Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures
Final Report for JHPCN Joint Research of FY 2022

jh220041

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

Kengo Nakajima（The University of Tokyo）

Abstract
The overall goal of this international JHPCN project is to enable efficient simulations of cardiac electrophysiology on realistic whole-heart geometries, because these high-resolution and biologically-detailed simulations can help advancing the scientific understanding of the heart. The collaboration between The University of Tokyo and Simula Research Laboratory (Norway) has combined and further strengthened the expertise areas of the two partners, namely, high-performance computing and cardiac modeling. Continuing the work that was carried out in FY2021, the project partners have during FY2022 investigated the following aspects: 1. A second version of the simulator that is based on an implicit time integration scheme has been further studied with respect to the numerical stability and the overall computational efficiency, in comparison with the original explicit time integration scheme. 2. Advanced preconditioning techniques have been studied in connection with the new simulator based on implicit time integration which requires solving a sparse linear system per time step. 3. More effort has been invested in studying opportunities of further code optimization, for example, the role of work distribution among OpenMP threads with respect to the effectiveness of communication-computation overlap. 4. Both versions of the simulator have been ported from the CPU platforms to Nvidia GPUs, using OpenACC directives and Standard Parallelism (StdPar). 5. Experimentation with huge-scale simulations has been carried out, involving unstructured computational meshes with up to 1.5 billion tetrahedral elements, for the purpose of studying the scalability of the simulator and identifying further opportunities of performance improvement. Two supercomputer systems, Oakbridge-CX and Wisteria/Odyssey (including Wisteria/Aquarius), have been heavily used during FY2022 for the above-mentioned research activities.

1. Basic Information

(1) Collaborating JHPCN Centers

Tokyo

(2) Theme Area (Please choose one)

Large scale computational science area

(3) Research Areas (Choose one area, only for HPCI resource using project)

Very large-scale numerical computation

Very large-scale data processing

Very large-scale information systems

(4) Project Members and Their Roles

Kengo Nakajima (U Tokyo): Administration, numerical algorithms (alg.)

Xing Cai (Simula/Norway): Administration, numerical alg.

Akihiro Ida (JAMSTEC): Numerical alg.

Toshihiro Hanawa (U Tokyo): Optimization

Akira Naruse (NVIDIA): Optimization

Masatoshi Kawai (U Tokyo): Numerical alg.

Tetsuya Hoshino (U Tokyo): Optimization

Yohei Miki (U Tokyo): Optimization

Masaharu Matsumoto (Fukushima Univ): Numerical alg.

Glenn Terje Lines (Simula/Norway): Cardiac electrophysiology, mathematical modeling

Johannes Langguth (Simula/Norway): Optimization

Jonas van den Brink (Simula/Norway): Cardiac electrophysiology, mathematical modeling, cardiac simulations.

Kristian Gregorius Hustad (Simula/Norway): Optimization, cardiac simulations

Hermenegild Arevalo (Simula/Norway): Cardiac electrophysiology, mathematical modeling, cardiac simulations

James Trotter (Simula/Norway): Mathematical modeling, optimization

Aadrash Bussooa (Simula/Norway): Cardiac simulations

Maria Hernandez Mesa (Simula/Norway): Cardiac electrophysiology, cardiac simulations

Lena Myklebust (Simula/Norway): Cardiac electrophysiology, numerical simulations
Lisa Pankewitz (Simula/Norway): Cardiac electrophysiology, numerical simulations

2. Purpose and Significance of the Research

The proper functioning of the heart relies on coordinated electrical activities. However, many questions about cardiac electrophysiology still remain unanswered. It is thus important to develop simulation codes that can effectively use modern supercomputers for doing “in-silico” experiments, which can assist in the scientific understanding of the underlying mechanisms of cardiac electrophysiology. This project extends the project partners’ already obtained expertise in very large-scale simulations over simple computational meshes, into the regime of large-scale unstructured meshes. Such meshes are essential for faithfully simulating realistic scenarios of the heart. Successful results from this project will help to advance the field of cardiac electrophysiology. Moreover, experience obtained in effectively using the latest processor architectures are applicable to many other scenarios of computation over unstructured meshes.

3. Significance as JHPCN Joint Research Project

The significance of this JHPCN joint research project is due to two aspects. First, The University of Tokyo 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 Univ. Tokyo. Such hands-on experience on supercomputing is lacking for the Norwegian partner whose main expertise is on cardiac modeling and simulation. Second, the Oakbridge-CX system and the Wisteria/BDEC-01 system (Odyssey & Aquarius), 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 chosen hardware systems, in particular Wisteria/BDEC-01, also provide realistic and future-oriented testbeds for the performance portability of the simulation codes to be developed.

4. Outline of Research Achievements up to FY2021 (Only for continuous projects)

This project is a continuation of project jh210021. During FY2021, investigations were carried out in the following topics. (1) Performance enhancement of the original explicit-time-integration based simulator. This topic included applying SIMD vectorization and lookup tables to the part of the simulator that solves a system of nonlinear ordinary differential equations (ODEs) per mesh point. Improved performance of the ODE part was obtained on the Oakforest-PACS and Oakbridge-CX systems. Re-ordering of mesh entities and overlapping MPI communication with computation were also studied in the performance enhancement topic. (2) Development of a new simulator that is based on implicit time integration. The motivation is that the original explicit-time-integration based simulator often suffers from insufficient numerical instability (or even complete instability), which in turns means having to use extremely small time step sizes (or not being able to simulate at all). A prototype version of the implicit-time-integration based simulator was completed, together with some tests of applying algebraic multigrid as the preconditioner of the newly appeared linear system solving step. (3) Large-scale

simulations, where a computational mesh consisting of 55 million tetrahedral computational cells was used to perform scalability tests on three different systems (Oakforest-PACS, Oakbridge-CX & Wisteria/Odyssey). (4) Porting of the explicit-time-integration based simulator to the A64FX architecture, so that the simulator could run on Wisteria/Odyssey. The correctness of the simulation results were verified, and the benefits of communication-computation overlap was also confirmed on Wisteria/Odyssey.

5. Details of FY2022 Research Achievements

(1) Study of an implicit version of the simulator

As mentioned earlier, a second version of the simulator was implemented during FY2021, based on an implicit time integration scheme. The difference between this implicit time-integration based simulator and the original explicit time-integration based simulator is that the 3D diffusion equation, which is the PDE part of the whole mathematical model, is discretized in time by backward Euler instead of forward Euler. The motivation is that the implicit PDE solver is expected to be considerably more stable numerically, allowing much larger time steps than its explicit counterpart. (The ODE part of the mathematical model typically allows relatively large time steps, so any constraint on the time step size is posed by the PDE part.) The disadvantage of the implicit PDE solver is that it requires solving a sparse linear system per time step, whereas the explicit counterpart only needs to execute one sparse matrix-vector (SpMV) multiply operation per time step.

The new implicit time-integration based simulator was not properly studied during FY2021, thus an investigation was carried out in FY2022. There are two questions: 1. How much more numerically stable is the implicit PDE solver than its explicit counterpart? 2. Does the advantage in numerical stability translate into a performance upper hand? To answer these two questions, we chose a realistic whole-heart geometry and adopted two unstructured tetrahedral meshes of different resolutions, with 3 and 55 million tetrahedral cells, respectively. For the coarse mesh of 3 million cells, we used 64 nodes on Wisteria/Odyssey, with 4 MPI processes per node and 12 OpenMP threads per MPI process. Both the original explicit time-integration based simulator and the new implicit time-integration based simulator were executed to simulate the electrical activity in the heart during a so-called epical pacing scenario, where the ODE part is updated on each "outer" time step of size 200μs. Inside an "outer" step, we allowed multiple "inner" steps (the PDE solver is called per "inner" step), where the size of the "inner" step was tested between 5μs and 200μs. For the fine mesh of 55 million cells, we used 512 nodes on Wisteria/Odyssey, and the "inner" step size was experimented between 0.5μs and 40μs. Otherwise the same simulation and hardware configuration were used.

Table 1. Comparison between explicit and implicit simulators on a 3-million cell mesh

Explicit version		Implicit version		
Inner step	Time usage	Inner step	Time usage	Avg iters
5μs	4.9 ms	5μs	56.4 ms	258.32
N/A	N/A	10μs	37.6 ms	171.91
N/A	N/A	20μs	25.5 ms	123.00
N/A	N/A	50μs	17.8 ms	88.53
N/A	N/A	100μs	14.9 ms	76.03
N/A	N/A	200μs	14.3 ms	74.75

Table 2. Comparison between explicit and implicit simulators on a 55-million cell mesh

Explicit version		Implicit version		
Inner step	Time usage	Inner step	Time usage	Avg iters
0.5μs	195.5 ms	0.5μs	1192.3 ms	1256
N/A	N/A	1μs	670.8 ms	748.0
N/A	N/A	10μs	224.6 ms	336.2
N/A	N/A	20μs	418.7 ms	669.7
N/A	N/A	40μs	307.2 ms	670.0

In Tables 1 and 2, we have reported the average total time usage per "outer" time step of the

explicit and implicit simulators. We have also listed, for the implicit simulator, the average total number of conjugate gradient iterations needed to solve the sparse linear systems accumulated over all the "inner" steps within an "outer" time step. We can observe that the explicit simulator easily becomes numerically unstable, can only use the smallest "inner" step size, whereas the implicit simulator is always numerically stable and work with all the "inner" time step sizes. With respect to the time usage, the implicit simulator did not manage to beat the explicit counterpart, although on the fine mesh (55 million cells) the distance is quite close. It needs to be commented that no preconditioner was used for the conjugate gradient iterations, so the answer to the second question above needs further experiments.

(2) Advanced preconditioner based on AMG

To reduce the computational time of Simula simulation, we investigate the implicit time integration scheme. We can expect that the implicit time integration scheme is faster than the explicit time integration scheme because of the 10 or 100 times larger width of the time step. Then, in the implicit time integration scheme, we must solve a Poisson equation with a large degree of freedom(DoF) for updating the biopotential in heart simulation at each step. Therefore, the development of a fast solver is essential. The coefficient matrix of a system of linear equations derived from simula simulation has different properties depending on the width of the time step. The larger width of the step leads to the stronger dominance of the off-diagonal components and the larger condition number. Therefore, it requires a solver with stability and tolerance to the large condition number. This solver also must support large-scale problems. A Krylov subspace method with an algebraic multigrid (AMG) precondition is suitable for such conditions. The preconditioned Krylov subspace method is widely used to stably solve systems of linear equations with a large condition number. Especially, the AMG precondition has characteristics that the convergence ratio is not depending on the DoF of the target problem if the target system of linear equation is derived from the geometrical structures. The mesh used in the Simula simulation is based on a geometric structure (hexahedral mesh), we can expect that the AMG precondition shows effectiveness.

In parallelizing AMG preprocessing, the following issues are important: 1) How to generate coarse grids in a massively parallel environment, 2) Implementation of a fast triple sparse matrix product, 3) Adoption of an effective smoother, and 4) Implementation of a fast sparse matrix-vector product.

As for the issue 1, the sequentiality of the coarse grid generation is an impediment to parallelization. Then we have applied the multi-coloring approach for parallelizing the coarse grid generation (Fig.1). Although parallelization based on a domain decomposition approach is available. However, it has a large impact on the convergence ratio. Compared with the domain decomposition approach, the multi-coloring approach shows a similar convergence ratio to the sequential. The multi-color ordering is also parallelized with a hierarchical approach for achieving complete parallelization.

As for the issue 2, we have used the PETc library for simple implementation because of the difficulty of the implementation of the matrix products with sparse matrices. As for the issue 3, we are implementing a multiplicative Schwarz-type block multi-color Gauss-Seidel (MS-BMC-GS) smoother. This smoother shows a better convergence ratio and higher cache locality than

general Gauss-Seidel smoother. By using the AMG preconditioner shows high effectiveness than existing smoother (Fig. 3).

As for the issue 4, we are planning to implement a Sell-C-σ storage format to improve the performance of the solver. Compared with the CRS storage format, the Sell-C-σ shows better performance in the effective use of SIMD instruction and higher cache locality.

As for the overall implementation of the AMG preconditioned Krylov subspace method, we have completed the implementation of the parallel coarse grid generation and triple sparse matrix product with the PETc library. Now we are implementing the multiplicative Schwarz-type block multi-colored Gauss-Seidel smoother. After this implementation is completed, it will be evaluated and applied to the Simula simulation.

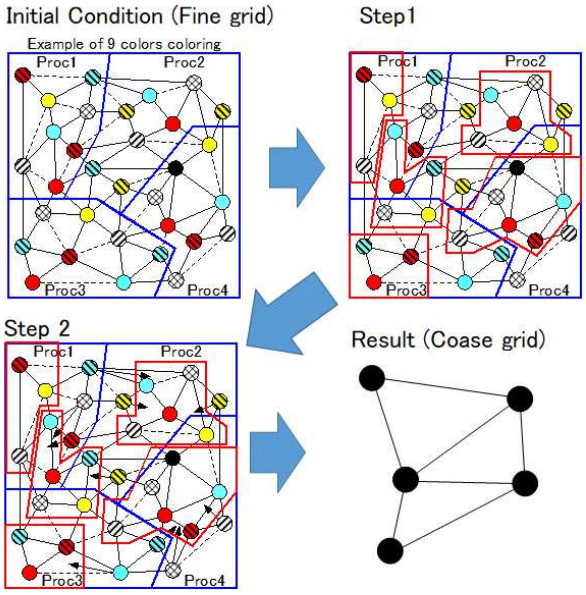

Fig. 1 Process of parallel coarse grid generation

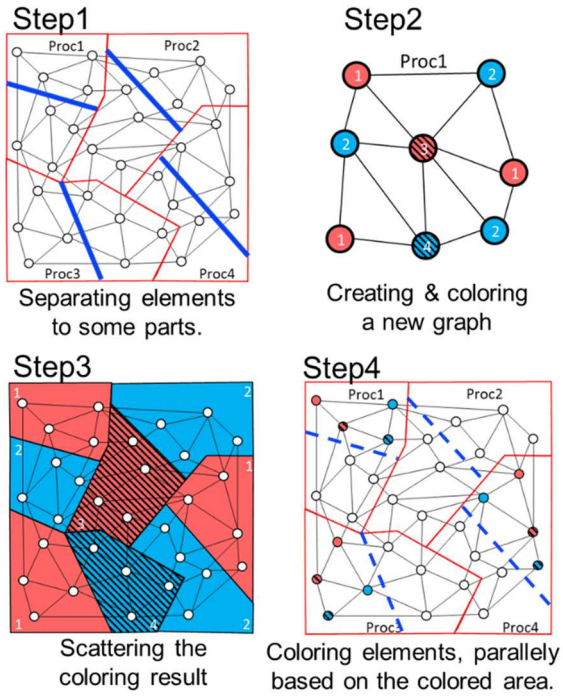

Fig. 2 Hierarchical parallel multi-color ordering

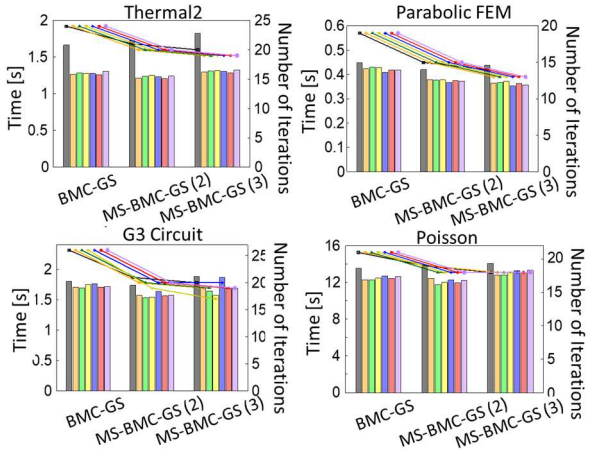

Fig3. The effectiveness of the multiplicative Schwartz type block multi-color Gauss-Seidel smoother.

(3) Further code optimizations

We have continued investigations for the communication-computation overlapping, and applied it to parallel multigrid solver including forward/backward substitutions. Fig.4 shows the preliminary results on Odyssey using 128 and 512 nodes. Improvement of the performance is 15-20%。

Furthermore, we have also verified the performance impact of SIMD vectorization on the ODE part when running the simulator on A64FX processors. (This topic was not investigated in

FY2021.) Detailed time measurements show that, on average, compiler-enabled SIMD execution of the ODE part gives a speedup of 6.6x over a non-vectorized comparison baseline.

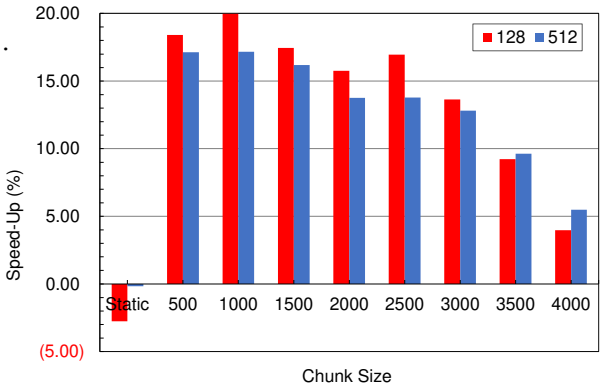

Fig.4 Effects of CC-Overlapping (Speed-Up) for Parallel MG Solvers using 128/512 nodes of Wisteria/Odyssey

(4) Porting to GPU clusters

Both versions of the simulators, including their ODE and PDE solvers, have been ported to the GPU architecture using OpenACC directives and Standard Parallelism (StdPar), rather than using dedicated languages such as CUDA. The sparse matrix array layout used in the PDE solver was not suitable for GPUs as is, so the array layout of the relevant part was changed so that the original AoS or GPU-friendly SoA can be selected at build time. Other than that, no major code changes were made. In the case of OpenACC, the OpenACC directives were added, and in the case of StdPar, the for loops were rewritten as std::for_each or std::transform_reduce. As shown in the Fig.5, both the OpenACC code and the StdPar code can achieve higher performance than the CPU on GPUs and good scalability. The OpenACC and StdPar codes can also be used on CPUs rather than exclusively on GPUs, and they achieve similar performance on CPUs compared to the original OpenMP code for CPU. The progress was presented as a joint paper at the domestic conference in December 2022 [8].

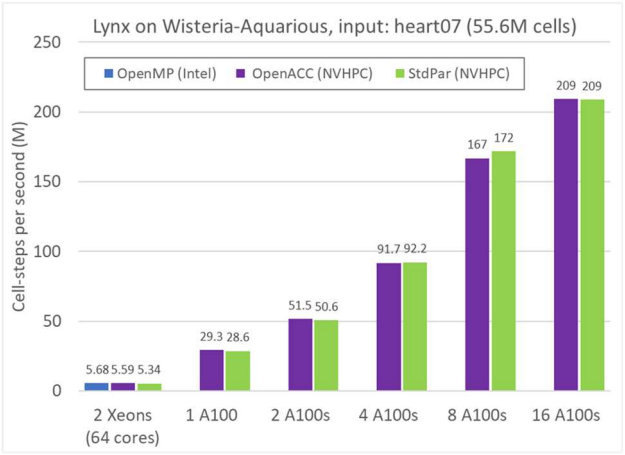

Fig.5 Results of porting of Lynx code to Aquarius with NVIDIA A100 GPUs [8]

(5) Large-scale simulations

The largest computational mesh used during FY2021 had 55 million tetrahedral computational cells. In order to fully test the parallel scalability of the simulator, as well as identifying potential performance hotspots, we have carried out simulations in FY2022 with much higher mesh resolutions. In the following, we will show a dissection of the time usage of the explicit simulator, as we change the number of compute nodes, MPI processes and OpenMP threads used. For these experiments, the ODE part and PDE part are solved equally frequently (without "inner" steps inside each "outer" time step). We focus on two unstructured meshes in this report, with 186 million and 1.5 billion tetrahedral cells, respectively. Both the Oakbridge-CX system and the Wisteria/Odyssey system were used. To respect the NUMA architecture within each node, we have always used two MPI processes per node and 28 OpenMP threads per MPI process on Oakbridge-CX. On Wisteria/Odyssey, the number of MPI processes adopted per node is 4, and the number of OpenMP threads per MPI process is 12.

Table 3. Time usage dissection of the explicit simulator on Oakbridge-CX for a global mesh of 186 million cells

#nodes	#cores	Total	PDE	ODE	MPI
1	56	577.6	205.41	368.55	3.66
2	112	298.1	105.43	186.65	6.01
4	224	158.8	54.15	95.19	9.48
8	448	79.5	28.39	47.02	4.09
16	896	41.1	15.06	23.37	2.64
32	1792	20.6	7.97	11.65	1.00
64	3584	10.9	4.04	5.86	0.96
128	7168	6.0	2.05	2.93	1.06
256	14336	3.3	0.98	1.48	0.83

Table 4. Time usage dissection of the explicit simulator on Wisteria/Odyssey for a global mesh of 186 million cells

#nodes	#cores	Total	PDE	ODE	MPI
8	384	116.1	25.04	88.07	3.02
16	768	60.2	13.61	44.03	2.59
32	1536	36.0	8.82	22.18	5.01
64	3072	19.9	4.82	11.07	3.98
128	6144	10.5	2.39	5.56	2.55
256	12288	5.2	1.17	2.74	1.33
512	24576	2.6	0.39	1.38	0.80
1024	49152	1.4	0.19	0.70	0.55
2048	98304	1.0	0.09	0.36	0.59

Table 5. Time usage dissection of the explicit simulator on Oakbridge-CX for a global mesh of 1.5 billion cells

#nodes	#cores	Total	PDE	ODE	MPI
16	896	308.7	110.16	184.01	14.50
32	1792	160.8	57.32	92.40	11.09
64	3584	83.8	30.19	46.30	7.32
128	7168	46.1	15.94	23.24	6.94
256	14336	34.6	7.99	11.67	14.90

It can be seen from the above four tables that the explicit simulator in general scales well with respect to the amount of CPU cores used. Concerning the computing speed, Oakbridge-CX

Table 6. Time usage dissection of the explicit simulator on Wisteria/Odyssey for a global mesh of 1.5 billion cells

#nodes	#cores	Total	PDE	ODE	MPI
128	6144	77.3	14.41	43.99	18.93
256	12288	38.6	7.43	22.01	9.21
512	24576	19.7	3.74	11.03	4.91
1024	49152	10.0	1.73	5.52	2.76
2048	98304	5.0	0.80	2.72	1.46

has an upper hand over Wisteria/Odyssey, especially for the ODE part. The time dissection shows that the only part that does not scale is the time usage due to point-to-point MPI communication (which is needed after each SpMV of the PDE step). We will show a deep-dive below by looking closer at the particular case of using 128 nodes (512 MPI processes in total) on Wisteria/Odyssey for the mesh of 186 million cells.

For the particular case of using 128 nodes of Wisteria/Odyssey for running the explicit simulator using 512 MPI processes on the 186-million-cell mesh, we have namely compared in detail the actual MPI time usage per process with a predictive cost model[1] of point-to-point MPI communication. In Fig. 6, due to space limit, we have only shown the slowest process time per node. There are two observations. First, the cost per process (per node) differs considerably, meaning that some MPI processes will finish way ahead of the others. Such an imbalanced situation of communication is not beneficial for performance, partially explaining why the MPI cost does not scale with respect to the amount of nodes used, as shown in the preceding four tables. A remedy is to try mapping the subdomains more appropriately to

[1] A. Thune, S. Reinemo, T. Skeie and X. Cai. *Detailed Modeling of Heterogeneous and Contention-Constrained Point-to-Point MPI Communication*. IEEE Transactions on Parallel and Distributed Systems. vol 34, pp. 1580-1593, 2023.

the hardware, with the hope that a more balanced situation of communication can be achieved. The second observation is that the prediction model of point-to-point communication cost has relatively good accuracy (with an overall error at 18.1% for this particular case). This cost prediction model will be used in our future work to further identify communication performance hotspots.

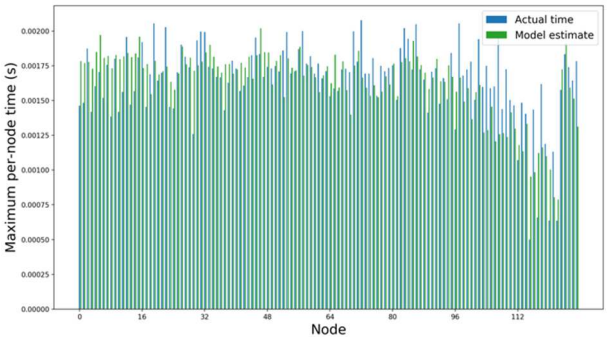

Fig.6 Comparing the actual and model-estimated MPI time usage per node on Wisteria/Odyssey

6. Self-review of Current Progress and Future Prospects

Compared with the project plan made at the beginning of FY2022, we consider that the research activities have been carried out accordingly during FY2022.

It is noted that some of the observations from our research activities of FY2022 require additional investigations in our future work. One such example is a thorough investigation of the potential performance benefit of applying advanced preconditioner to the sparse linear system solver in each PDE step of the implicit simulator. Another example is a closer investigation of the MPI overhead, especially in connection with better partitioning and mapping strategies.

7. List of Publications and Presentations

(1) Journal Papers (Refereed)

[1] X. Shen (+), J. van den Brink (+), A. Berg-Dahl (+), T. R. Kolstad (+), E. S. Norden (+), Y. Hou (+), M. Laasma (+), Y. Aguilar-Sanchez (+), A. P. Quick (+), E. K.S. Espe (+), I. Sjaastad (+), X. H.T. Wehrens (+), A. G. Edwards (+), C. Soeller (+), W. E. Louch (+) (2022) Prolonged β-adrenergic stimulation disperses ryanodine receptor clusters in cardiomyocytes and has implications for heart failure. **eLife 11:e77725**. https://doi.org/10.7554/eLife.77725

[2] M. H. Mesa (+), J. van den Brink (+), W. E. Louch (+), K. J. McCabe (+), P. Rangamani (+) (2022) Nanoscale organization of ryanodine receptor distribution and phosphorylation pattern determines the dynamics of calcium sparks. **PLOS Computational Biology18(6):e1010126**. https://doi.org/10.1371/journal.pcbi.1010126

[3] K. G. Hustad (+), X. Cai (+) (2022) Resource-efficient use of modern processor architectures for numerically solving cardiac ionic cell models. **Frontiers in Physiology 13:904648**. https://doi.org/10.3389/fphys.2022.904648

(2) Proceedings of International Conference Papers (Refereed)

(3) Presentations at International conference (Non-refereed)

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

[4] 中島研吾，通信と計算のオーバーラップによる前処理付き並列反復法，第 27 回計算工学会講演会，2022 年 6 月

[5] 中島研吾，Wisteria/BDEC-01（Odyssey）における前処理付き反復法の高速化，2022 年並列／分散／協調処理に関する『下関』サマー・ワークショップ，日本応用数理学会「行列・固有値問題の解法とその応用」研究部会（MEPA），2022 年 7 月

[6] 中島研吾，通信・計算オーバーラップによる並列多重格子法，日本応用数理学会年会 2022，2022 年 9 月

[7] 中島研吾，通信・計算オーバーラップによる並列多重格子法，情報処理学会研究報告 2022-HPC-187（13），2022

[8] A. Naruse, J.D. Trotter (+), J. Langguth (+), X. Cai (+), K. Nakajima, High resolution simulation of cardiac electrophysiology on realistic whole-heart geometries on Wisteria/BDEC-01 (Aquarius)，情報処理学会研究報告 2022-HPC-187（15），2022

(5) Published open software library and so on.

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