Development of Parallelized AMR-PIC Plasma Simulation Code with Dynamic Domain Decomposition*)

Hideyuki USUI[1,3)], Yohei YAGI[1,3)], Masaharu MATSUMOTO[1,3)] and Masanori NUNAMI[2,3)]

[1)]*Graduate School of System Informatics, Kobe University, Kobe 657-8501, Japan*
[2)]*National Institute for Fusion Science, Toki 509-5292 Japan*
[3)]*Japan Science and Technology Agency (JST) / CREST, Tokyo 102-0076, Japan*

(Received 7 December 2012 / Accepted 22 August 2013)

To maintain load balance among processes in parallel calculation, we introduced a dynamic load balancing technique called Dynamic Domain Decomposition (DDD) into our newly developed multi-scale Particle-In-Cell (PIC) simulation code in which Adaptive Mesh Refinement (AMR) is incorporated. To evaluate the effectiveness of DDD, we performed test simulations with a model in which four particle clusters are non-uniformly distributed with different velocities. We confirmed that load imbalance among processes caused by non-uniform plasma distribution was successfully resolved by DDD and the computational time becomes almost half of that for simulation of the same model without using DDD.

Keywords: Particle-In-Cell, parallel computing, dynamic load balance, plasma simulation

DOI: 10.1585/pfr.8.2401149

1. Introduction

Full Particle-In-Cell (PIC) simulation [1] is a very powerful numerical method which enables us to examine various plasma phenomena in association with wave-particle interactions occurring in space because it can treat electron as well as ion kinetics with no fluid approximation. Conventional PIC simulation codes adopt uniform grids of which size is basically determined by the Debye length to avoid the numerical instability. When non-uniform plasma is treated, the grid size should be small enough in comparison with the smallest Debye length corresponding to the maximum density. The smallest grid size which is used in the high density region has to be assigned to the other regions of relatively low density. This treatment is not efficient at all in terms of the usage of limited computer resources such as the amount of memory and calculation time.

To achieve an efficient simulation with reasonable cost of computer resources, spatial and temporal resolutions can be adjusted locally and dynamically depending on the local scales of phenomena. To realize this efficiency in PIC simulations, we developed a new electromagnetic PIC code called PARMER (Full PARticle simulation code with adaptive MEsh Refinement) [2] by incorporating the Adaptive Mesh Refinement (AMR) technique. AMR has been used in the field of computational fluid dynamics as a useful method to investigate multi-scale phenomena. In the AMR simulations, grids with different spacing are dynamically created in hierarchical layers according to the local conditions of phenomena. Fine grids are applied only regions where high resolution is needed. Figure 1 shows an example of hierarchical layers with different grid size used in the AMR system. The grid size in the Level-1 layer is half of that of the base level called Level-0. In PARMER, time step interval also becomes half of that in the lower level.

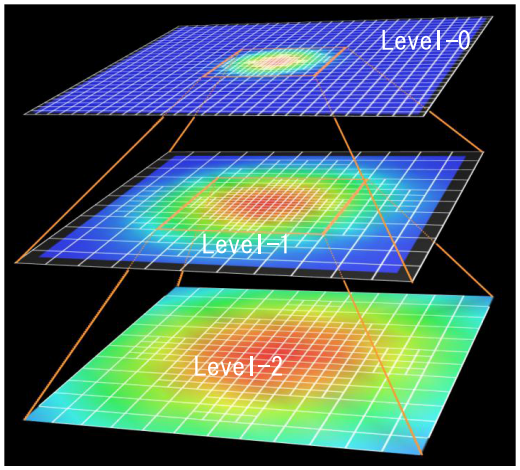

Fig. 1 Hierarchical layers with different grid size in AMR.

To realize high performance computation in PIC simulation by using a supercomputer, we usually parallelize codes by adopting the domain decomposition method. In the domain decomposition parallelization, we divide the whole simulation system into sub-domains and each sub-domain is assigned to each process. In PIC simulations, particles located in each sub-domain move across the do-

author's e-mail: h-usui@port.kobe-u.ac.jp
*) This article is based on the presentation at the 22nd International Toki Conference (ITC22).

main boundaries and eventually the number of particles in each sub-domain changes. In addition, in the AMR scheme, hierarchical grid layers are dynamically created or deleted in each sub-domain. Then the amount of calculation load assigned to each process becomes different and the load balancing between processes cannot be guaranteed through the simulation run. Such a load imbalance causes low efficiency of parallel calculation.

2. Dynamic Domain Decomposition for AMR-PIC

To resolve load imbalance in parallelized PARMER simulation, we developed the Dynamic Domain Decomposition (DDD) method with which we dynamically re-decompose the whole simulation space into new sub-domains so that each process is assigned with equal load. In PIC codes, computational task is mostly devoted to particle calculation because the number of particles is generally several ten or hundred times larger than that of spatial grids. Therefore, in DDD, we consider load balance among processes in terms of cost of particle calculation.

Figure 2 schematically shows the calculation cost of particles in a hierarchical system. In PARMER, calculation cost of particles in the child level increases twice of the parent level because the time step interval Δt becomes half of that of the parent level, $\Delta t \rightarrow \Delta t/2$, in accordance with the change of the grid spacing from Δx to $\Delta x/2$ in the child level. To synchronize to the parent level, calculation loops for particles located in the child level has to be doubled. For example, when 100 particles are located in the Level-1, they are considered to have 200 particle loops to synchronize in time with the Level-0. 100 particles located in the Level-2 have 400 particle loops from a viewpoint of the Level-0 because there are two layers difference between levels.

In consideration of the difference of the load for the particle calculation between hierarchical levels, the number of particle loops distributed to each process should be

$$\frac{\sum_{\text{cell}} 2^L N_{\text{particle}}}{N_{\text{process}}}, \tag{1}$$

where N_{particle} is the number of particles located in a grid belonging to the Level L domain, and N_{process} is the number of processes used in the parallel simulation. The base level corresponds to $L = 0$ and L increases in the higher hierarchical level with fine grids. Note that Δt at $L = 0$ is divided by 2^L on the level L. Then the numerator of the value (1) implies the total loop number of particle calculation in the whole simulation system required for updating Δt. To realize load-balanced parallel calculation, we need to somehow decompose the whole domain in such a manner that the value given by (1) becomes the same in each sub-domain.

To determine the manner of decomposing the domain into sub-domains so that load balance is achieved between processes, we first number all grid points in the simulation domain in such a manner that neighboring grids are closely ordered in a series with a space-filling curve. To realize this type of numbering the grids, we hire the Morton ordered curve [3]. Firstly a three dimensional grid (i, j, k) can be expressed in bit representation as a binary number $(i_3 i_2 i_1 i_0, j_3 j_2 j_1 j_0, k_3 k_2 k_1 k_0)$. The corresponding Morton number is then obtained by interleaving these bits into one binary number in such a manner as $(k_3 j_3 i_3 k_2 j_2 i_2 k_1 j_1 i_1 k_0 j_0 i_0)$. The obtained binary number is converted into a decimal number. All the grid points are newly numbered in the above-stated manner and we obtain one-dimensional sequential ordering of spatially close grids in memory. We decompose the grids by dividing the Morton order into the number of processes so that the average load given by (1) is equally assigned to each process.

Figure 3 shows an example of domain decomposition using DDD. Panel (a) shows a snapshot of plasma density profile obtained in a two dimensional simulation of plasma clusters. With conventional domain decomposition with fixed sub-domains, as shown in Panel (b), load balance among processes is not achieved because each plasma cluster moves in various directions and the number of particles is not constant in each sub-domain. Meanwhile, Panel (c) shows sub-domain assignment by DDD. Although the shape of each sub-domain seems irregular, each sub-domain consists of neighboring grids.

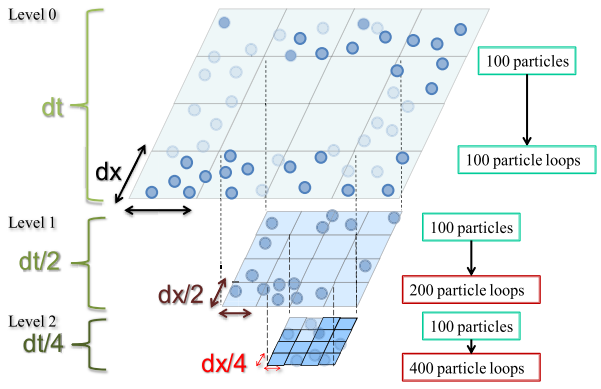

Fig. 2 Relation between the number of particle and that of particle loop in PARMER.

3. Evaluation of DDD

To evaluate DDD, we performed a test simulation using PARMER with the following model. Figure 4 shows the test simulation model. We initially have four particle clusters in a three-dimensional space in addition to uniform background plasma. Each particle cluster has constant velocity to the center of simulation space. When conventional domain decomposition with fixed sub-domains is used, load assigned to each process is not balanced and it changes in time because particle clusters move across

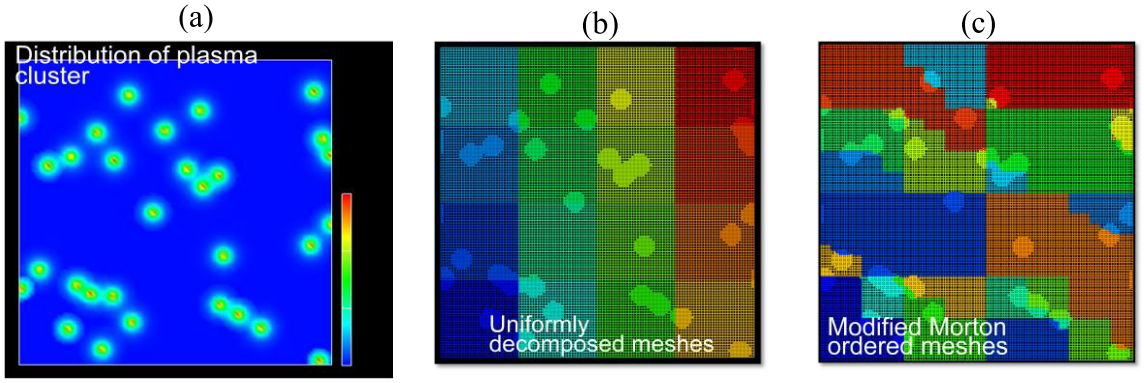

Fig. 3 A demonstration of domain decomposition. In panel (a), many plasma clouds are randomly distributed. In panel (b), fine grids are adaptively created at each cloud. A case of conventional decomposition with uniform and fixed sub-domains is shown. Each sub-domain is shown in different color. To achieve the load balance we determine the sub-domains by using the Morton ordering so that the particle loads becomes uniform between processes as shown in panel (c).

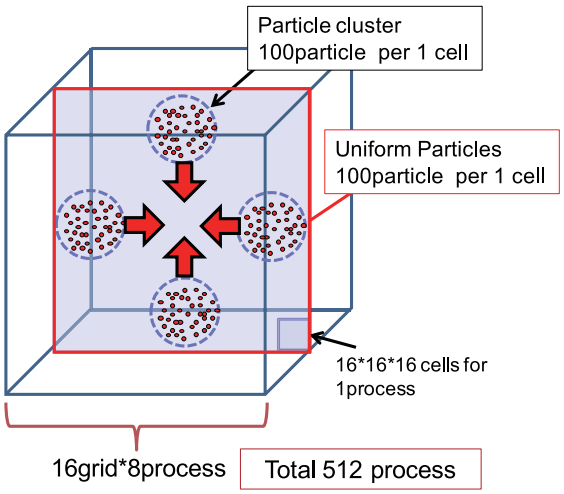

Fig. 4 Schematic image of test simulation.

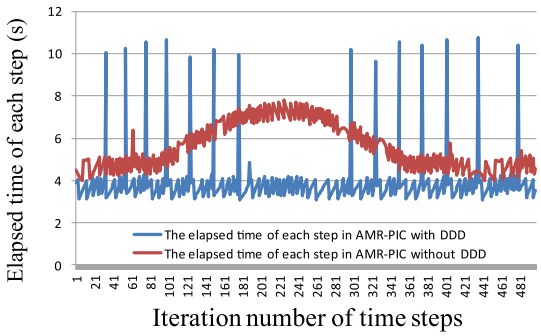

Fig. 5 Variation of elapsed time at each time step.

boundaries of sub-domains. We compare total calculation time between cases with and without DDD. The test simulations were performed with the K computer using 512 processes. In the present test simulation, two hierarchical grid layers are used. When four clusters merge in the center of the simulation space, the particle density increases fourfold. Then the spatial grids at the high density region

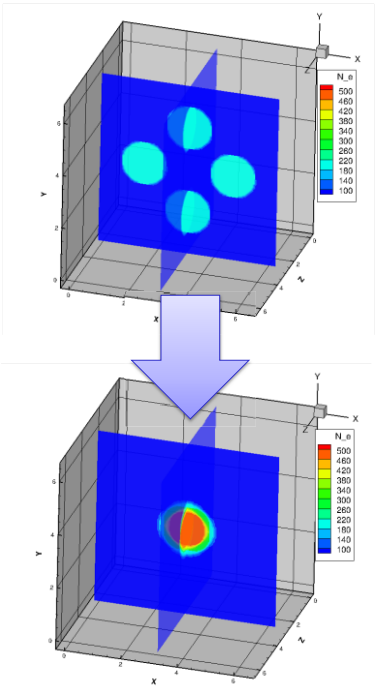

Fig. 6 Contour maps of particle density at the first step and 250 time steps.

are refined by the AMR method because the local Debye length becomes half of that of original one cluster.

Figure 5 indicates variation of elapsed calculation time at each step. Blue and red lines in the figure show cases with and without DDD. The elapsed time shown in the vertical axis implies the calculation time of the slowest process among those participated in parallel simulation. As shown in red, the elapse time for non-DDD case gradually increases and reaches the maximum value around 220 time steps. Figure 6 shows density contours of particle clusters at the initial stage and around 220 time steps when they merge in the center of the simulation space, respec-

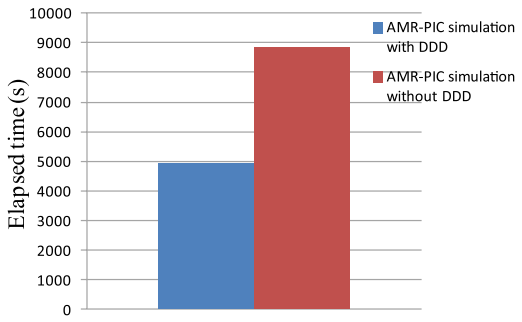

Fig. 7 Total elapsed time of PARMER with/without DDD.

tively. In non-DDD case in which fixed sub-domains are used, as clusters approach to each other to the center, the load of the process handling the center region increases in time. When the four clusters overlap in the center around 220 time steps, the elapse time becomes the maximum as shown in Fig. 5. Meanwhile, the elapsed time for the case with DDD seems almost constant, some spikes are regularly found though. The spikes are due to overhead of DDD which is mainly caused by arrangement of decomposition and data transfer between processes. For the moment, the overhead costs 1.5 times the calculation cost of one time step. Although the overhead needs to be reduced, it is negligible because we do not need to operate DDD at every time step. The overhead reduction is left as our future work.

Figure 7 shows the total elapsed time of two simulations. Blue and red colored bars correspond to cases with and without DDD, respectively. We found that the total elapsed time for the DDD case is almost half of that for non-DDD case even though some overheads exist in the DDD treatment. From the results shown in Figure 5 and 7 we can conclude that DDD works fine to resolve load imbalance problem and can decrease calculation time in this model.

4. Conclusion

We introduced dynamic load balancing technique called DDD into our AMR-PIC code called PARMER to resolve load imbalance problem. In DDD, grids points are numbered with the Morton ordering so that neighboring grids are aligned. Average particle load is obtained in consideration of sum of particle loops in hierarchical grid layers. Load balance is realized in a manner that average particle load is assigned to each process by dividing the Morton ordering into the number of process. To evaluate DDD, we performed PARMER simulations with non-uniform particle distribution model. We confirmed that load imbalance among processes was successfully resolved by DDD and the computational time becomes almost half of that for simulation of the same model without using DDD.

[1] C.K. Birdsall and A.B. Langdon, *Plasma Physics via Computer Simulation* reprinted (Taylor & Francis, New York, 2005).
[2] H. Usui *et al*., Procedia Computer Science **4**, 2337 (2011).
[3] G.M. Morton, *A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing* (IBM, Ottawa, 1966).