Application of Block-structured Adaptive Mesh Refinement to Particle Simulation

Usui, Hideyuki
Kito, Saki
Nunami, Masanori
Matsumoto, Masaharu

(Citation)
Procedia Computer Science,108:2527-2536

(Issue Date)
2017

(Resource Type)
journal article

(Version)
Version of Record

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 108C (2017) 2527–2536

Procedia Computer Science

International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

Application of Block-structured Adaptive Mesh Refinement to Particle Simulation

Hideyuki Usui[1], Saki Kito[1], Masanori Nunami[2], and Masaharu Matsumoto[3]

[1] Graduate School of System Informatics, Kobe University, Kobe, Hyogo, Japan
h-usui@port.kobe-u.ac.jp
[2] National Institute for Fusion Science, Tajimi, Gifu, Japan
[3] University of Tokyo, Tokyo, Japan

Abstract

We implemented the particle treatment in the block-structured adaptive mesh refinement (AMR) framework which we have developed. In the AMR framework, the simulation domain is divided into multiple sub-domains and they are assigned to a number of processes for parallel calculation using MPI. A sub-domain is composed of multiple block-structured regions each of which has the fixed number of grids. When high resolution is required at a certain region in the sub-domain, a block-structured region with refined grids, which is called child block, is locally created. To apply this AMR framework to the particle simulations such as particle-in-cell simulation, we set up several arrays for the particle treatment for each sub-domain assigned to one process. These arrays are shared among all the blocks consisting of the corresponding sub-domain. For the particle calculation in each block, we also set up several other arrays which are privately defined and used in each block. The functions for these arrays for the particle treatment are described in this paper. To test the implementation of the particle treatment in the AMR framework, we performed test simulations by adopting the sugarscape model which was proposed for the simulation of an artificial society by using many agents representing inhabitants in a certain area. We treated the inhabitants as a bunch of particles and assign the sugar amount at each grid as the environment in a two-dimensional simulation domain. In the simulation, we initially place two peaks of sugar and randomly distribute the inhabitants. For simplicity, we examined the motions of the inhabitants agents by assuming no temporal variation of the sugar. The simulation results show that the inhabitant agents were accelerated and gathered to the two sugar peaks as we expected. When the inhabitant density exceeded a certain criteria, child blocks with refined grids were adaptively created at the corresponding region. We confirmed that the motions of the inhabitants were not influenced by the block boundaries, which ensures that the particle treatment to the AMR framework has been properly implemented.

Peer-review under responsibility of the scientific committee of the International Conference on Computational Science

Keywords: Block-structured adaptive mesh refinement, particle simulation, domain decomposition, multi-agent simulation, sugarscape

1877-0509 © 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the scientific committee of the International Conference on Computational Science
10.1016/j.procs.2017.05.255

1 Introduction

The adaptive mesh refinement (AMR) method (e.g. [2]) has been used in various fields of science and engineering (e.g. [4], [8]) . The AMR technique enables us to perform efficient numerical calculation because it can adaptively create fine cells regions where higher resolution is required. However, it is not easy to implement the AMR technique in conventional simulation codes which use uniform grid size. The most difficult part in the AMR implementation is the management of the hierarchical layers with different grid sizes which have to be created or removed adaptively in a simulation run depending on the requirement of local high resolution. For those who are interested in adopting the AMR technique to their own simulation programs, several AMR frameworks have been developed. One of the famous frameworks is called PARAMESH [7] which has been used in several fields.

Meanwhile, we developed a new plasma particle-in-cell (PIC) code using AMR called PARMER [10]. PARMER incorporates cell-based AMR technique with which we adaptively generate hierarchical layers with different grid sizes at the local regions where high resolution is needed. However, as mentioned above, the AMR technique used in PARMER is not easy to be implemented to other uniform-grid simulation codes. To overcome this problem, we developed an AMR framework with which the AMR technique can be relatively easily ported to generic simulation programs which hire the uniform grid system [9]. In our AMR framework, we applied the AMR technique to a block-structured region consisting of the fixed number of grids. A generic simulation program written either in the C or Fortran language using uniform grid size can be implemented in each block in the AMR framework. Once a situation occurs where high resolution is needed in a local region, the corresponding block-structured region is divided into eight for three-dimensional case and new block-structured regions with uniform grid with half size of the original one are generated. The block-structured regions are managed in a fully threaded tree (FTT) data structure which allows recursive refinement on a block-by-block basis [6]. The detail of the numerical process of this grid refinement has been described in [9]. By applying this AMR framework to magnetohydrodynamics(MHD) simulation [5], we could successfully simulate the fine structure of Kelvin-Helmholtz instability in plasma by using hierarchical levels of different grid sizes.

For the application of the AMR framework, we implemented the particle treatment in the AMR framework so that the AMR simulations can be easily realized even for the particle model such as PIC model widely used in the plasma particle simulation. In Section 2, we describe the detail of the implementation of the particle treatment to the AMR framework. To test the particle implementation, we applied the particle-ported AMR framework to the sugarscape model proposed for the simulation of an artificial society by using many agents representing inhabitants in a certain area [3]. In the sugarscape simulation, sugar amount is distributed at each grid point in a two-dimensional space. On top of this plane, inhabitants, which are represented by many agents, consume the sugar assigned at each grid point at each time step. For the test of the particle-ported AMR framework, we treat the inhabitants as a large number of particles and simulated the response of the inhabitants to the sugar placed in the simulation domain. For simplicity, we examined the motions of the inhabitants agents by assuming no temporal variation of the sugar in the test simulation. The detail of the test simulation is described in Section 3.

2 Particle Treatment in the Block-structured AMR Framework

As stated above, we have developed the block-structured AMR framework by which AMR technique is relatively easily ported in a generic simulation program. The blocked-based AMR framework has a feature that a region which needs to be refined is limited to a block-structured one consisting of the fixed number of cells. Here we describe the particle implementation to the AMR framework. In this study, we take the PIC model which is used in the plasma simulation. In the PIC simulations [1], each particle has its position and velocity which are independent of spatial grids set up in the simulation domain. As time elapses, particles in the simulation domain can move to arbitrary regions. When we handle the particles in the AMR framework, we need to manage the particles which move across the block boundaries.

The following is the general idea for the particle treatment. In the block-structured AMR framework, we adopt domain decomposition for process parallelization using MPI. This implies that each sub-domain of the simulation space is assigned to each process. Figure 1(a) shows four sub-domains distributed to each process. For simplicity, a two-dimensional simulation space is considered. Each sub-domain consists of multiple block regions as shown in Panel (b). Each block region has its own spatial grids as well as particles which exist in the block region. The numerical treatment of the spatial grids are described in the previous study [9]. Regarding particles, we will first focus on the treatment in one sub-domain assigned to one process.

Figure 2 (a) shows an illustration of the array setup for the particle treatment in one process. Four arrays called PA, Plabel, Pbucket and PTbucket are defined in each process. They are commonly used among blocks composing the corresponding sub-domain. PA, which means particle array, is a multi-dimensional array which contains the data of position and velocity ($\boldsymbol{r}$,$\boldsymbol{v}$) of all the particles located in the sub-domain assigned to a process. The species number and index number of the each particle are also attached together with ($\boldsymbol{r}$,$\boldsymbol{v}$) in PA. Plabel contains the element numbers of PA in which particle data is no longer used because the corresponding particles are transferred to other processes. The particles coming into the corresponding process are stored at the empty elements of PA whose numbers are stored in Plabel. Arrays called Pbucket and PTbucket will be explained later.

In each block, another four arrays called PT, PTtrans, LPA, LNP are defined. The same set of arrays is defined in other blocks. PT, which is the particle table, is prepared for the reference

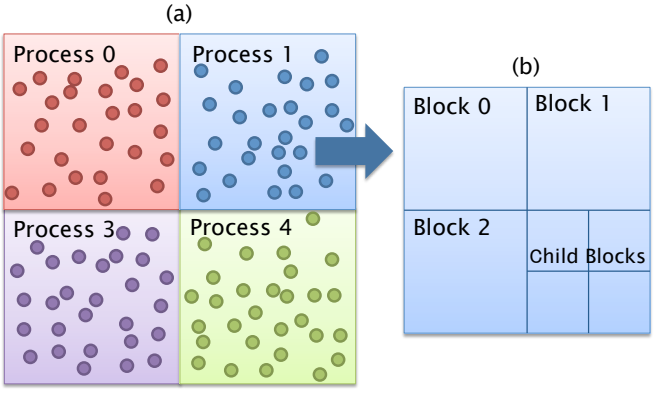

Figure 1: (a) Domain decomposition and (b) multiple blocks assigned in one sub-domain.

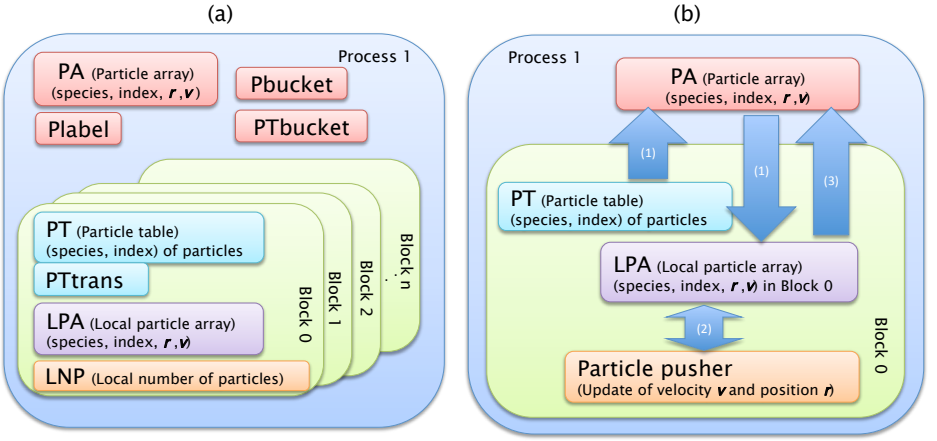

Figure 2: (a) Arrays prepared for the particle treatment in a sub-domain assigned to a process. (b) Procedure of the particle treatment in a process which take cares one sub-domain.

of particles located in the block. PTtrans contains the species number and the index number of the particles which newly enter into the block from the neighbor blocks. The information of PTtrans is added to PT. LPA, which is local particle array, contains the particle position and velocity ($\boldsymbol{r}$,$\boldsymbol{v}$) copied from PA according to the particle information in PT. LNP, which is local number of particles, is a counter variable which stores how many particles are located in the block for each species.

Figure 2(b) indicates how these arrays are used for the particle treatment in one sub-domain. As shown in Figure 2(a), all the particle data are stored and maintained in PA. The following basic steps are carried out for each block in a sub-domain.

1. The particle data including ($\boldsymbol{r}$,$\boldsymbol{v}$) are copied to LPA based on the information in PT.
2. The particles in LPA are updated by the particle pusher routine in each block.
3. The updated particle data ($\boldsymbol{r}$,$\boldsymbol{v}$) are copied back to PA.

When LPA is copied back to PA in the step (3), the positions of the updated particles are checked if they stay in the same block or sub-domain. In checking the particles, the treatment is different depending on the updated particle position. If there is a particle which moves to a different neighboring block, the particle index and the species number are inserted in the array called PTtrans owned by the corresponding neighboring block. After the particle check, the information of PTtrans will be appended to PT in each block and the updated PT is used in the step (1) stated above. If there is a particle which moves to a different sub-region handled by another process, the index and species number of the corresponding particle are stuck in the array called PTbucket which is shared in a process as shown in Figure 2(a). After the particle check, the particle data ($\boldsymbol{r}$,$\boldsymbol{v}$) corresponding to PTbucket are stored to Pbucket. We prepare two arrays of Pbucket for one direction. If we take the x direction for example, one Pbucket is used to store the particles which move across the boundary in the positive x direction and the other Pbucket is used for the negative x direction. Another two arrays of Pbucket are prepared for the y direction as well as the z direction.

Once the packing of the particle data to Pbucket is done, the data are copied to P-send array as shown in Figure 3. The contents of P-send are transferred to the adjacent sub-domains

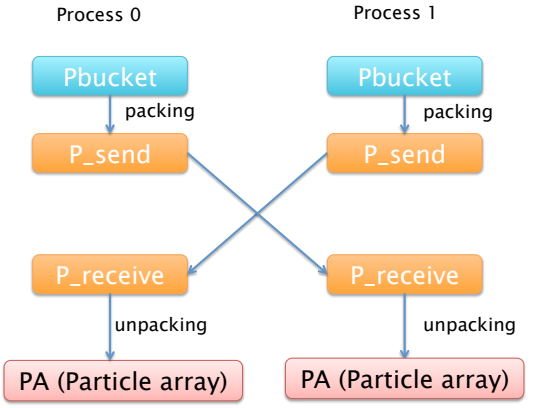

Figure 3: Particle data transfer between sub-domains using MPI.

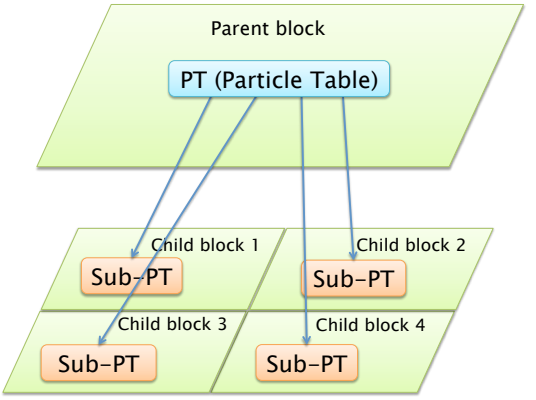

Figure 4: Division of the parent PT to the child PTs in child blocks.

and stored in the corresponding P-receive array. Exchange of the particle data with P-send and P-receive among multiple processes is carried out with MPI. The particle data stored in Pbucket are erased from PA. The erased element in PA is reused for the particles newly coming into the current sub-domain.

When high resolution is locally required, child blocks with refined grids are adaptively generated in the parent block as shown in Figure 1 (b). In such a situation, PT in the parent block is divided into the number of the child blocks and the corresponding part of PT in the parent block will be transferred to each sub-PT, which is schematically shown in Figure 4. Since all the particle data is stored in PA which is a common array in a sub-domain, the particle treatment for the child block is the same as that for the other blocks.

3 Application to Sugarscape Model

3.1 Particle-base sugarscape model (PSS model)

To test the particle-ported AMR framework described in Section 2, we adopted the sugarscape model simulation [3] which is one of the multi-agent simulations used for the study on the social science. As shown in Figure 5, the general agent-based simulation contains three components which are agents, environment, and their rules. Agents and environment evolve according to have their own rules. Meanwhile, they can interact each other and their states also changes in time through the interactions. In the sugarscape model the agents correspond to a bunch of inhabitants and the environment is represented by the amount of sugar assigned at each cell. The basic rule for the agents is that the inhabitants move to the region where the sugar amount is larger than that at the present position and consume the local sugar at a certain rate. When the sugar amount at the inhabitant position becomes zero, the corresponding inhabitants will die. The sugar amount decreases in time when the inhabitants keep consuming. The original sugarscape model is based on the cellular automaton in which the agents and the environment are both defined by cells. In such a model, the overlap of the agent positions is not permitted because one cell corresponds to an agent.

In this study, by using particles for inhabitant agents, we simulate the inhabitant motions which are independent of the spatial grids where the sugar amount is distributed. Hereafter, we

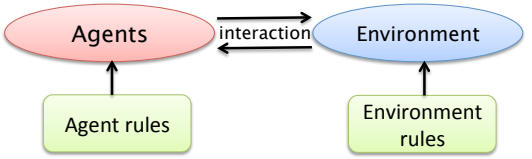

Figure 5: Concept of Agent-based simulation.

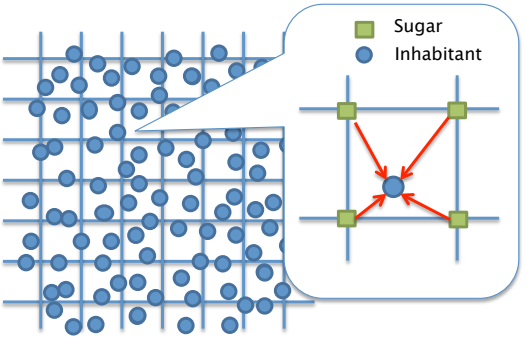

Figure 6: The distribution of inhabitant agents in the simulation domain and the relation between an inhabitant and the sugar amount assigned at grids in PSS.

call this version of sugarscape model "particle-based sugarscape (PSS)" model. Figure 6 shows the initial distribution of inhabitant agents in the simulation domain as well as the relation between an inhabitant and the sugar amount assigned at the adjacent grids. The inhabitants are represented by many particles indicated by small blue circles. One of the advantages in the PSS simulation is that multiple inhabitants can exist in one cell, which is different from the conventional model. The environment data which corresponds to the amount of sugar for the sugarscape model are assigned at each grid point.

In this test simulation, we focus on the inhabitants motions. Therefore, for simplicity, we assume no temporal variation of the sugar amount at each grid point. To trace the inhabitants trajectories, we solve the equations of motion for all the inhabitants by using the FDTD method. The followings are the equations of motion to be solved.

$$\frac{\partial \boldsymbol{x}}{\partial t} = \boldsymbol{v} \tag{1}$$

$$\frac{\partial \boldsymbol{v}}{\partial t} = \alpha \boldsymbol{F_s} \tag{2}$$

where $\boldsymbol{x}$ and $\boldsymbol{v}$ denote the position and velocity of each inhabitant, respectively. $\boldsymbol{F_s}$ and α indicate the spatial gradient of the sugar amount obtained at inhabitant position and its coefficient, respectively. In PSS, $\boldsymbol{F_s}$ is given at each grid point by the following equation.

$$\boldsymbol{F_s} = -\nabla S_{i,j} \tag{3}$$

where $S_{i,j}$ denotes the sugar amount at the grid point (i, j). However, we need $\boldsymbol{F_s}$ at the inhabitant position which is independent of the grid points. We obtain $\boldsymbol{F_s}$ at the inhabitant position by the interpolation using those defined the neighboring grid points with respect to the corresponding inhabitant as shown in the right panel of Figure 6.

The velocity and position of each inhabitant are updated at every time step by solving Equations (1) and (2). As shown in the above equations, the inhabitants obtain large acceleration when $\boldsymbol{F_s}$ is large at their positions.

3.2 Implementation of PSS model to the Particle-ported AMR Framework

To test the function of the particle-ported AMR framework, we implemented the PSS model simulation to the framework and performed a test simulation. The followings are the parameters used for the test simulation. The simulation domain consists of 4×4 base blocks. Since we assigned one process to each block, we used 16 processes for the simulation. The particle transfer between the processes were conducted with the methods described in the previous section by using MPI. A base block consists of 128×128 grid points and the total number of the grid points in the simulation domain is 512×512. We set the grid size $dr = 1.0$ for the base blocks which correspond to the Level-0 layer of the AMR hierarchy. We distributed 100,000 inhabitants randomly in the simulation domain. We set that the criteria for the grid refinement is the number density of the inhabitants n at each grid. When n exceeds a certain value, the block which contains the corresponding grid point creates the child blocks which have the half grid size of that used in the parent block [9]. In the test simulation, the criteria for the generation of Level-1 block is $2n_0$ where n_0 denotes the average of the initial inhabitant density in the simulation domain. When n exceeds $2n_0$, the grid refinement is conducted and the grid size becomes $0.5dr$ for the child blocks. Level-2 child blocks are also created when n exceeds $3n_0$. As for the environment, we distributed sugar amount at each grid point in the simulation domain. We set two peaks of the sugar amount in the simulation domain. The initial distribution of the sugar amount and the inhabitants is plotted in the xy plane as shown in Figure 7. For simplicity, we assumed no temporal variation of the sugar in the test simulation. A bunch of small dots shown in the simulation domain indicate the representative inhabitants. The color map shows the distribution of the sugar amount. As shown in the map two mountains of sugar amount are set in the simulation domain. The sugar amount is normalized to an arbitrary value in this simulation.

Figure 8 shows spatial profiles of n in color maps at different time steps. The density values are normalized to n_0. Panels (a) and (b) show the results obtained in the PSS model simulation using the particle-ported AMR framework. White lines in the both panels show the block boundaries. As shown in Panels (a) and (b), n became higher in the regions of the two

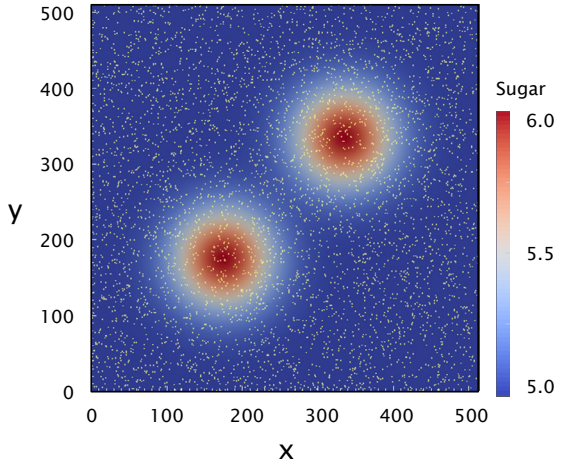

Figure 7: Initial distribution of sugar amount in color map and inhabitants with dots.

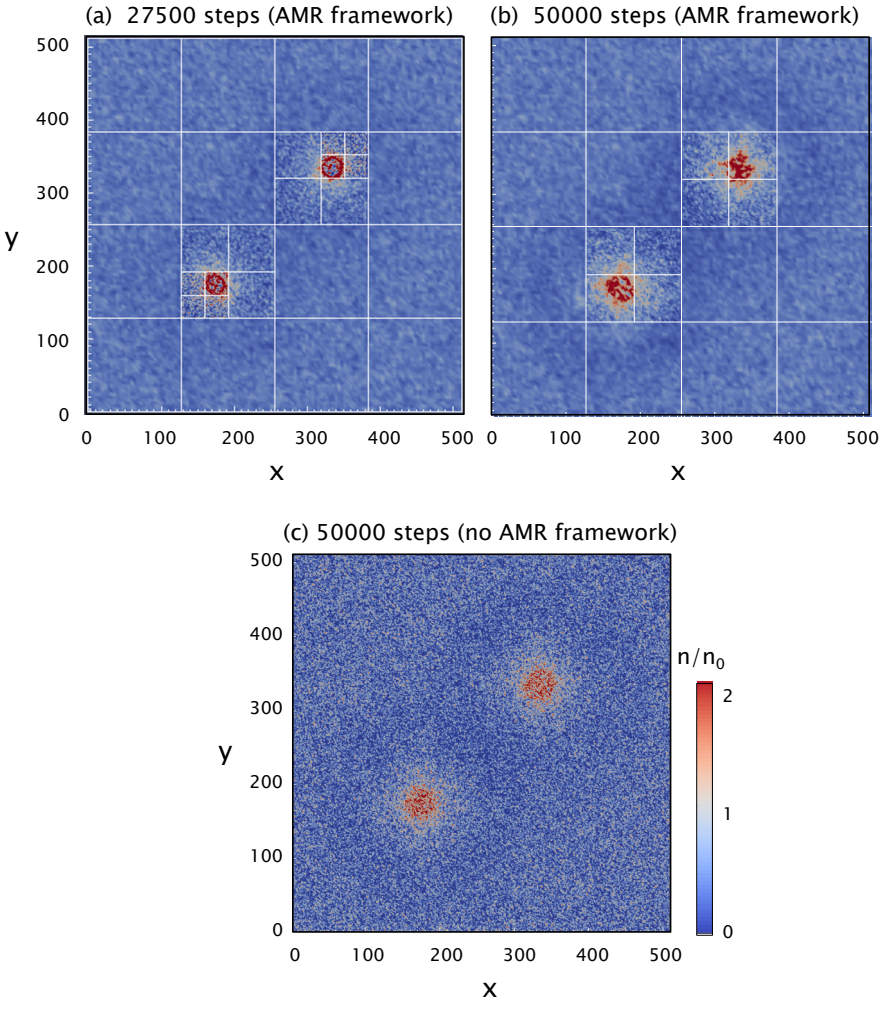

Figure 8: Inhabitants number density at (a) 27,500 steps and (b) 50,000 steps obtained in the PSS model simulation using the particle-ported AMR framework. For comparison, inhabitants number density obtained in the PSS model simulation without using the AMR framework at (c) 50,000 is also shown.

peaks of the sugar amount. It is because the inhabitants, which were uniformly distributed at the initial stage, were accelerated by $\boldsymbol{F_s}$ toward the sugar peaks. When n exceeds the criteria, child blocks with fine grids are created in the regions of sugar peaks. As shown in Panels (a) and (b), child blocks were created in the base blocks covering the areas of $128 < x, y < 256$ and $256 < x, y < 384$. Particularly in the earlier stage as shown in Panel (a), child blocks in the Level-2 layer were created in the Level-1 child blocks covering the areas of $128 < x, y < 192$ and $320 < x, y < 384$. The grid size in the Level-2 child blocks is $0.25dr$ which is quarter to that in the Level-0 layer. At the later time steps, as shown in Panel (b), the maximum density became lower than $3n_0$ which is the criteria for the generation of the Level-2 child block. Then the Level-2 child blocks became no more needed and they were removed from the simulation

domain. Then only the child blocks with the Level-1 refinement remained in the regions of $128 < x, y < 256$ and $256 < x, y < 384$.

To examine whether or not the behavior of the inhabitants shown in Panels (a) and (b) are reasonable, we compare the number density profile of Panel (b) with the one shown in Panel (c). The density profile shown in Panel (c) was obtained in the simulation using the uniform grid size of $0.25dr$ without any block in the AMR framework. Note that the gird size of $0.25dr$ in Panel (c) is the same as used in the Level-2 child blocks as shown Panel (a). It is clearly shown that the both profiles are qualitatively the same. This implies that the inhabitants agents were accelerated toward the two sugar peaks and the density maximums are eventually formed at the sugar peak positions in the both simulations. From this comparison, we can confirm that the dynamics of the inhabitants were properly simulated in the simulation using the particle-ported AMR framework. This means that the particle treatment in the blocks and the particle transfer between different sub-domains worked properly and correctly in the particle-ported AMR framework. The maximum density values, however, do not quantitatively agree when we compare Panels (b) and (c). Although we have not examined the reason in detail yet, it may be caused by the difference of the grid resolution outside the sugar peak regions between the two simulations. In the simulation corresponding to Panel (b), the Level-0 block with the grid size of $1.0dr$ is used outside the sugar peak regions while the grid size of $0.25dr$ is uniformly used in the simulation corresponding to Panel (c). Since the value of $\boldsymbol{F_s}$ used in the equation of motion for the inhabitants depends on the local grid size, the acceleration of the inhabitants to the sugar peaks becomes different between the two simulations. As a result, it can happen that the manners of the inhabitants gathering to the sugar peaks become slightly different between the two simulations, which can cause the difference of the density profiles between Panels (b) and (c) measured at the same time steps.

In the test simulation described above, we hired the inhabitants density n as a criteria for the grid refinement. Another criteria could be the value of $\boldsymbol{F_s}$ used in the equation of motion for the inhabitants as shown in Equation 2. Since $\boldsymbol{F_s}$ defined in Equation 3 depends on the local grid size, we could refine the grids in the region where the value of $\boldsymbol{F_s}$ is large around the sugar peaks.

4 Conclusions

We implemented the particle treatment in the block-structured AMR framework which we have developed. In applying the particle treatment in this framework, we set up several arrays for each process to share the particle data with the blocks within the corresponding sub-domain. We also prepare some arrays for the particle treatment in each block. The functions for these arrays for the particle treatment were described in this paper. To test the implementation of the particle treatment in the AMR framework, we adopted the sugarscape model which was proposed for the simulation of an artificial society by using many agents representing inhabitants in a certain area. We modified the original model and developed a new sugarscape model simulation program in which the agents are treated as particles which can move freely in the simulation domain. We call this model PSS model. We performed the PSS model simulation by using the particle-ported AMR framework. The simulation results show that the particles representing the inhabitants were accelerated and gathered to the regions of the sugar peaks which were initially set in the two-dimensional domain. When the number density of the inhabitants exceeds a certain value, child blocks with refined grids were adaptively created at the corresponding region. We confirmed that grid refinement up to the Level-2 layer was successfully conducted in the simulation. By comparing the simulation results with those obtained without

using the AMR framework, we could also confirm that the particle treatment in the blocks and the particle transfer between different processes worked properly and correctly in the particle-ported AMR framework.

We can also apply the particle-ported AMR framework to the multi-agent simulations which can handle various social phenomena by using a bunch of particles for the agents. One of the significant categories of the multi-agent simulation is the evacuation simulation in which we can trace and examine the human behavior in various extreme situations such as fire, flood, and earthquake.

One of the remaining issues is to improve the efficiency of simulation using the particle-ported AMR framework in terms of memory consumption and calculation time. The advantage for using the AMR method is the efficient usage of the computational resources in comparison with the uniform-grid simulations because grid refinement is carried out only at a local region where high resolution is required. In the test simulation described in the previous section, however, it turned that the computational time and the amount of the required memory are much larger than those in the simulations without using the particle-ported AMR framework. It is mainly because we defined additional arrays for the particle treatment in each block as well as in each sub-domain. The efficiency in the particle treatment inside the sub-domains and the particle transfer between different sub-domains using MPI should be improved in the next version of the particle-ported AMR framework, which is left as an important future work.

References

[1] C. K. Birdsall and A. B. Langdon. *Plasma Physics via Computer Simulation*. Plasma Physics Series. Institute of Physics Publishing, 1991.

[2] D. Dezeeuw and K. G. Powell. An adaptively refined cartesian mesh solver for the euler equations. *Journal of Computational Physics*, 104(1):56–68, 1993.

[3] J. M. Epstein and R. L. Axtell. *Growing Artificial Societies: Social Science from the Bottom Up*. The MIT Press, Cambridge, MA, USA, 1996.

[4] K. Fujimoto. A new electromagnetic particle-in-cell model with adaptive mesh refinement for high-performance parallel computation. *Journal of Computational Physics*, 230(23):8508–8526, 2011.

[5] T. Hatori, A. M. Ito, M. Nunami, H. Usui, and H. Miura. Level-by-level artificial viscosity and visualization for mhd simulation with adaptive mesh refinement. *Journal of Computational Physics*, 319:231–241, 2016.

[6] A. M. Khokhlov. Fully threaded tree algorithms for adaptive refinement fluid dynamics simulations. *Journal of Computational Physics*, 143(2):519–543, 1998.

[7] P. MacNeice, K. M. Olson, C. Mobarry, R. de Fainchtein, and C. Packer. Paramesh: A parallel adaptive mesh refinement community toolkit. *Computer Physics Communications*, 126(3):330–354, 2000.

[8] J. Muller, S. Simon, U. Motschmann, J. Schule, K. H. Glassmeier, and G. J. Pringle. Aikef: Adaptive hybrid model for space plasma simulations. *Computer Physics Communications*, 182(4):946–966, 2011.

[9] H. Usui, A. Nagara, M. Nunami, and M. Matsumoto. *Development of a Computational Framework for Block-Based AMR Simulations*, volume 29 of *Procedia Computer Science*, pages 2351–2359. 2014.

[10] H. Usui, M. Nunami, T. Moritaka, T. Matsui, and Y. Yagi. *A Multi-Scale Electromagnetic Particle Code with Adaptive Mesh Refinement and Its Parallelization*, volume 4 of *Procedia Computer Science*, pages 2337–2343. 2011.