R008-P12
ポスター１：9/24　PM1/PM2（13:45-18:15）

#天野　孝伸 [1)], 松本　洋介 [2)]
[(1] 東大,[(2] 千葉大高等研究基幹

Dynamic Load-Balancing Framework for Kinetic Plasma Simulations

#Takanobu Amano[1)],Yosuke Matsumoto[2)]
[(1]University of Tokyo,[(2]Institute for Advanced Academic Research, Chiba University

Kinetic plasma simulations have been playing increasingly important roles in the numerical modeling of collisionless and weakly collisional space, astrophysical, and laboratory plasmas. The standard numerical method for solving Vlasov or Boltzmann equation is the so-called Particle-In-Cell (PIC) scheme. The PIC scheme approximates the velocity distribution function by the superposition of computational particles that move continuously in phase space. While it has been proven useful for modeling plasma dynamics with a relatively small number of particles, it does not necessarily fit well with the conventional parallelization strategy based on the static domain decomposition on a distributed memory system. Namely, any inhomogeneity in the plasma density naturally introduces unequal computational loads among different processes, substantially degrading the parallelization efficiency. In particular, since the inhomogeneity often significantly evolves over time, the computational load balancing must be performed dynamically. Given the ever-increasing parallelism in modern supercomputers, the dynamic load-balancing capability is now a necessary feature for a parallel PIC-type code.

We have been developing a generic dynamic load-balancing framework for PIC-type plasma simulation codes. It is based on ”chunking” the computational domain. In other words, the entire domain is divided into small chunks, with the number of chunks typically more than ten times larger than the number of processes. The chunks are dynamically distributed among different processes to balance the computational load. The framework automatically handles the distribution of chunks and boundary exchanges between the chunks (both particles and mesh quantities). It is designed such that an application programmer is able to implement solvers without knowing the details of parallelization and load balancing. A standard explicit PIC simulation code implemented on top of the framework has been working with good parallelization efficiency.

We report the present status of the framework and discuss future perspectives. Some of the objectives include the implementation of higher-order shape functions to the explicit PIC code with Esirkepov’s density decomposition scheme and a hybrid simulation algorithm (particle ions and fluid electrons). Furthremore, migration to fully asynchronous implementation using task parallelism and Kokkos library for heterogeneous architecture (including GPUs) will be discussed.