§27. Development of Full Particle-in-cell Code with Adaptive Mesh Refinement Technique and its MPI Parallelization

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

A new full kinetic particle-in-cell (PIC) simulation code with adaptive mesh refinement technique (AMR) has been developed toward large-scale numerical simulations on multi-scale plasma physics in fusion and space plasmas, such as electromagnetic interaction between an artificial magnetosphere and the solar wind[1]. In order to manage the flexible grid system required to the dynamical grid refinement, the spatial and hierarchical relationships among the spatial grids are represented as an oct-tree data structure by using pointers.[2] Super particles are collected to the nearest grids as a list data structure connected by pointers (Fig.1). The potential compatibility of the AMR-PIC code using pointers with massive parallel computers is considered by using Plasma Simulator in National Institute for Fusion Science (NIFS).

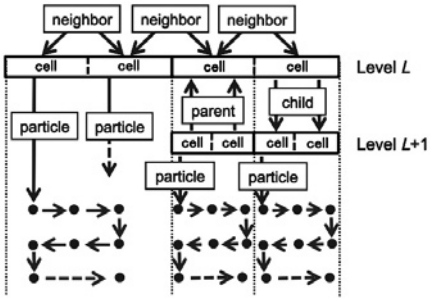

Fig. 1. Schematic diagram of pointers connecting among grid (cell) and particle data. Dots stand for super particles connected from the nearest grid by pointer 'particle'. Pointer 'neighbor' and 'parent/child' are used for the connection in the same and the other hierarchical grids, respectively.

The present code incorporated the inter-process communication using MPI for multi-node computations. The preliminary benchmark with fixed process domain without adapting mesh exhibited excellent parallel performance, with ~95% efficiency using 2048 processors (Fig.2).

Paralleling the AMR-PIC code and its optimization for large-scale distributed environments in actuality requires further complexity, due to the aforementioned unconventional data structure along with the inhomogeneous data distribution across multiple processors. Moreover, when AMR is enabled, the processors carrying finer resolution layers go through finer time steps, resulting in the significant increase in computational cost, and this phenomenon occurs dynamically during the course of simulation. Therefore it is anticipated that the traditional method of equal and stationery partition of a simulation space cannot yield a feasible computational performance. Thus, the present effort introduces dynamic domain decomposition scheme in aim to equalize the computational load on each processor.

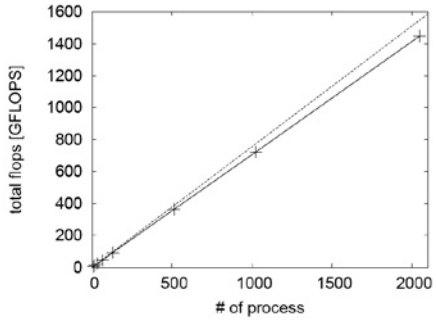

Fig. 2. Effective performance (GFLOPS) as a function of the number of used MPI process in the case of uniform domain decomposition for uniform plasmas. Dashed line indicates the theoretical performance proportional to the number of process.

Among many available space-partitioning schemes, we employ the Morton ordering in which a three dimensional space is expressed in one-dimensional array (Fig.3) [3,4]. The merit for employing is that a 3D problem can be virtually reduced into 1D problem, and more isotropic 3D decomposition can be achieved than the conventional raster numbering. Another notable feature included in this present parallelization task is the autonomic spatial managing system in which each process determines its own domain without using a "coordinator" process. This paradigm is necessitated for the recent explosive growth in process number for supercomputers, within which the program design with the central management is no longer viable.

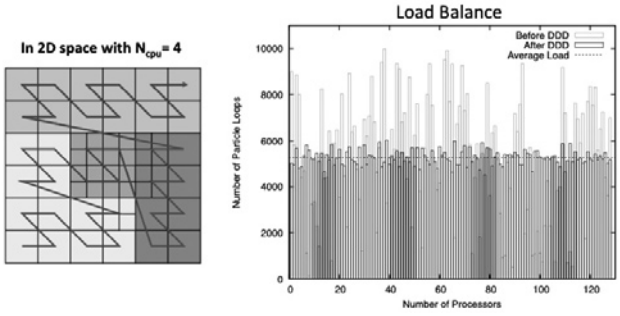

Fig. 3. (Left) Schematic picture of Morton Curve in two-dimensional case. (Right) Averaging of numerical cost (number of particle loops) for each process by using dynamic domain decomposition (DDD). Gray and black bars stand for the particle loops before DDD and that after DDD, respectively.

These innovative numerical techniques introduced in this current work are being implemented to run on massively parallel computational environments to simulate the multi-scale dynamics, and are expected to reveal new insights into space, fusion, and basic plasmas.

1) Moritaka T., et al: Plasma.Fusion.Res **6** (2011) 241101.
2) Khokhlov A.,: J.Comp.Phys **143** (1998) 519.
3) Matsui T et al.,: 19th PDP (2011) 277.
4) Usui H et al; Procedia Computer Science **4**(2011) 2337.