§35. Development of a Multi-Scale Simulation Code with Adaptive Mesh Refinement and its Application to Conventional Codes

Usui, H., Yagi, Y., Matsumoto, M., Nagara, A. (Graduate School of System Informatics, Kobe Univ.), Nunami, M.

To investigate multi-scale phenomena in space plasma environment including plasma kinetic effects, we have been developing a new electromagnetic Particle-In-Cell (PIC) code by incorporating the Adaptive Mesh Refinement (AMR) technique[1]. Figure 1 shows an example of AMR-PIC simulation on interaction between a plasma flow and small-scale magnetic dipole. Four levels of hierarchical spatial grid system are adaptively created during simulation run. Fine meshes are created in regions of high number density of plasma as well as high density of magnetic field.

For the parallel computation with many processors, we adopt the domain decomposition method by which we divide the whole simulation region into sub-domains and assign them to each processor. To realize the most efficient parallel computing we need to solve the issue of load imbalance between processors. In AMR simulations, hierarchical grid layers are adaptively created or deleted at arbitrary region. Therefore the number of spatial grids as well as particles belonging to each sub-domain is not always constant. As one of the methods to resolve the load imbalance between processes in AMR-PIC simulations, we developed a new scheme called dynamic domain decomposition (DDD). In DDD, we first number all the grid cells in the simulation region in such a manner that neighboring cells are closely ordered in a series with a space-filling curve such as the Morton ordered curve. Then we decompose the cells by dividing the order into the number of processors in the following manner.

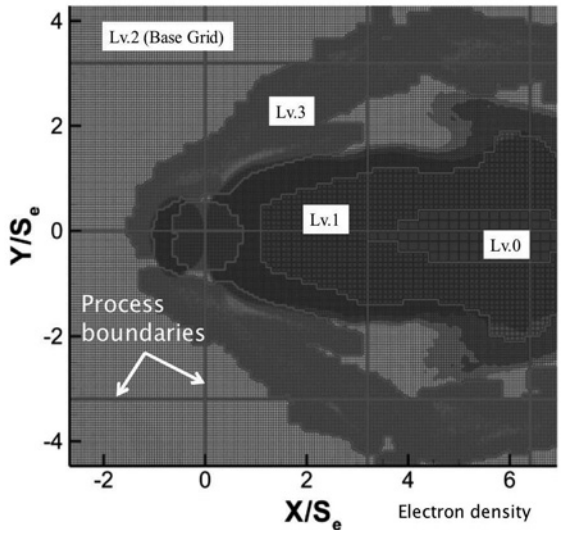

Fig. 1. Contour map of electron density for a case of interaction between a plasma flow and small dipole field located at the origin superimposed with four levels of hierarchical spatial grid system shown in red.

In our hierarchical system, 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, the calculation loops for particles located in the child level has to be doubled. In consideration of the difference of the load for the particle calculation between hierarchical levels, the number of particle loops distributed to each processor should be

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

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 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, the whole domain is decomposed with the Morton ordered curve in such a manner that the value (1) becomes the same in each sub-domain.

Figure 2 shows the effect of DDD in test simulations in which four plasma clouds with constant velocity in different directions collide in time. For no-DDD case, conventional domestic decomposition in which process boundaries are fixed is applied. When DDD is applied, total calculation time is much reduced. This result indicates the effectiveness of DDD for AMR-PIC simulation.

We have been also developing a framework with which AMR can be incorporated in conventional codes relatively easily. As an example, we are applying this framework to an MHD code developed in NIFS.

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

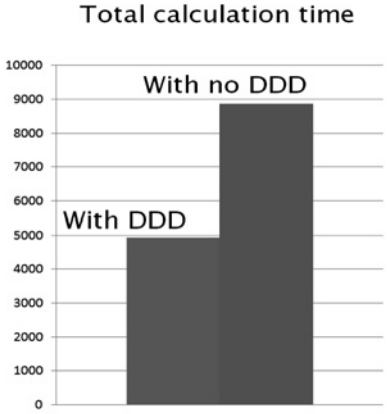

Fig. 2. Calculation time with and without DDD.