24P150-P

Acceleration of plasma Particle-In-Cell simulation by GPU

プラズマ PIC 法シミュレーションの GPU による高速化

Kosuke Sawamoto, Yasunori Mizuno and Hiroshi Inuzuka
澤本　晃佑, 水野　保則,　犬塚　博

Shizuoka University 3-5-1, Johoku, Naka-ku, Hamamatsu 432-8561, Japan
静岡大学浜松　〒432-8561 静岡県浜松市中区城北 3-5-1

Particle-In-Cell (PIC) method is widely used in plasma simulation. In this poster we report the efficient algorithm that runs PIC codes on Graphics Processing Unit (GPU). Especially, we focus on algorithms for the process of particle-to-grid calculation which is the bottle neck operation for GPU. We try to optimize the video memory access pattern by using a hash search algorithm. The calculation time of GPU is approximately 30x faster than that of CPU (Intel Core2 Duo).

1. Introduction

PIC methods are one of the most important methods for plasma simulation and spend a long time when a lot of particles are operated.

The peak performance of NVIDIA GPU GeForce GTX570 we use in our study is approximately 1.3TFLOPS. Using GPU for general purpose computing, called General Purpose GPU (GPGPU), is become wide spread and we expect that GPU can speed up PIC simulation too. In this poster, we report about our algorithm for GPU, its calculation time and simple simulation result.

2. PIC method

PIC method handles electric and magnetic field as differential values on calculation grids [2]. The code contains below (i-iii) processes. Process (i) calculates the grid charge density from particle position, process (ii) calculates the grid field from grid charge density by solving Poisson's equation and process (iii) moves particle from grid field.

We have implemented all 3 processes to compute on GPU. Within these processes, (i) is the most difficult process to optimize for GPU. Process (i) executes the calculation of equation (1) that depends on index of particle i and index of grid j. Operating these two indexes makes parallelization difficult. To solve this problem, existent algorithms [3,4] keep particles sorted state.

$$q_j = \sum_i q_i \frac{X_{j+1} - x_i}{\Delta x}\ , q_{j+1} = \sum_i q_i \frac{x_i - X_j}{\Delta x} \quad (1)$$

X: grid position, x: particle position, q: particle or grid charge, i: index of particle, j: index of grid.

2. Algorithm

When we implement PIC code on GPU, it is the most important to optimize the pattern of video memory access because its access speed depends on the pattern of parallel access [1]. Therefore the random access that depends on particle index i or grid index j of equation (1) is very slow. Access time of the worst case is about 1/8 slower than that of the best pattern.

In our study, we try to use a hash search algorithm to dissolve the random access at one dimensional

electrostatic model. Fig. 1 shows the image of data structure. In this algorithm, particles that exist in particular range are stored in a linked list and links point the top of array. 32 particles are stored in this array and one list is operated by 32 parallel processes. This data structure can optimize the video memory access by handling proper size array for fast memory access pattern with one link.

In this data structure, particles that have moved to another cell which handled by the neighbor list should be inserted to the proper list after the process (iii). We draw these particles by using packing operation.

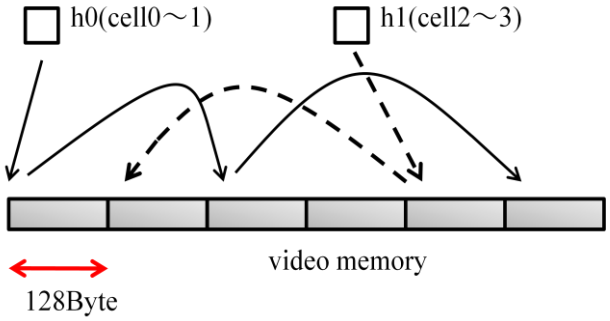

Fig. 1. An image of data structure

4. Results

We execute a simple simulation, electron-electron two-stream instability [2], to measure calculation times and to ensure that the simulation result calculated by GPU is correct.

Table I shows CPU (Intel Core2 Duo) times and GPU times of processes (i-iii) and packing. These times are average values of 1000 steps. In this simulation, the number of particle is 8M and the number of grid is 65536 and CPU executes a single thread. Although there is some overhead of packing operation, GPU time is faster.

Fig. 2 shows v_x-x phase-spaces of simulation result. In images of phase-space, position x versus velocity v of particles are plotted. In this simulation, the number of particle is 2M and the number of grid is 16384. Since the result of CPU and of GPU is same and we can see the instability occurs among two streams, we can conclude that the result of GPU is correct.

Table I. calculation time

	CPU time [ms]	GPU time [ms]
(i) density	158.8	1.5
(ii) field	29.3	0.16
(iii) move	233.7	1.9
packing	-	9.3
total	421.8	12.9

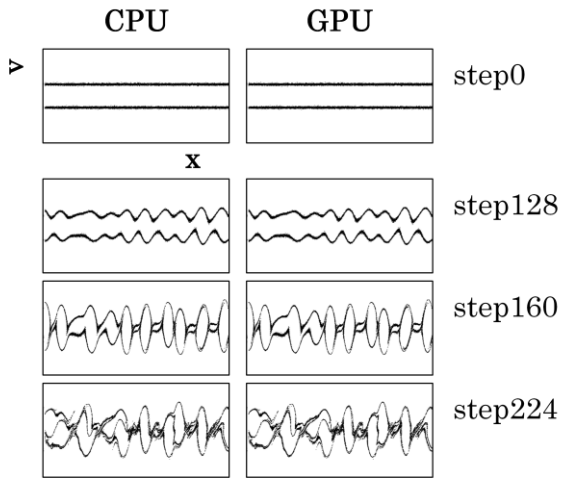

Fig. 2. Electron-electron two stream instability

References

[1] NVIDIA: CUDA Programming Guide. (2011)

[2] C. K. Birdsall and A. B. Longdon: *Plasma physics via computer simulation*. (1985)

[3] George Stantchev, William Dorland, Nail Gumerov: *Fast parallel Particle-To-Grid interpolation for plasma PIC simulations on the GPU*, j. Parallel Distrib. Comput. 68(2008) 1339-1349

[4]Victor K. Decyk and Tajendra V. Singh: *Plasma Particle-in-Cell Codes on GPUs: A Developer's Perspective*, http://idre.ucla.edu/hpc/research/documents/HPCC-victor-gpu.pdf (2010)