jh200036-MDHI

High resolution simulation of cardiac electrophysiology on realistic whole-heart geometries

Kengo Nakajima（The University of Tokyo）

Abstract

This international project in JHPCN aims to combine the expertise of The University of Tokyo in HPC and the expertise of Simula Research Laboratory (Norway) in cardiac modeling, with the objective of enhancing a 3D simulator of cardiac electrophysiology over realistic whole-heart geometries. The enhanced 3D simulator is expected to efficiently use modern supercomputers, such as Oakforest-PACS and Oakbridge-CX, to simulate realistic scenarios of electrophysiology in the heart. These high-resolution and biologically-detailed simulations are considered as an important tool for advancing the scientific understanding of the electrophysiology in the heart, and eventually for improving the medical treatment and drug design of various heart diseases.

The start of this 2-year project is an existing simulator of cardiac electrophysiology. We aim to carry out a series of improvements with respect to both the software implementation and the underlying numerical strategy. Specifically, SIMD vectorization will be enabled to deliver high single-core performance of the non-memory-traffic constrained computational tasks. OpenMP parallelization will be combined with MPI-based parallelization to achieve optimal single-node performance. Multi-thread-tasking will be investigated, together with suitable data restructuring, for alleviating the MPI communication overhead in multi-node simulations. Implicit integration in the time direction will be adopted and implemented to allow larger timesteps, thereby giving the potential of overall time saving.

1. Basic Information

(1) Collaborating JHPCN Centers

Information Technology Center, Univ. Tokyo

(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

- Kengo Nakajima (U Tokyo): Project administration, numerical algorithms and parallel programming.
- Xing Cai (Simula/Norway): Numerical algorithms, code parallelization and optimization, as well as project coordination together with Prof. Nakajima.
- Akihiro Ida (U Tokyo): Numerical algorithms and parallel programming.
- Toshihiro Hanawa (U Tokyo): Code parallelization, profiling and optimization.
- Masatoshi Kawai (U Tokyo): Numerical algorithms and parallel programming.
- Tetsuya Hoshino (U Tokyo): Code parallelization, profiling and optimization.
- Masaharu Matsumoto (U Tokyo): Numerical algorithms and parallel programming.
- Glenn Terje Lines (Simula/Norway): Cardiac electrophysiology, mathematical modeling.
- Johannes Langguth (Simula/Norway): Code parallelization, profiling and optimization.
- Jonas van den Brink (Simula/Norway): Preparation of geometries and physiological parameters for subcellular simulations.
- Kristian Gregorius Hustad (Simula/Norway): Code parallelization, profiling and optimization.
- Hermenegild Arevalo (Simula/Norway): Preparation of geometries and parameters, running simulations, result analysis.

2. Purpose and Significance of Research

Coordinated electrical activities in the heart are vital for its function. Biophysically accurate simulations of cardiac electrophysiology require extremely fine spatial and temporal resolutions. In addition, unstructured computational meshes must be adopted to precisely capture the realistic 3D geometry of the heart. The relevant mathematical model in this context is a nonlinear

3D reaction-diffusion equation, with the transmembrane electrical potential being the primary unknown field, while calcium handling in each computational cell is described by a system of nonlinear ordinary differential equations (ODEs). Such a whole-heart simulation, if using tens of millions of computational cells, can easily take hours or even days of execution time by current mainstream simulation codes when running on a small or medium-scale cluster of multicore CPUs. The slow turn-around times have so far limited the use of such ambitious simulations in a realistic setting where computational cardiologists can for example experiment, "in-silico", about how/when/where arrhythmia arises, or the sensitivity of certain dysfunctions of the heart with respect to the calcium channel mechanism and/or the conductivity properties of the heart.

In this project, we aim to increase the scale of state-of-the-art whole-heart simulations by tenfold, i.e., using hundreds of millions of computational cells instead of tens of millions. At the same time, we have the ambition of reducing the turn-around time from hours to minutes. We will achieve these ambitious goals by combining performance-enhancing numerical strategies, hardware-aware code optimization and parallelization-overhead reduction. It is also remarked that the core computations, i.e., numerically solving partial differential equations (PDEs) on unstructured meshes and millions of ODE systems, are present in many other computational science applications. This means that the advances achieved in this proposed project will extend well beyond the domain of cardiac electrophysiology simulations.

3. Significance as JHPCN Joint Research Project

The necessity of implementing this JHPCN joint research project is due to two aspects. First, UTokyo has world-leading expertise in implementing and optimizing advanced numerical code. This expertise has been built up via developing real-world applications for running on cutting-edge supercomputers at UTokyo. Such hands-on experience on supercomputing is lacking for the Norwegian partner. Second, the Oakforest-PACS and Oakbridge-CX systems (plus the upcoming Wisteria system) are of a suitable size for achieving the ambitious goal of this project, whereas access to world-leading supercomputers has been very scarce for the Norwegian partner. The two hardware systems also provide a valuable testbed for the performance portability of the simulation codes to be developed. The high-speed file cache systems available at UTokyo also provide good possibilities of in-situ huge-scale data analysis.

4. Outline of Research Achievements up to FY2019

jh20036 is is partially related to, but not a direct continuation of, JHPCN project jh180024/jh190040 (Physiologically realistic study of subcellular calcium dynamics with nanometer resolution), which has obtained good results about optimizing cardiac simulations on regular computational meshes.

5. Details of FY2020 Research Achievements

(1) Overview

Overall, the 3D simulator of cardiac electrophysiology carries out a time integration where each iteration consists of first individually solving a system of nonlinear ODEs (the so-called *cell model* that models the transmembrane ionic currents and cellular calcium handling) on each computational cell, and then solving a 3D diffusion equation (PDE) that couples all the computational cells in a realistic whole-heart geometry. The research achievements during FY2020 center around improving the performance of the ODE and PDE computations, which are detailed below. Generally, problems are solved explicitly in time direction, where invserins of large-scale matrices are not required. Because implicit integration in the time direction allows larger timesteps, thereby giving the potential of overall time saving, we plan to introduce and implement implicit integration in time direction. In FY.2020, preliminary studies for implicit methods using multigrid methods were also conducted.

(2) Enabling SIMD vectorization of ODE computation

The numerical solution of the ODEs per computational cell typically adopts explicit time integrators such as the forward Euler (FE) method or the general first-order Rush-Larsen (GRL1) method. The associated computational intensity (number of floating-point operations per memory load/store) is relatively high. This means that enabling SIMD vectorization on modern CPU architectures is important for achieving good performance of the ODE computation, which was lacking on the existing simulator before the project start.

There are in general two strategies for SIMD vectorization. The first is to rely on the automatic vectorization capability of compilers, whereas the second is to directly insert SIMD vectorization intrinsics (such as AVX2 or AVX-512) or use a portable vectorization library such as VCL. During FY2020 we have tested both strategies of SIMD vectorization.

To help trigger automatic vectorization by the compiler, we have found that it is important to restructure the overall data structure in the following way. Instead of storing together the various ODE state variables for each cell (such that the data structure for all the cells is "an array of structs"), which is adopted by the old simulator, we have grouped each state variable from all the cells together (such that the overall data structure is now "a struct of arrays"). Another useful hint given to the compiler is the **omp simd** directive associated with OpenMP parallelization of the loop that traverses all the cells. (The ODE computation is embarrassingly parallel over the cells.) On the Oakbrige-CX system, in order to encourage the compiler to automatically adopt AVX-512 intrinsics instead of AVX2, we found it necessary to add the clause of **simdlen(8)** in addition, which already seems to be the default behavior of the Intel compiler on Oakforest-PACS. For comparison, we have also used the VCL library to explicitly vectorize the ODE computation. Experiments have shown that this manual approach does not bring any noticeable performance benefits over the automatic compiler vectorization (at least when AVX-512 intrinsics are automatically enabled). Therefore, the automatic vectorization approach is favored, due to its simplicity and minimum code restructuring required.

In the following, we report the achieved ODE computation performance in various tests. These experiments cover both Oakforest-PACS and Oakbridge-CX, and three biologically detailed cell ODE models are investigated: the 19-state Ten-Tusscher-Panfilov model from 2006 (TT06), the 24-state Jæger-Tveito model (JT), and the 39-state Grandi-Pasqualini-Bers model (GPB). The used ODE solvers are the forward-Euler (FE) method and the generalized first-order Rush-Larsen (GRL1) method. For all the experiments, instead of reporting the detailed time results, we use the performance metric as millions of cell steps executed per second (the higher the better). We can see from Figures 1-8 that compiler-enabled automatic vectorization provides substantial performance improvement over a naïve version without SIMD vectorization. The Oakbridge-CX system provides better ODE performance than the Oakforest-PACS system, in both single-thread and single-node scenarios, independent of the cell model and the ODE solver. method.

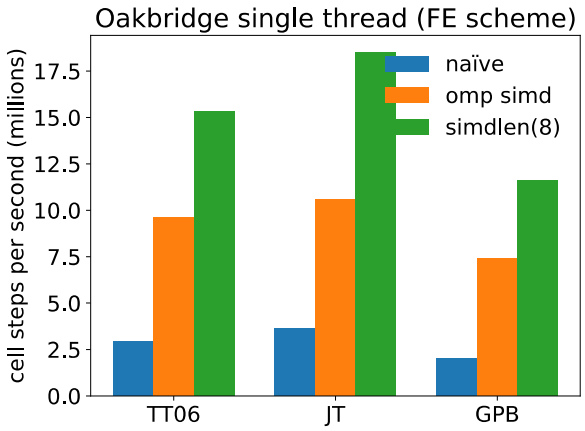

Figure 1 Single-thread ODE performance on Oakbridge

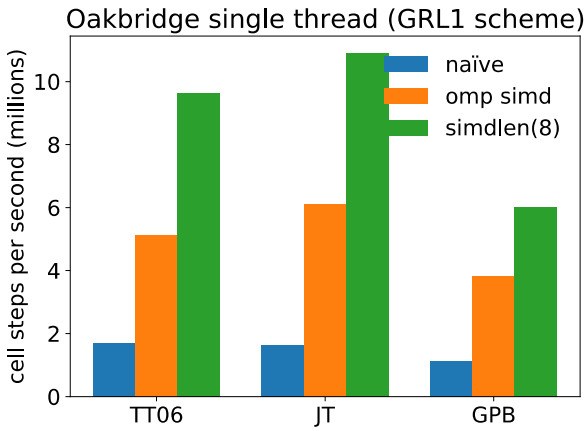

Figure 2 Single-thread ODE performance on Oakbridge

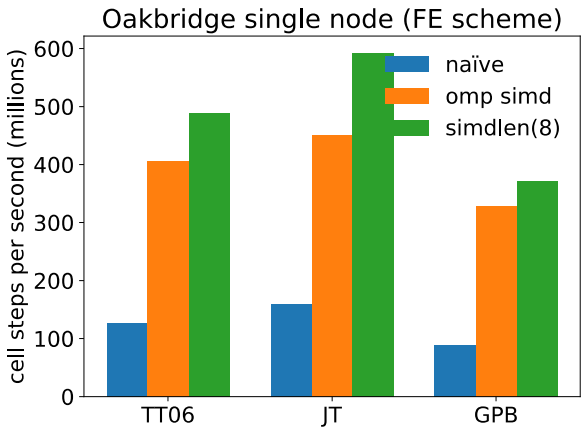

Figure 5 Single-node ODE performance on Oakbridge

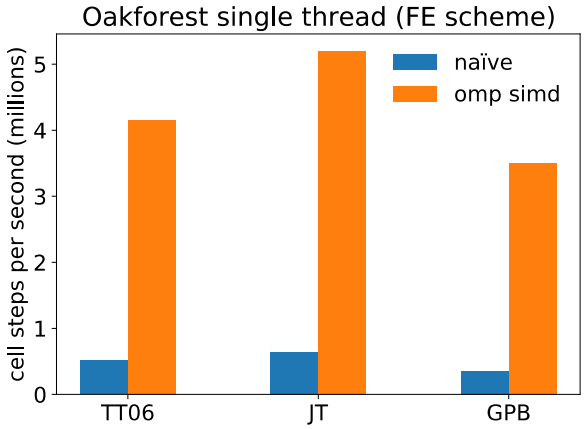

Figure 3 Single-thread ODE performance on Oakforest

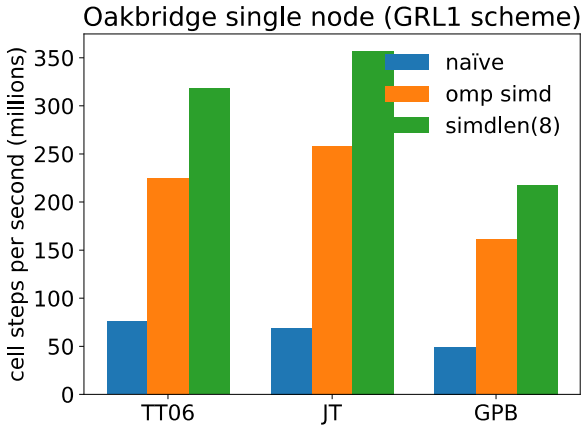

Figure 6 Single-node ODE performance on Oakbridge

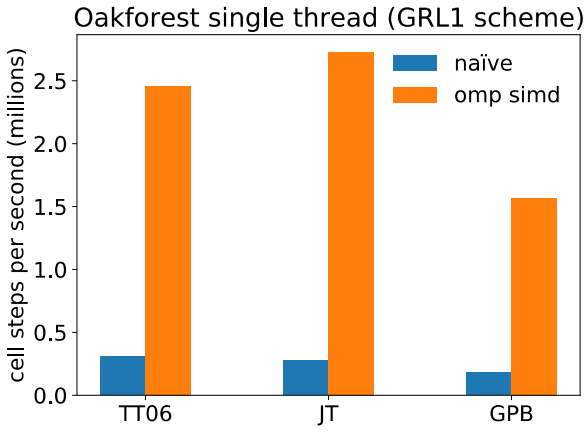

Figure 4 Single-thread ODE performance on Oakforest

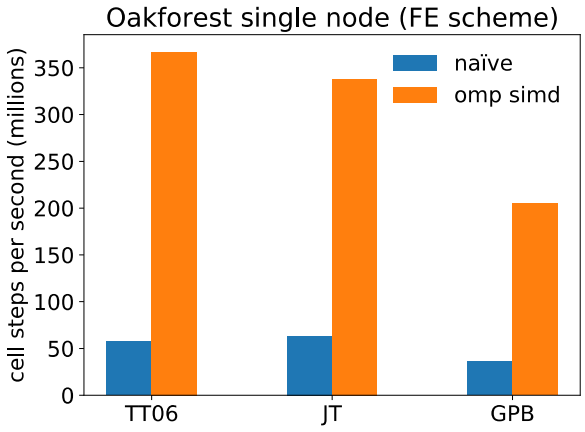

Figure 7 Single-node ODE peformance on Oakforest

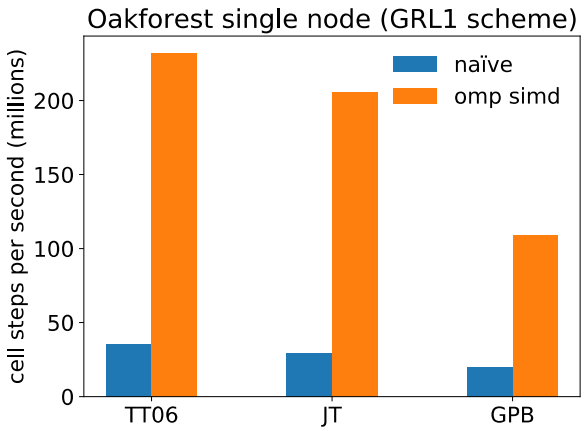

Figure 8 Single-node ODE performance on Oakforest

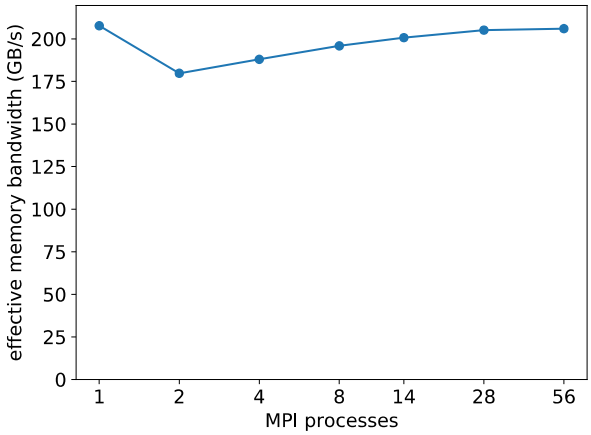

Figure 9 Single-node PDE performance on Oakbridge

(3) Mixing OpenMP and MPI parallelization for PDE computation

Each compute node of the Oakbridge-CX and Oakforst-PACS systems has, respectively, 56 and 68 cores. The latter is also well known to support 4 threads per core. The old simulator focused on using MPI parallelization both inter-node and intra-node, thus having the risk of extensive MPI overhead when using a large number of compute nodes. We have therefore enabled OpenMP+MPI parallelization for the PDE computation (the same as for the ODE computation).

In Figure 9, we study the obtained PDE performance on a single compute node of Oakbridge-CX, where we change the number of MPI processes used while keeping the total number of OpenMP threads at 56. The reported PDE performance uses the metric of "effective memory bandwidth achieved", which is calculated as the minimum incurred amount of memory traffic (assuming perfect caching) divided by the time used. We can see that using one MPI process that spawns 56 OpenMP threads gives the best single-node PDE performance. It can also be observed that the effective memory bandwidth achieved is quite close to the STREAM benchmark measured memory bandwidth, meaning that the single-node PDE performance is approaching its realistic upper limit on Oakbridge-CX.

On the Oakforest-PACS system, good single-node PDE performance is also associated with using a single MPI process that spawns at least 68 OpenMP threads. In Figure 10, we further study the effect of using more than one thread per core. It shows that the best single-node PDE performance is associated with using one MPI process that spawns in total 272 OpenMP threads.

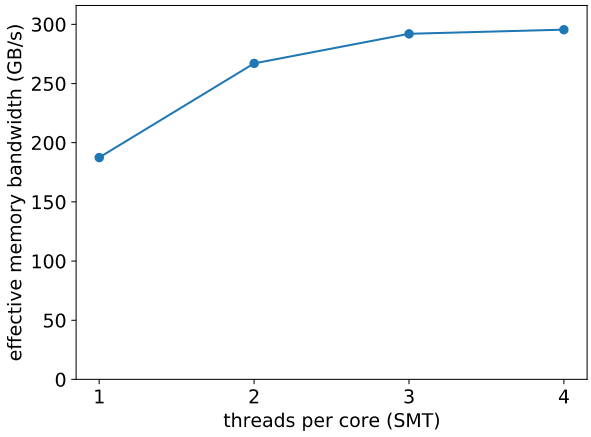

Figure 10 Single-node PDE performance with respect to OpenMP threads per core

(4) Multi-node strong scaling study

We have also measured the strong scaling results of running a parallel simulation (ODE+PDE) of cardiac electrophysiology, which uses a realistic whole-heart geometry that involves over 14 million computational cells (each being a tetrahedron). As shown in Figure 11, up to 256 compute nodes have been used on Oakbridge-CX, where each node always uses a single MPI process that spawns 56 OpenMP threads. The performance metric

used is number of cell steps computed per second. It can observed a gap between the actual parallel performance and the perfect scaling, mostly indicating the impact of MPI communication overhead.

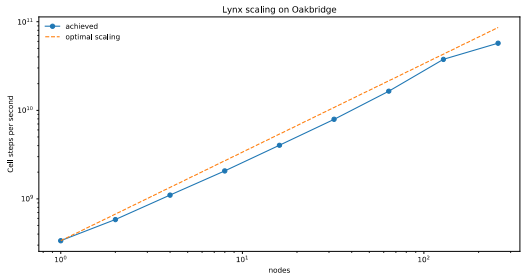

Figure 11 Strong scaling measurements on Oakbridge

(5) Multigrid Method for Impicit Time Integration

The parallel multigrid method is expected to play an important role in scientific computing on exa-scale supercomputer systems for solving large-scale linear equations with sparse coefficient matrices. Because solving sparse linear systems is a very memory-bound process, efficient method for storage of coefficient matrices is a crucial issue. Nakajima et al. implemented sliced ELL method to parallel conjugate gradient solvers with multigrid preconditioning (MGCG) for the application on 3D groundwater flow through heterogeneous porous media (pGW3D-FVM), and excellent performance has been obtained on large-scale multicore/manycore clusters [Nakajima, K., IEEE ICPADS 2014, 2014]. In the present work, we introduced SELL-C-σ with double/single precision computing to the MGCG solver, and evaluated the performance of the solver with OpenMP/MPI hybrid parallel programing models on the Oakforest-PACS (OFP) system using up to 2,048 nodes of Intel Xeon Phi. Because SELL-C-σ is suitable for wide-SIMD architecture, such as Xeon Phi, improvement of the performance over the sliced ELL was more than 35% for double precision and more than 45% for single precision on OFP.

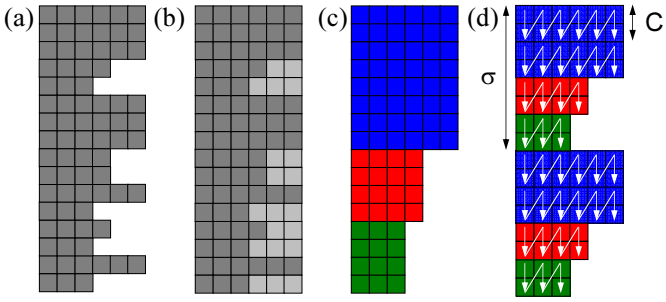

Figure 12 Formats of sparse matrix storage. (a) Compressed row storage (CRS); (b) Ellpack-Itpack (ELL), (c) Sliced ELL, (d) SELL-C-σ (SELL-2-8)

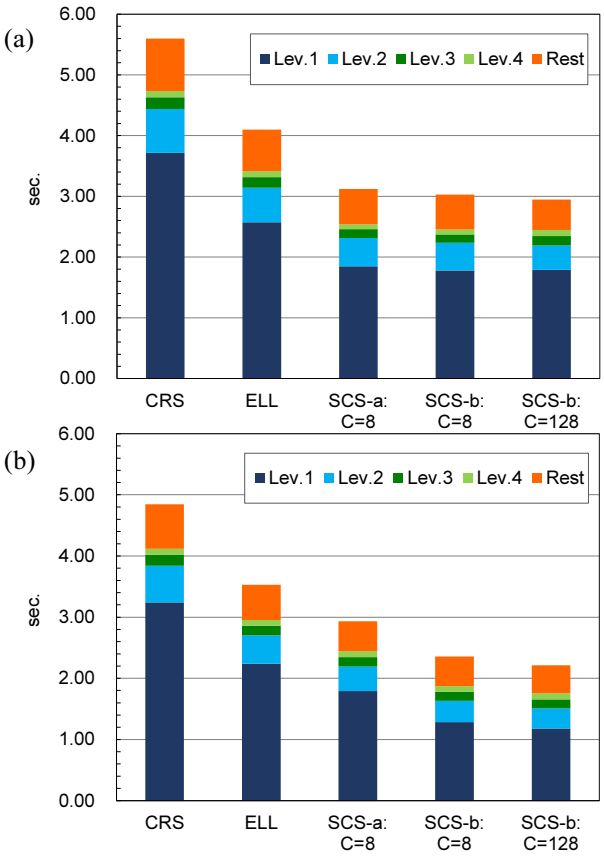

Figure 13 Elapsed Computation Time for MGCG Solver: 8 nodes of OFP (HB 4×16), 33,554,432 DOF, (a) Double Precision (FP64), (b) Single Precision (FP32)

Figure 13(a) and Figure 13(b) show computation time of MGCG for HB 4×16 with 8 nodes of OFP, where total problem size is 33,554,432 DOF. Computation time of MGCG for HB 4×16 with 8 nodes of OFP. Each of *Lev.h* (*h*=1 (finest level)~4 (coarser level)) shows total time for smoothing at the *h*-th level of multigrid computing, while *Rest* includes time for communications, coarse grid solver and conjugate gradient solver except multigrid. SELL-8-8

and SELL-128-128 are applied to SCS-b. Improvement of performance of MGCG solver over CRS is shown in Table 1. Generally, improvement of performance by SCS-b is excellent, and the effects are more significant in FP32 cases. Generally, performance improvement by single precision for CRS is lower compared to Sliced ELL and SELL-C-σ.

Table 1. Summary of Results in Fig.9 for OFP

a. Improvement over CRS (MGCG solver, Level-1 of Smoother)

	Double Precision (FP64)	Single Precision (FP32)
Sliced ELL	36.6%, 46.6%	15.6%, 44.7%
SCS-a (C=σ=8)	79.4%, 101.3%	58.7%, 81.0%
SCS-b (C=σ=8)	84.9%, 108.7%	90.9 %, 152.2%
SCS-b (C=σ=128)	90.1%, 107.8%	137.5%, 174.9%

b. Improvement over Sliced ELL (MGCG solver, Level-1 of Smoother)

	Double Precision (FP64)	Single Precision (FP32)
SCS-a (C=σ=8)	31.4%, 39.2%	20.3%, 25.1%
SCS-b (C=σ=8)	35.4%, 44.4%	49.6%, 74.3%
SCS-b (C=σ=128)	39.2%, 46.4%	59.4%, 89.9%

Performance of weak scaling has been evaluated using up to 2,048 nodes of OFP (131,072 cores). Maximum problem size is 8,589,934,592 DOF. Figure 14 shows results of weak scaling up to 2,048 nodes of OFP using HB 4×16, where "-d" denotes *double precision* (FP64), and "-s" is for *single precision* (FP32).

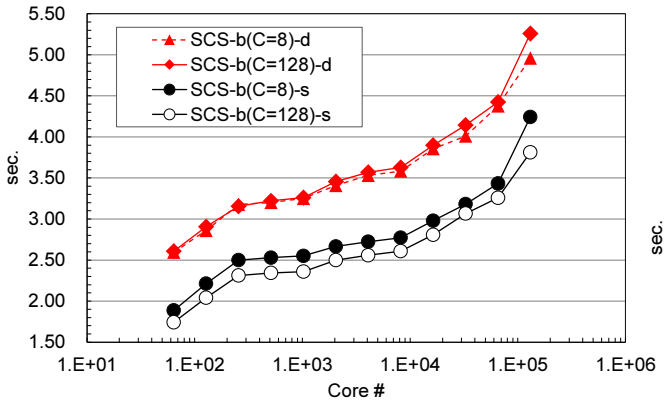

Figure 14 Elapsed Computation Time for MGCG Solver, Weak Scaling up to 2,048 nodes, Max. Problem Size: 8,589,934,592 DOF, HB 4×16, "-d": Double Precision (FP64), "-s": Single Precision (FP32)

Figure 15 compares double precision (FP64) and single precision (FP32) for both of SCS-b (C=σ=8) and SCS-b (C=σ =128) with HB 4×16. In both of double precision and single precision, performance of SCS-b (C=σ=8) and SCS-b (C= σ =128) is competitive, but SCS-b (C= σ =128) is slightly faster for single precision.

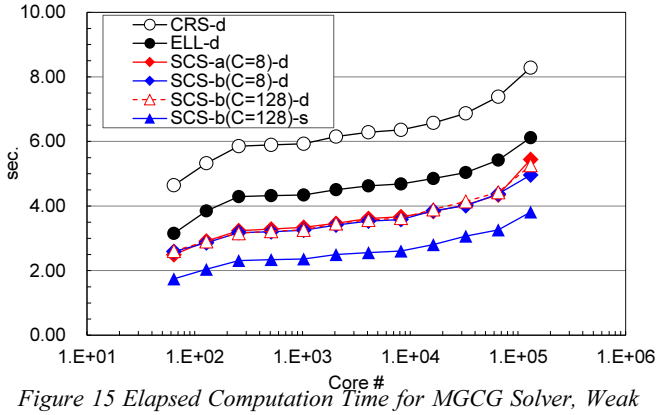

Figure 15 Elapsed Computation Time for MGCG Solver, Weak Scaling up to 2,048 nodes, Max. Problem Size: 8,589,934,592 DOF, HB 4×16, "-d": Double Precision (FP64), "-s": Single Precision (FP32)

Figure 16 compares HB 4×16 and HB 8×8 for SCS-b (C=σ =8) with double precision and for SCS-b (C=σ=128) with single precision. Generally speaking, HB 4×16 is faster than HB 8×8, but performance is similar at 1,024 and 2,048 nodes. Problem of CGA (Coarse Grid Aggregation) adopted in this work (Figure 17) is that the *coarse grid solver* works on a single MPI process and, problem size of the *coarse grid solver* is proportional to total number of MPI processes. Therefore, cost of the coarse grid solver is more significant with many nodes. Moreover, this effect is more significant, if number of threads for each MPI process is smaller, such as HB 4×16 in this work.

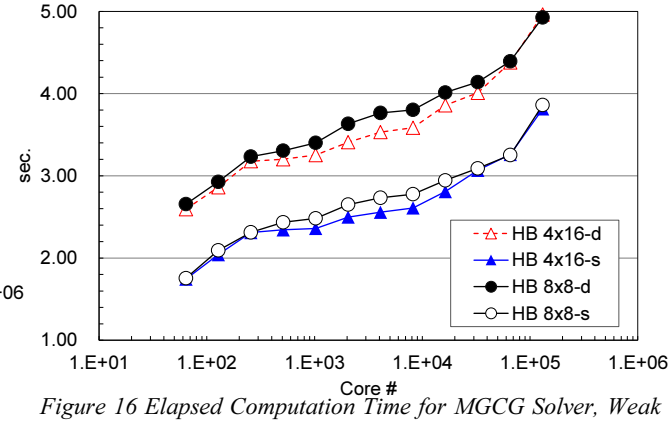

Figure 16 Elapsed Computation Time for MGCG Solver, Weak Scaling up to 2,048 nodes, Max. Problem Size: 8,589,934,592 DOF, Comparison of HB 4×16 and HB 8×8, "-d": Double Precision (FP64), SCS-b (C=σ=8), "-s": Single Precision (FP32), SCS-b (C=σ=128)

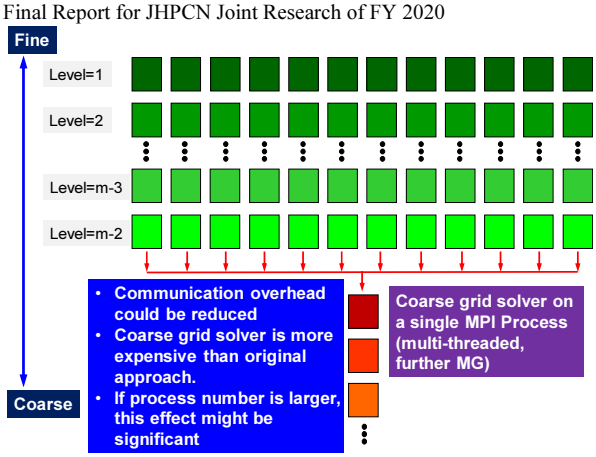

Figure 17 Procedures of coarse grid aggregation (CGA), where information of each MPI process is gathered in a single MPI process for computation at level=m-2 [14]

6. Progress during FY2020 and Future Prospects

Following the original 2-year project plan, we have carried out two activities in FY2020: (1) Single-node optimization of an explicit-method based simulator; (2) Scale-out optimization of the explicit-method based simulator, and (3) Preliminary Works on Mulgirgrid Solver. In (1) and (2), we have achieved close-to-maximum single-node performance, thanks to SIMD vectorization of the ODE computation and an effective OpenMP+MPI parallelization of the PDE computation. The multi-node scale-out optimization has had a good start, but will require more detailed profiling to identify potential optimization opportunities. This activity (1) and (2) will be continued in FY2021, while we will also carry out the other two planned research activities: (4) Development of a new, implicit-method based simulator (based on (3) in FY.2020); (5) Large-scale, realistic simulations of cardiac electrophysiology.

7. List of Publications and Presentations

[1] Journal Papers (Refereed)

[1] K.H. Jæger, K.G. Hustad (+), X. Cai (+), A. Tveito. *Efficient numerical solution of the EMI model representing the extracellular space (E), cell membrane (M) and intracellular space (I) of a collection of cardiac cells*. Frontiers in Physics, 8: 579461, 2021, DOI: 10.3389/fphy.2020.579461

[2] Proceedings of International Conferences (Refereed)

[2] K.H. Jæger, K.G. Hustad, X. Cai, A. Tveito. *Operator Splitting and Finite Difference Schemes for Solving the EMI Model.* Book chapter in "Modeling Excitable Tissue: The EMI Framework", pages 44-55, 2021, Springer, DOI: 10.1007/978-3-030-61157-6_4

[3] K.G. Hustad, X. Cai, J. Langguth, H. Arevalo, *Efficient simulations of patient-specific electrical heart activity on the DGX-2.* Poster presented at the GTC-2020 Conference.

[4] J. Langguth, N. Gaur, H. Arevalo, C. Jarvis, N. Altanaite, Q. Lan, X. Cai. *Towards detailed Organ-Scale Simulations in Cardiac Electrophysiology.* Poster presented at the GTC-2020 Conference.

[5] Nakajima, K., Gerofi, B., Ishikawa, Y., Horikoshi, M., Efficient Parallel Multigrid Solver on Intel Xeon Phi Cluster, IXPUG (Intel Extreme Performance Users Group) HPC Asia 2021, 2021

[6] Nakajima, K., Ogita, T., Kawai, M., Efficient Parallel Multigrid Methods on Manycore Clusters with Double/Single Precision Computing, IEEE Proceedings of the 16th International Workshop on Automatic Performance Tuning (iWAPT 2021) in conjunction with 35th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2021), 2021 (in press)

[3] International conference Papers (Non-refereed)

[4] Presentations at domestic conference (Non-refereed)

[5] Published library and relating data

[6] Other (patents, press releases, books and so on)