Parallelized Adaptive Mesh Refinement Particle–In–Cell Scheme with Dynamic Domain Decomposition

Hideyuki Usui$^{1,3}$, Y. Yagi$^{1,3}$, M. Matsumoto$^{1,3}$, M. Nunami$^{2,3}$ and A. Nagara$^{1,3}$

$^1$Graduate School of System Informatics, Kobe University, Japan
$^2$National Institute for Fusion Science, Toki, Japan
$^3$Japan Science and Technology Agency(JST)/CREST, Tokyo, Japan
Outline

- Multi-scale interactions between weak dipole field and the solar wind
- PARMER: PIC code using adaptive mesh refinement
- Parallelization of PARMER
- Summary
Earth’s magnetosphere

- Magnetopause located around $10 R_e$
- Ion inertia length in solar wind ($S_i$): 100 km

Magnetosphere is much larger than $S_i$

This system is based on the solar wind interaction with a weak dipole field (10s kilometers)

Magnetic field structure is smaller than $S_i$

Small scale dipole field interaction

Mag-sail

Converts solar wind momentum to the thrust through interaction with dipole field

Spacecraft with a superconducting coil
Magnetosphere formation (smaller than ion inertia length)

In a case of weak dipole field, almost no interaction will take place because ions are very little affected by the dipole field. However,

- Electron scale magnetosphere is created
- Cross-field current at the boundary to sustain the field structure
Multi-scale plasma particle simulation

Multi-scale characteristics

- Spacecraft: ~ 1m
- Dipole field: ~ 1-10 km
- Ion inertia length: 100 km

Multi-scale analysis including plasma kinetic effects

- Conventional 3D EM-PIC simulation codes hire uniform and fixed spatial grid system → needs huge resources.
- It is better that the resolution is adaptively adjusted during a simulation run, depending on local plasma phenomena.

Combination of AMR (Adaptive Mesh Refinement) and PIC
Multi-scale interaction between weak dipole field and the solar wind

PARMER: PIC code using adaptive mesh refinement

Parallelization of PARMER

Summary
AMR–PCI Simulation of the solar wind – small dipole field interaction

• Four hierarchical levels
• Criteria for the mesh refinement
• Fine mesh at the boundaries and coarse mesh in the tail
• Fixed process boundaries
• Load balance not maintained

Lv.0
Lv.1
Lv.2 (Base Grid)
Lv.3

Electron density

Process boundaries
• Multi-scale interaction between weak dipole field and the solar wind

• PARMER: PIC code using adaptive mesh refinement

• Parallelization of PARMER

• Summary
Domain decomposition is commonly used.
PARMER hires the domain decomposition method.
Particles and field data located at process boundaries are shared through MPI.
Causes of Load Imbalance in PARMER

- Particles can move over process boundaries then the number of particles in each process domain changes in time.
- Due to local mesh refinement, the number of grids assigned to each process dynamically changes.
Test simulation (test particle model)

- Particle cluster (200 particle/cell)
- Background (100 particle/cell)
- 16grid*8process
- Total 512 processes
- Two cases (Fixed process boundaries and dynamic domain decomposition)
Introduction of Dynamic Domain Decomposition (DDD)

Calculation time for each time step

- Load balance is achieved with DDD
- Spikes imply overheads for the DDD treatment (negligible)

The calculation time is shortened because the load balance is obtained between processes.
Dynamic Domain Decomposition (DDD) scheme (1)

How do we realize the dynamic load balancing for the mesh refinement case?

**Step 1:** Number all the base cells (Level 0) in such a manner that neighboring cells are closely ordered in a series.

Use a space-filling curve such as the Morton order curve
Step 2: Evaluate the computational load based on the number of particle loops

In dynamic load balancing, the particle loops in the hierarchical layers should be taken into account.
Step 3: Calculate the total particle loops and obtain the particle load for each process.

![Mathematical expression](image)

\[
\sum_{cell} 2^L N_{\text{particle}} \quad \frac{N_{\text{process}}}{N_{\text{process}}}
\]

This coefficient is necessary depending on which level of layer the particles are located.

\(N_{\text{particle}}\): the number of particle in each cell
\(L\): the hierarchical level of the corresponding cell
\(N_{\text{process}}\): the number of process in parallelization
Step 4: Divide the Morton curve into the number of process and decompose the domain so that the particle load becomes the same in each sub-domain.

Each process has approximately the same amount of particle load.
Summary

• We have been developing a new plasma particle code for multi-scale simulation by incorporating adaptive mesh refinement.

• To achieve the load balancing in the domain-decomposition parallelization, we consider the particle load by counting the particle loops and decompose the domain by using a space filling curve.

• We will evaluate the efficiency of the parallelization in detail by using the larger number of processes by using K-computer.