluation and case study of a coupling software Ppopen-MATH/MP. *Procedia Comput. Sci.* **29**, 924–935 (2014)
14. Satoh, M., Tomita, H., Yashiro, H., Miura, H., Kodama, C., Seiki, T., Noda, A.T., Yamada, Y., Goto, D., Sawada, M., Miyoshi, T., Niwa, Y., Hara, M., Ohno, T., Iga, S., Arakawa, T., Inoue, T., Kubokawa, H.: The non-hydrostatic icosahedral atmospheric model: description and development. In: *Progress in Earth and Planetary Science*, pp. 1–18 (2014)
15. Hasumi, H.: Documentaion for CCSR Ocean Component Model (COCO) Version 4.0s. Center for Climate System Research, April (2007)
16. Jones, P.H.: First- and second-order conservative remapping schemes for grids in spherical coordi-nates. *Mon. Weather Rev.* **127**, 2204–2210 (1999)
17. HPCG: High Performance Conjugate Gradients: <https://software.sandia.gov/hpcg/>
18. Nakajima, K.: Optimization of serial and parallel communications for parallel geometric multi-grid method. In: *Proceedings of the 20th IEEE International Conference for Parallel and Distributed Systems (ICPADS 2014)*, pp. 25–32 (2014)
19. Monakov, A., Lokhmotov, A., Avetisyan, A.: Automatically tuning sparse matrix-vector multiplication for GPU architectures. *Lect. Notes Comput. Sci.* **5952**, 112–125 (2010)
20. Nakajima, K.: Automatic tuning of parallel multigrid solvers using OpenMP/MPI hybrid parallel programming models. *Lect. Notes Comput. Sci.* **7851**, 435–450 (2013)
21. Katagiri, T., Ohshima, S., Matsumoto, M.: Auto-tuning of computation kernels from an FDM Code with ppOpen-AT. In: *Proceedings of IEEE MCSoc2014*, pp. 91–98 (2014). doi:[10.1109/MCSoc.2014.22](https://doi.org/10.1109/MCSoc.2014.22)
22. Jitsumoto, H., Todoroki, Y., Ishikawa, Y., Sato, M.: Grid-oriented process clustering system for partial message logging. In: *Proceedings of the 4th Fault Tolerance for HPC at eXtreme Scale (FTXS) 2014*, in conjunction with DSN2014 (2014)
23. Jitsumoto, H., Todoroki, Y., Sato, M.: Design and evaluations of application based fault tolerance framework with stencil model. In: *G8 ESC Workshop at Kobe* (2014)
24. Jitsumoto, H., Kamoshida, Y.: Application-level checkpoint/restart framework with optimal checkpoint interval. In: *HPC in Asia Workshop Poster Session at ISC'13* (2013)
25. Katagiri, T., Kise, K., Honda, H., Yuba, T.: FIBER: a general framework for auto-tuning software. *Proc. ISHPC-V, Lect. Notes Comput. Sci.* **2858**, 146–159 (2003)

Optimization in the Real World

Toward Solving Real-World Optimization Problems

Fujisawa, K.; Shinano, Y.; Waki, H. (Eds.)

2016, XII, 194 p. 69 illus., 28 illus. in color., Hardcover

ISBN: 978-4-431-55419-6