§22. Development of a Multi-Scale Electromagnetic Particle Code with Adaptive Mesh Refinement and its Parallelization

Usui, H. (Kobe Univ.), Nunami, M.,
Moritaka, T. (Osaka Univ.), Matsui, T. (Kobe Univ.),
Yagi, Y. (Kobe Univ.)

In plasma physics, multiscale phenomena such as electron-ion coupling are very important. However, to study the multiscale phenomena with conventional plasma Particle-In-Cell (PIC) code, enormous number of grids and macro-particles as well as huge calculation time is required because it hires uniform grid in space and time. To solve this problem and investigate multiscale phenomena including plasma kinetic effects, we have been developing a new PIC code with Adaptive Mesh Refinement (AMR) technique[1] with which we can realize high-resolution calculation saving computer resource. In the AMR-PIC code, fine meshes are adaptively, locally and hierarchically generated where some physical quantities exceeds a criteria.

To realize process parallel computing, a domain decomposition method is commonly used in which a whole simulation region is divided into the number of process and each sub-region is assigned to each process. The issue of introducing the domain decomposition to the AMR-PIC is the load balancing between processes because the number of spatial grids and macro-particles belonging to each sub-domain is not always constant. To avoid the imbalance between processes, we introduced a new scheme called dynamic domain decomposition (DDD). To determine the manner of decomposing the domain into sub-domains in DDD so that load balancing is achieved between processors, we first need to number all the cells in the simulation domain in such a manner that neighboring cells are closely ordered in a series with a space-filling curve such as the Morton ordered curve. With a space-filling curve, all the cells are numbered or ordered along one dimension. In dividing the space filling curve, we additionally need to consider the cost of particle calculation in each cell. In hierarchical system, calculation cost of particles increases twice of the parent level as approaching deep hierarchical level because the time step interval becomes half of the parent level, $\Delta t \rightarrow \Delta t/2$, according to the grid spacing becomes $\Delta x \rightarrow \Delta x/2$. To synchronize to the parent level, the calculation loops for particles located in the child level has to be doubled. Since the cost of particle calculation is dominant in particle codes is about 70-80% of the total cost, the number of particle calculation loops should be balanced between each processor. The number of particle loops distributed to each processor should be

$$\frac{\Sigma_{cell} 2^L N_{particle}}{N_{process}} \quad (1)$$

where $N_{particle}$ is the number of particles located in a cell 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 the time step interval Δt at the base level (L=0) is divided by 2L on the level L. Then the numerator of the above quantity implies the total loop number of particle calculation in the whole simulation system required for updating Δt.

Fig. 1 shows a result on DDD by performing a simple one-dimensional simulation in which a local dense plasma cloud is initially loaded in the central of the simulation space. The number of cells is 5120 and we initially put 400 particles in each cell from the cell number 2,433 to 2,688 which are located at the center region. In the other cells, we put 20 particles per cell. Note that we used 32 processors in the parallelization and there is no hierarchical grid system introduced in the simulation for simplicity. In such a situation, 2L shown in the numerator of the quantity (1) becomes the unity because of L=0 and the whole domain should be divided into sub-domains so that the number of particles in each sub-domain becomes uniform. In Fig.1, it is obviously shown that the calculation time for DDD becomes almost uniform among processors while inbalancing of calculation time is seen for the conventional fixed decomposition case. This difference implies that synchronization of calculation time among processors has been achieved for the DDD case in which uniform number of particles is assigned in each processor. Due to the achievement of the dynamic load balancing, the total calculation is shortened and the calculation becomes approximately six times faster than the conventional method for the current simple model.

We incorporated this DDD scheme in our AMR-PIC code and are performing test simulations to examine the parallelization efficiency as well as the scalability with respect to the number of processors.

1) Usui, H. et al.: Procedia Computer Science, **4**(2011) 2337.

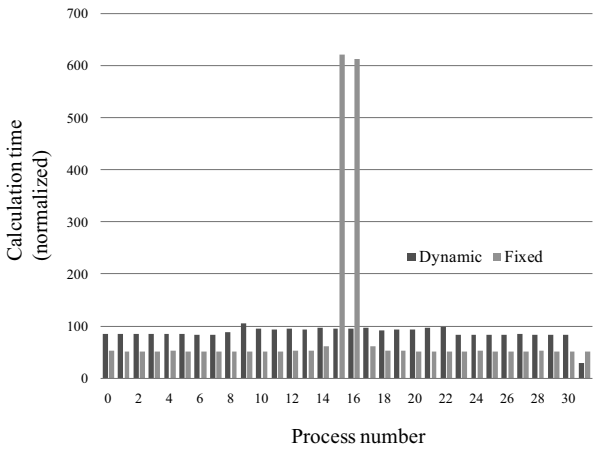

Fig. 1. One example of calculation time in each process for cases using the DDD scheme shown in red and the fixed domain decomposition shown in green.