jh230018

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

Kengo Nakajima（The University of Tokyo）

Abstract
To further advance the scientific understanding of the intricate electrical activities in the heart, which are of vital importance for the human body, more elaborate mathematical modeling is needed with the support of supercomputing. In this project, we aim to implement and optimize a new simulator of cardiac electrophysiology, based on the latest cell-resolved modeling approach. The expected level of physiological details and the resulting computational resolution can only be achieved by an effective use of modern supercomputers. By combining efficient numerical algorithms with hardware-compatible software implementations, we want to enable computational scientists to carry out novel "in-silico" experiments. 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. During FY2023, this international JHPCN project carried out research on the following topics: (1) Software optimization of various components of the cell-resolved simulator; (2) Algorithmic developments needed for the parallelization process; (3) Some simulation studies of the cell-resolved simulator for comparison with other state-of-the-art cardiac simulators. The achieved results are expected to pave the way for completing an optimized cell-resolved simulator based on distributed-memory parallelization in FY2024, which will also be accelerated by GPU computing.

1. Basic Information

(1) Collaborating JHPCN Centers

Tokyo, Nagoya

(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

(4) Project Members and Their Roles

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

Xing Cai (Simula/Norway): Administration, numerical algorithms

Akihiro Ida (JAMSTEC): Numerical alg.

Toshihiro Hanawa (U Tokyo): Optimization

Akira Naruse (NVIDIA): Optimization

Masatoshi Kawai (Nagoya U): Numerical alg.

Tetsuya Hoshino (Nagoya U): Optimization

Yohei Miki (U Tokyo): Optimization

Kazuya Yamazaki (U Tokyo): Optimization

Masaharu Matsumoto (Fukushima Univ): Numerical alg.

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

Johannes Langguth (Simula/Norway): Optimization

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

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

An emerging paradigm in the modeling of cardiac electrophysiology is to fully resolve the individual cardiac cells. The exceptional level of detail in such models allows in-silico experiments to investigate the importance of factors such as the shape of the individual cardiac cells, the distribution of the subcellular structures, and the density and location of individual ion channels across the membranes.

This project aims to implement and optimize a new simulator of cardiac electrophysiology, based on the latest cell-resolved modeling approach. The purpose is to enable computational scientists to carry out novel in-silico experiments that will only become possible through a combination of efficient numerical algorithms with hardware-compatible software implementations.

The original plan for FY2023 has five main aspects. 1. Development of an MPI-parallel, cell-resolved simulator. 2. Partitioning strategies for the cell-resolved simulator. 3. Optimization of numerical methods for intra- and extracellular solvers. 4. Porting to GPU platforms. 5. Preliminary cell-resolved simulations of cardiac electrophysiology.

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 (now retired) and the Wisteria/BDEC-01 system (Odyssey & Aquarius), have been 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 FY2022 (Only for continuous projects)

Although this project is a not a direct continuation of project jh220041, because the adopted approach to modeling cardiac electrophysiology has been dramatically improved from the traditional monodomain model to a cell-based model, nevertheless, the research results of project jh220041 are relevant (with some of them being directly reusable) and thus worth a brief mentioning. During FY2022, investigations were carried out in the following topics. 1. A second simulator of the monodomain model 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 monodomain simulator based on implicit time integration which requires solving a sparse linear system per time step. 3. 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 monodomain 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.

5. Details of FY2023 Research Achievements

Before describing the research results achieved in FY2023, let us first briefly introduce the cell-based modeling of cardiac electrophysiology. In short, this new modeling approach fully resolves the individual cardiac cells (termed *intracellular domains*), the surrounding area in-between the individual cells (termed *extracellular domain*), and the cell membranes that are the interfaces between the intracellular and extracellular domains. (In a traditional model of cardiac electrophysiology, such as the monodomain model, the intracellular, extracellular and membrane domains coexist everywhere throughout the entire heart, which constitutes a major simplification of the physiology.)

Inside each intracellular domain (i.e., each cardiac cell), an individual Laplace equation applies, whereas in the entire extracellular domain a global Laplace equation applies. The coupling between the intracellular and extracellular domains is through boundary conditions applicable on the cell membranes, whose dynamics are described by a set of nonlinear ordinary differential equations. The mathematical model is summarized as follows:

$$
\begin{aligned}
\nabla\cdot\sigma_i\nabla u_i^k &= 0 && \text{in } \Omega_i^k,\\
\nabla\cdot\sigma_e\nabla u_e &= 0 && \text{in } \Omega_e,\\
C_m\frac{\partial v^k}{\partial t} + I_{\text{ion}}^k(v^k,s^k) = n_e\cdot\sigma_e\nabla u_e &= -n_i^k\cdot\sigma_i\nabla u_i^k && \text{at } \Gamma_k,\\
v^k &= u_i^k - u_e && \text{at } \Gamma_k,\\
\frac{\partial s^k}{\partial t} &= F_k(s^k,v^k) && \text{at } \Gamma_k,\\
C_g\frac{\partial w^k}{\partial t} + I_{\text{gap}}^{k,j}(w^k) = n_i^j\cdot\sigma_i\nabla u_i^j &= -n_i^k\cdot\sigma_i\nabla u_i^k && \text{at } \Gamma_{k,j},\\
w^k &= u_i^k - u_i^j && \text{at } \Gamma_{k,j},
\end{aligned}
$$

(1) A preliminary parallel implementation of a cell-resolved simulator

Since there are three types of domains: intracellular, extracellular and membranes, any parallelization needs to parallelize the computations on each of the three domain types. We have here adopted an operator-splitting strategy, which in addition to decoupling the membrane ODEs from the PDEs, also allows separate solutions of the individual intracellular Laplace equations and the global extracellular Laplace equation. The numerical coupling between the intracellular and extracellular domains is realized by an iterative procedure with appropriate boundary conditions. For all the Laplace equations, finite element discretization with linear tetrahedral elements is used.

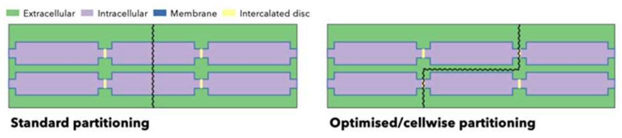
We have developed a preliminary MPI parallel implementation that is based on separately partitioning the three domain types (with the help of three independent mesh partitioners). The global extracellular Laplace equation is solved in parallel by all the MPI processes that collaboratively run a parallel conjugate gradient (CG) solver with a parallel AMG preconditioner. The individual intracellular Laplace equations can in principle be concurrently solved by the different MPI processes each using, e.g., a serial preconditioned CG solver. However, in the preliminary parallel implementation, the mesh partitioner of the intracellular domains cannot guarantee to delegate whole intracellular domains to the MPI processes. Therefore, some of the intracellular Laplace equations may require several MPI processes to collaboratively carry out a parallel solver, which incurs unnecessary inter-process communication and synchronization. Regarding the membrane computation (solving the ODE system per membrane mesh point), the execution is embarrassingly parallel, where the membrane mesh points are equally distributed among the MPI processes. Another major weakness of the preliminary parallel cell-resolved simulator is due to the three independent mesh partitioners, which may incur extensive inter-process communication to enable the numerical coupling between the intracellular, extracellular and membrane domains.

MPI				Intracellular		Extracellular	
procs	cells	Total	Membrane	Assembly	Solve	Assembly	Solve
1	4x4	15.69	5.02	0.063	7.41	0.015	2.887
4	8x8	30.65	5.09	0.064	19.47	0.015	5.691
16	16x16	90.96	5.37	0.073	72.60	0.018	12.55
64	32x32	438.41	5.53	0.103	385.65	0.024	46.60

The above table dissects the time usage of the three domain types of the preliminary version of the parallel cell-resolved simulator. The individual cells are configured as a 2D lattice where the number of cells grows linearly with the number of number of MPI processes (i.e., a weak scaling test). It can be seen that the membrane computation scales very well (almost constant time usage with the increasing MPI processes). The extracellular part scales less efficiently, due to the growing number of CG iterations with the number of processes, as well as the communication overhead. The intracellular part suffers severely due to the unnecessary inter-process communication and synchronization. This thus calls for more appropriate mesh partitioning (to be discussed in the following topic).

(2) **Improving mesh partitioning for the parallel cell-resolver simulator**

As mentioned above, the preliminary parallel implementation uses three independent mesh partitioners to divide the work on the intracellular, extracellular and membrane domains. The three resulting sets of mesh partitioning results will thus not align with each other. There can arise extensive inter-process communication when enforcing the numerical coupling between the three domains. Regarding the intracellular part, the adopted mesh partitioner cannot guarantee that whole cardiac cells are delegated to the different MPI processes. That is, some of the cardiac cells may be divided between several processes (as shown in the left plot of the figure below), thus leading to unnecessary inter-process communication and synchronization.



Therefore, it is important to achieve the partitioning results of the three domains that, to a much larger extent, align with each other. Currently, we are working on a new partitioning strategy where the tetrahedral elements of the extracellular domain and the tetrahedral elements of all the intracellular domains are decomposed by a “one-go” partitioner. The main idea is to formulate a graph that only contains the extracellular and intracellular elements, where a multi-objective graph partitioner will produce the partitioning results of the extracellular domain and the intracellular domains in “one go”. In order to not let the partitioning cut through any of the intracellular domains, we are experimenting with clustering the elements of each intracellular domain into one special graph node. A desirable partitioning result is depicted in the right plot of the figure above. We note that the partitioning results of the intracellular and extracellular domains also determine a partitioning of the membrane mesh points, which will incur minimum inter-process communication when enforcing the numerical coupling between the membrane domain and the other two domains. We will continue with this research topic in FY2024.

(3) Further optimizations of different parts in the intracellular and extracellular solvers

We have carried out optimizations of various components needed in the intracellular and extracellular solvers. These are detailed below.

(3a) ***Using sector cache to speedup SpMVs***

Sparse matrix-vector multiplication (SpMV) is a fundamental operation in sparse linear algebra. For example, each CG iteration involves one SpMV. The performance of SpMV heavily depends on data reuse in the caches. When the sparse matrix is stored in, e.g., the standard compress sparse row format, the only data of an SpMV operation that will benefit from data reuse is the input vector. Therefore, a reserved part of a cache only for the input vector can prevent competition from the other non-reusable data, thus improving the SpMV performance, in comparison with letting all data equally compete for the entire cache.

The *sector cache* technique which is available on the A64FX processor serves the goal as described above. The A64FX's sector cache uses a way-based cache partitioning mechanism to redirect some of a program's data to separate parts (i.e., sectors) of the L1 or L2 caches. The purpose is to avoid cache pollution or thrashing that occurs when reusable and non-reusable data are mixed. Assigning reusable data to its own sector may thus improve cache reuse, which is a primary concern in memory-bound kernels such as SpMV.

The code snippet below shows how to configure the A64FX's sector cache for an SpMV kernel with a matrix in the compressed sparse row format:

```
1 #pragma procedure scache_isolate_way L2=M L1=N
2 #pragma procedure scache_isolate_assign a
3 #pragma procedure scache_isolate_assign colidx
4 #pragma omp for
5 for (int r=0; r<num_rows; r++)
6   for (int i=rowptr[r]; i<rowptr[r+1]; i++)
7     y[r] += a[i]*x[colidx[i]];
```

Line 1 allocates *M* cache ways to sector 1 in the L2 cache and *N* ways to sector 1 in the L1 cache, while the remaining cache ways default to sector 0. Lines 2–3 assign the arrays `a` and `colidx` to sector 1, whereas the other data defaults to sector 0. Specifically, values of `a` and `colidx` are used only once during a single SpMV operation, they are thus typically read entirely from memory rather than cache. The other data, however, such as the vectors stored in the `x` and `y` arrays, are frequently reused from cache. It is thus desirable to isolate such reusable data in a separate sector.

For the commonly encountered scenario where the SpMV operations are performed repeatedly, the following classification of sparse matrices, based on their input dimensions, can be used to understand whether or not using the sector cache is beneficial:

(1) The matrix and vectors together fit into cache.

(2) The matrix and vectors together do not fit into cache, but `x`, `y` and `rowptr` together fit into a cache partition.

(3) `x`, `y` and `rowptr` together do not fit into a cache partition, while either

(a) `x` completely fits into a cache partition, or

(b) `x` does not fit into a cache partition.

Matrices in class (1) are expected to not benefit from the sector cache, since there are no capacity misses in this case. Matrices from class (2), on the other hand, are expected to benefit most from the sector cache, because cache misses caused by accesses to `x`, `rowptr`, and `y` are avoided. Finally, when the matrix dimensions are large enough such that `x`, `rowptr`, and `y` together do not fit into cache, i.e., case (3), only cache misses due to accesses to `x` can be avoided by using the sector cache. If `x` fits completely into its own cache partition, then it incurs no capacity misses. Otherwise, isolating `x` lowers the reuse distance of references to `x`, potentially avoiding some cache misses.

Different sector cache configurations were evaluated, and it was found that assigning 11 or 12 (out of 16) L2 cache ways for reusable data yields the best SpMV performance across a large collection of 490 sparse matrices. At the same time, enabling the L1 sector cache degrades performance due to overly aggressive hardware prefetching and premature eviction of non-reusable data.

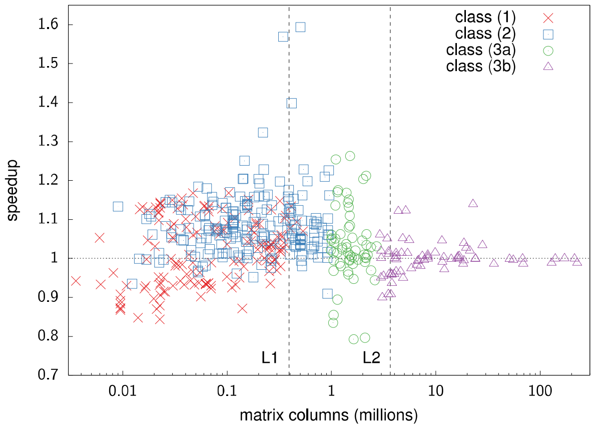


Fig.1 Speedup versus matrix columns (vector size) for SpMV on A64FX using sector cache with 5 L2 ways.

Fig. 1 shows the size of the x-vector versus speedup after assigning 11 ways in L2 cache for reusable data. As expected, the sector cache almost always improves performance for matrices in class (2). The improvement can be up to 1.6x, but it is usually no more than 20%. Some matrices in class (3a) and (3b) show similar speedup, though the impact of the sector cache gradually decreases as matrix dimensions increase.

We also found the speedup to coincide with a reduction in L2 demand cache misses, as shown in Fig. 2. In particular, matrices experiencing the highest speedup also observed a corresponding reduction in L2 demand misses. On the other hand, some matrices have a significant reduction in L2 demand misses, but with little or no speedup. It is possible that the L1 or L2 caches are the main bottlenecks instead of memory latency or bandwidth in these cases.

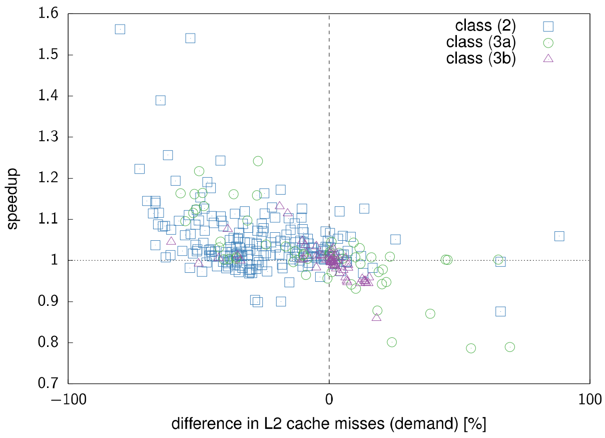


Fig.2 Speedup versus difference in L2 demand cache misses for SpMV on A64FX using sector cache with 5 ways in L2.

(3b) ***Experimenting with different linear solvers for the intracellular Laplace equation***

An ideal situation for the parallel cell-resolved simulator is that whole intracellular domains are delegated to different MPI processes, so that each MPI process can concurrently solve one or more intracellular Laplace equations, without collaboration from the other processes. It is thus important to achieve good performance when solving an intracellular Laplace equation with the help of OpenMP threads. We have studied four linear solvers: CG with a Gauss-Seidel preconditioner (from the MFEM library, only capable of using one thread), CG preconditioned with AMG (from the hypre library), a single-threaded direct solver from the UMPACK library and a direct solver from the SuperLU library. The following table shows the time usage (depending on the number of OpenMP threads) when solving a Laplace equation with 453,572 unknowns.

Threads	CG-Gauss-Seidel	CG-AMG (hypre)	UMFPACK	SuperLU
1	127.90	21.26	15.69	25.10
2	–	12.34	–	18.77
4	–	7.60	–	15.71
8	–	6.39	–	15.82
16	–	5.04	–	16.43
24	–	5.23	–	17.37

It can be seen that CG+AMG achieves the fastest computing time for this particular example.

We have been also developing our own AMG solver. In FY.2023, we have implemented massive parallelization and also applied the Multiplicative Schwartz Type Block Multi-Color Gauss-Seidel (MS-BMC-GS) smoother and confirmed its effectiveness. MS-BMC-GS is a smoother suitable for massively parallelization and is a method to improve convergence by increasing operations that guarantee high localities. The evaluation results confirmed an additional 10-20% improvement in convergence from the Block Multi-Color Gauss-Seidel smoother. The parallelization itself is not finished, and it will continue next year.

(3c) ***Communication-Computation Overlapping in Parallel Multigrid Methods***

Preconditioned iterative methods based on the Krylov subspace technique are widely employed in various scientific and technical computing. When utilizing large-scale parallel computing systems, the communication overhead tends to increase with the growth in the number of nodes, making its reduction a crucial challenge. In parallel finite element methods (FEM) and finite volume methods (FVM), halo communication and computation overlapping (CC-Overlapping) are commonly employed, often in conjunction with the dynamic loop scheduling feature of OpenMP. This approach has been primarily applied to sparse matrix-vector products (SpMV) and explicit solvers. Previous studies by the author have proposed reordering techniques for applying CC-Overlapping to processes involving global data dependencies, such as the Conjugate Gradient method preconditioned by Incomplete Cholesky Factorization (ICCG). Successful implementations on massively parallel supercomputers demonstrated high parallel performance, but the application of CC-Overlapping was limited to SpMV. In [5], we proposed a method to apply CC-Overlapping to the forward and backward substitutions of the IC(0) smoother of the parallel Conjugate Gradient method preconditioned by Multigrid (MGCG). Using up to 4,096 nodes on Wisteria/BDEC-01 (Odyssey) with A64FX, performance improvement of approximately 40+% was achieved compared to the original implementation, while improvement was 20+% on 1,024 nodes of Oakbridge-CX system with Intel Xeon CPU's.

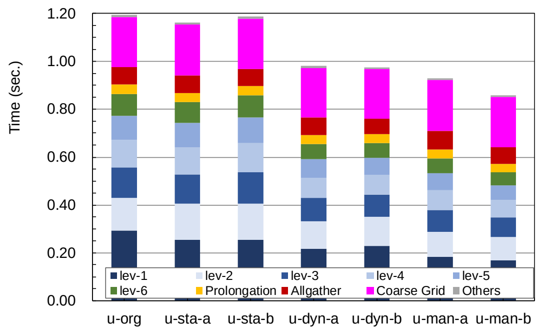
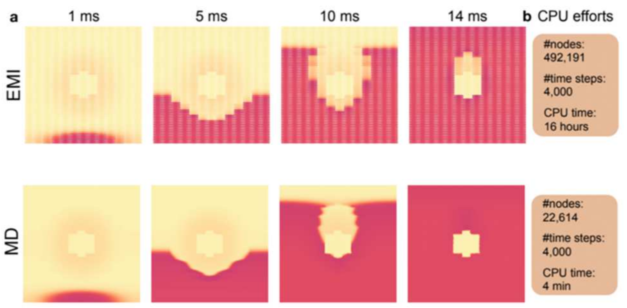


Fig.3. Computation Time for MGCG Solver on ODY (Ultra Tiny (u): 128×64×64 meshes/MPI Process, Detailed Breakdown for Each Procedure in MGCG Solver (4,096 nodes) [5]

(4) Experimenting with cell-resolved simulations

At the same time as improving the preliminary implementation of the cell-resolved simulator, we have also carried out some comparisons between the cell-resolved simulator and the traditional monodomain simulator. In the following figure, **EMI** (extracellular-intracellular-membrane) denotes the cell-resolved simulator, whereas **MD** stands for the monodomain simulator. This is an example experiment of a small infarction scar in the heart. Here, we consider a collection of 13×65 cells, and the plots are simulation snapshots from different time points. It has been confirmed by domain scientists that the cell-resolved simulation is physiologically more accurate than the monodomain simulation.



6. Self-review of Current Progress and Future Prospects

Compared with the project plan made at the beginning of FY2023, some of the research topics have not progressed as far as originally expected. Specifically, the GPU acceleration of the cell-resolved simulator has been postponed, because we feel that a solid MPI version must be completed first. Currently, intensive research effort is being put on devising and implementing the “one-go” mesh partitioner which is vital for achieving good parallel MPI performance (by minimizing inter-process communication and eliminating unnecessary inter-process synchronization). Once the “one-go” mesh partitioner is in place, we envision to be able to carry out large-scale cell-resolved simulations, which will greatly exceed the current state of the art. Following that, GPU acceleration will be investigated so that even larger scales of cell-resolved simulations will be attempted to further advance the research field of cardiac electrophysiology.

7. List of Publications and Presentations

(1) Journal Papers (Refereed)

[1] Ryo Yoda, Matthias Bolten, Kengo Nakajima, Akihiro Fujii, Coarse-grid operator optimization in multigrid reduction in time for time-dependent Stokes and Oseen problems, Japan Journal of Industrial and Applied Mathematics (in press), 2024

[2] Akihiro Fujii, Teruo Tanaka, Kengo Nakajima, Light Weight Coarse Grid Aggregation for Smoothed Aggregation Algebraic Multigrid Solver, IEEE Access (in press), 2024

[3] Yen-Chen Chen, Kengo Nakajima, A Cascadic Parareal Method for Parallel-in-Time Simulation of Compressible Supersonic Flow, IPJS Transaction on Advanced Computing Systems (in press), 2024

(2) Proceedings of International Conference Papers (Refereed)

[4] Sergej Breiter, James Trotter (+), Karl Fürlinger. Modelling Data Locality of Sparse Matrix-Vector Multiplication on the A64FX. Proceedings of the SC23 workshops, 2023. https://doi.org/10.1145/3624062.3624198

[5] Kengo Nakajima, Communication-Computation Overlapping for Parallel Multigrid Methods, IEEE Proceedings of iWAPT 2024 in conjunction with IPDPS 2024 (in press) , 2024

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

[6] Kengo Nakajima, Communication-Computation Overlapping in Parallel Multigrid Methods, 21st Copper Mountain Multigrid Conference, 2023

[7] Xing Cai (+), James Trotter (+), Enabling Cell-Based Simulations of Cardiac Electrophysiology with High-Performance Computing. SIAM PP24, 2024.

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

[8] 中島研吾，並列多重格子法における通信・計算オーバーラップの最適化，2023 年並列／分散／協調処理に関する 『函館』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），2023 年 8 月 3 日　北海道函館市

(5) Published open software library and so on.

(6) Other (patents, press releases, books and s