Bull. Nov. Comp. Center, Comp. Science, 42 (2018), 61–66

Optimization of the Particle-In-Cell method for general-purpose computers*

A.V. Snytnikov

Abstract. Presented in this paper are the software techniques to improve the performance of the Particle-in-Cell method for general-purpose computers equipped with processors like Intel Xeon/Nehalem or AMD Phenom. The software techniques include particle storing in cells in either fixed size array or list, field values kept together in one array. One must note that since these techniques depend on the method only, they are suitable for virtually all computer architectures.

Introduction

This paper was inspired by the effect of anomalous heat conductivity observed at the GOL-3 facility in the Budker Institute of Nuclear Physics [1]. The GOL-3 facility is a long open trap where the dense plasma is heated up in a strong magnetic field when injecting a powerful relativistic electron beam of a microsecond duration. The effect is in a decrease of the plasma electron heat conductivity by 100 or 1000 times compared to the classical value for the plasma with the temperature and density observed in the experiment. Anomalous heat conductivity arises because of the turbulence that is caused by the relaxation of the relativistic electron beam in the high-temperature Maxwellian plasma. The physical problem is to define the origin and mechanism of the heat conductivity decrease. This is of great importance for the fusion devices, because the effect of anomalous heat conductivity helps in heating the plasma and, also, in confining it. The problem of heat transport in the fusion devices was widely discussed in [2, 3] and in some recent publications [4].

Several papers dealing with the super-particle management in the PIC method have recently appeared. First of all, the research by Lapenta [5, 6] and a more general approach presented by Welch [7] should be mentioned. The first approach deals with the statistical problems arising on adaptive grids when there are too few super-particles in a cell. The second approach deals with the particle sources on a uniform grid such as ionization, emission from boundaries, etc. Comparing the super-particle management of the algorithms proposed by Lapenta and Welch, one can see that there is one major difference between them. In the first case, the pairwise coalescing of the super-particles conserve charge, energy, and momentum.

*Supported by RFBR under Grants 18-07-00364 and 16-07-00434.

Optimization of the particle storage in the PIC method was also done in [8, 9].

1. Model description

1.1. Basic equations. The mathematical model employed for the solution of the problem of the beam relaxation in plasma consists of the Vlasov equations for ion and electron components of the plasma and also of the Maxwell equation system. These equations in the conventional notation have the following form:

$$\frac{\partial f_{i,e}}{\partial t} + \vec{v}\frac{\partial f_{i,e}}{\partial \vec{r}} + \vec{F}_{i,e}\frac{\partial f_{i,e}}{\partial \vec{p}} = 0, \qquad \vec{F}_{i,e} = q_{i,e}\Big(\vec{E} + \frac{1}{c}[\vec{v}, \vec{B}]\Big),$$

$$\operatorname{rot}\vec{B} = \frac{4\pi}{c}\vec{j} + \frac{1}{c}\frac{\partial \vec{E}}{\partial t}, \qquad \operatorname{rot}\vec{E} = -\frac{1}{c}\frac{\partial \vec{B}}{\partial t},$$

$$\operatorname{div}\vec{E} = 4\pi\rho, \qquad \operatorname{div}\vec{B} = 0.$$

In the present study this equation system is solved by the method described in [10]. All the equations will be further given in the non-dimensional form using the following basic quantities:

- characteristic velocity of light $\tilde{v} = c = 3 \cdot 10^{10}$ cm/s,
- characteristic plasma density $\tilde{n} = 10^{14}$ cm^{-3},
- characteristic time $\tilde{t}$ is the plasma period (a value inverse to the electron plasma frequency) $\tilde{t} = \omega_p^{-1} = \left(\frac{4\pi n_0 e^2}{m_e}\right)^{-0.5} = 5.3 \cdot 10^{-12}$ s.

The Vlasov equations are solved by the PIC method. This method implies the solution to the equation of motion for model particles, or superparticles. The quantities with the subscripts i and e are related to ions and electrons, respectively:

$$\frac{\partial \vec{p}_e}{\partial t} = -\Big(\vec{E} + [\vec{v}_e, \vec{B}]\Big), \qquad \frac{\partial \vec{p}_i}{\partial t} = \kappa\Big(\vec{E} + [\vec{v}_i, \vec{B}]\Big),$$

$$\frac{\partial \vec{r}_{i,e}}{\partial t} = \vec{v}_{i,e}, \qquad \kappa = \frac{m_e}{m_i}, \qquad \vec{p}_{i,e} = \gamma\vec{v}_{i,e}, \gamma^{-1} = \sqrt{1 - v^2}.$$

The leapfrog scheme to solve these equations is employed:

$$\frac{\vec{p}_{i,e}^{\,m+1/2} - \vec{p}_{i,e}^{\,m-1/2}}{\tau} = q_i\left(\vec{E}^m + \left[\frac{\vec{v}_{i,e}^{\,m+1/2} - \vec{v}_{i,e}^{\,m-1/2}}{2}, \vec{B}^m\right]\right),$$

$$\frac{\vec{r}_{i,e}^{\,m+1} - \vec{r}_{i,e}^{\,m}}{\tau} = \vec{v}_{i,e}^{\,m+1/2}.$$

Here τ is the timestep.

The scheme proposed by Langdon and Lasinski is used to obtain the values of electric and magnetic fields. The scheme employs the finite-difference form of the Faradey and the Ampere laws. A detailed description of the scheme can be found in [10]. The scheme yields the second order of approximation with respect to space and time.

1.2. Problem statement. Let us consider the following problem statement. The 3D computation domain has the shape of a parallelepiped with the following dimensions:

$$0 \le x \le L_X, \qquad 0 \le y \le L_Y, \qquad 0 \le z \le L_Z.$$

Within this domain there is model plasma. The model plasma particles (the superparticles) are uniformly distributed within the domain. The density of plasma is set by the user as well as the electron temperature. The temperature of ions is considered to be zero. Beam electrons are also uniformly distributed along the domain. Thus, the beam is considered to be already present in the plasma, and the effects that occur while the beam is entering the plasma, are beyond the scope of this study.

The superparticles simulating the beam electrons differ from those simulating the plasma electrons by the value of their energy. The beam electrons initially have the energy of about 1 MeV, and the plasma electrons have the energy of about 1 keV. Moreover, the beam electrons have one direction of movement strictly along the axis X, and the plasma electrons have the Maxwellian velocity distribution for all the three dimensions.

There is one more difference between the superparticles simulating the beam electrons and the plasma electrons. They have different weights when computing the current and the charge density. Let us consider the ratio α of the beam density to the plasma density (usually α varies from 10^{-3} to 10^{-6}), then the contribution of a beam electron superparticle is α from the contribution of a plasma electron superparticle. In such a way it is possible to provide a large number of beam superparticles.

The main physical parameters of the problem under study are the following: the density and the temperature of the plasma electrons, the ratio of the beam density to the plasma density, and the energy of the beam.

2. Optimization

2.1. The need for optimization. Model particles are randomly situated inside the computation domain. Even if the particle coordinates are stored nearby in the coordinate array, this does not mean that the coordinate values are similar. It result in very random access to field arrays: in order to push a particle one needs to fetch the field values from the cell the particle is

located in. It is impossible to use the field values that were used for the previous particle since the current one is located in a different cell.

The cache usage would be much more efficient if the particles were ordered. Then the field values fetched for pushing a particle could be used once again for the next particle if it is closely located. To make the particles ordered, it is sufficient to attach them to cells, that is, to store the particles in some way in a cell. Full-scale sorting of particles is not necessary because the order of particles inside a cell is not important.

2.2. Particle optimization. Particles located in a cell could be stored in the form of a list or an array. The advantages of the list are obvious: the number of particles is not limited, a simple addition and removal, but there are also some disadvantages: a greater access time as compared to the array. If the particles of a cell are stored in the form of an array (a static array), then, for a 3D simulation, it results in a 5D array just for one coordinate (the coordinate X, for example). We are reminded that a particle has 6 attributes — three coordinates and three impulses.

The main bottleneck for the case of a static array is a limited maximal number of particles in a cell. The size of an array necessary to store all the particles in every cell cannot be defined before carrying out simulation. It is known from the simulations done that the maximal value of the density was 5 times higher than the average. This means that the maximal number of particles in a cell could be set to $5N$, N being the initial value of particles in a cell. In this case, the array will be 5 times bigger and its size will reach 70 Gb, for a mesh with $512 \times 64 \times 64$ nodes, and $N = 150$, for example. Here the mesh size is 150 Mb per mesh (six 3D arrays for a field and three 3D arrays for the current) and the amount of memory for particles (if they are stored in the usual form of six 1D arrays for the whole domain), is 15 Gb. So, it is clear that the optimization involving static arrays is fairly expensive in terms of memory.

The employment of dynamic re-allocation of memory (lists of particles in each cell) enables one not to use too much memory, but it requires implementing an efficient memory manager within the code, which is most likely possible, but it will scarcely result in the overall runtime decrease. In both cases the subroutine that implements particle pushing, receives 6 small arrays (with size not exceeding $5N$) containing the particle coordinates and impulses for a particular cell. These 6 arrays are formed from either a list or an array storing the particles for this cell.

All the above-mentioned optimizations were implemented in the code presented.

2.3. Field optimization. The electromagnetic field handling was optimized in the following way: all the six 3D arrays that store three components

of an electric field and three components of a magnetic field are arranged in one 4D array containing all the six above-mentioned 3D arrays at once and using one more index to identify a particular 3D array.

The 4D field array is organized in the following way: fd$[i][l][k][num]$, where i, l, and k are the indices for X, Y, and Z dimensions, respectively, and num codes a particular field component (0 for E_x, 1 for E_y, 2 for E_z, and 3 to 5 for a magnetic field in the same manner). The most important thing here is that all the values in one 3D cell denoted by the same i, l, k are stored closely in RAM. This results in a more efficient use of the cache. Once one of the field components (E_x, for example), is fetched, at the same time all the rest five components are also present in the cache, and since they all are necessary, and the fetching requests are about to be issued, this saves a lot of time.

2.4. Results. The efficiency of optimization is show in the table. The tests were conducted with a workstation equipped with AMD Phenom processor and with one of the nodes of NSU cluster (processor Intel Nehalem). In both cases the mesh was set big enough that even one of the field arrays does not fit the cache, that is, $64 \times 32 \times 32$ nodes with 50 particles in each cell.

Particles pushing time, one timestep, in seconds

Optimization type	AMD Phenom	Intel Nehalem
Basic non-optimized	13.25	7.22
Fields in 4D array	8.8	6.72
Particles in arrays in each cell (static 5D array)	12.51	5.67
Particles in lists for each cell (dynamic re-allocation)	10.5	10.3
Fields in 4D array and particles in arrays in each cell	10.92	3.67

It is evident that combining particle ordering (storing particles in arrays or in lists in each cell) with arranging an electromagnetic field into 4D array results in a significant increase in performance.

References

[1] Astrelin V.T., Burdakov A.V., Postupaev V.V. Generation of ion-acoustic waves and suppresion of heat transport during plasma heating by an electron beam // Plasma Physics Reports. — Vol. 24, No. 5. — P. 414–425.

[2] Cohen B.I., Barnes D.C., Dawson J.M., et al. The numerical tokamak project: simulation of turbulent transport // Computer Physics Communications. — May, 1995. — Vol. 87, Iss. 1–2. — P. 1–15.

[3] Jaun A., Appert K., Vaclavik J., Villard L. Global waves in resistive and hot tokamak plasmas // Computer Physics Communications. — December, 1995. — Vol. 92, Iss. 2–3. — P. 153–187.

[4] Gardarein J.-L., Reichle R., Rigollet F., et al. Calculation of heat flux and evolution of equivalent thermal contact resistance of carbon deposits on Tore Supra neutralizer // Fusion Engineering and Design. — October, 2008. — Vol. 83, Iss. 5–6. — P. 759–765.

[5] Lapenta G. Particle rezoning for multidimensional kinetic particle-in-cell simulations // J. Comput. Phys. — September, 2002. — Vol. 181, Iss. 1. — P. 317–337.

[6] Lapenta G., Brackbill J.U. Dynamic and selective control of the number of particles in kinetic plasma simulations // J. Comput. Phys. — November, 1994. — Vol. 115, Iss. 1. — P. 213–227.

[7] Welch D.R., Genoni T.C., Clark R.E., Rose D.V. Adaptive particle management in a particle-in-cell code // J. Comput. Phys. — 2007. — Vol. 227. — P. 143–155.

[8] Anderson D.V., Shumaker Dan E. Hybrid Ordered Particle Simulation (HOPS) code for plasma modelling on vector-serial, vector-parallel, and massively parallel computers // Computer Physics Communications. — 1995. — Vol. 87, Iss. 1–2. — P. 16–34.

[9] Tskhakaya D., Schneider R. Optimization of PIC codes by improved memory management // J. Comput. Phys. — 2007. — Vol. 225, Iss. 1. — P. 829–839.

[10] Vshivkov V.A., Grigoryev Yu.N., Fedoruk M.P. Numerical "Particle-in-Cell" Methods. Theory and Applications. — Utrecht-Boston: VSP, 2002.