Bull. Nov. Comp. Center, Comp. Science, 36 (2014), 79–88

Parallel template implementation of Particle-in-Cell method for hybrid supercomputers*

A.V. Snytnikov, A.A. Romanenko

Abstract. Recently, hybrid (GPU-equipped) supercomputers have have attained a very high performance level. The fact is, the solution of real physical problems with such supercomputers is restricted by the GPU programming complexity.

In order to simplify the development of high-performance plasma physics codes for hybrid supercomputers, a template implementation of Particle-in-Cell (PIC) method was carried out. The template parameters are problem-specific implementations of "particle" and "cell" (as C++ classes).

Thus, it is possible to develop a PIC code for a new plasma physics problem without GPU programming. Instead, the new physical features are just included into the existing code as new implementations of "particle" and "cell" classes.

1. Introduction

The objective of the present paper is to create a tool for the fast development of the new problem-oriented PIC codes for GPUs based on one highly optimized PIC code. Such a code is developed for the described problem of the beam-plasma interaction. Of course, there are a lot of PIC codes for GPUs (e.g. [1, 2]), but they are mostly targeting at a single particular computational or an algorithmic aspect and thus they are not yet suitable for physical computations.

The idea that might give an advantage to most of the exiting codes is that an optimized template code is provided by a programmer, and physics is introduced by the physicist as template parameters. This might give an advantage over fast but physically dubious codes written by programmers and, also, over correct but slow codes written by physicists.

This paper has been inspired by the effect of anomalous heat conductivity observed at the GOL-3 facility in the Budker Institute of Nuclear Physics [3]. 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 decreasing of the plasma electron heat conductivity by 100 or 1,000 times as compared to the classical value for the plasma with temperature and density observed in the experiment. An anomalous heat conductivity arises because of the turbulence that is caused by a relaxation of the relativistic electron beam

*Supported by the RFBR under Grants 14-07-00241 and 14-01-31088, and by the SB RAS Integration Projects 103 and 113.

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 confining it. The problem of heat transport in fusion devices was widely discussed (e.g. in [4, 5]) and some recent published works [6].

The novelty of the present research has the two aspects: the physical aspect and the numerical aspect. By present, from the physical point of view, heating the plasma with a beam has been considered for a long period of time, but the details of the process, namely, the parameters of the arising plasma instabilities are still unknown. The existing theory of the beam heating uses too many simplifications like strictly monochromatic beam, Maxwellian plasma, etc. Our objective is to determine the instabilities and to evaluate their parameters.

The numerical aspect of the novelty is that such problems are usually solved by means of the direct Boltzmann equation (e.g. [7]). The PIC method is expected to give a better picture of the turbulence and the underlying plasma instabilities, although it requires much more computational efforts, as pointed in [4], in order to obtain a physically realistic picture. As a matter of fact, in Russia there are no finite-difference Vlasov solver aimed at high-temperature turbulent plasma simulation, though about ten foreign solvers could be mentioned. This paper is the first step in developing such a solver.

This problem needs high-performance computing because of the necessity to have a large enough grid to simulate the resonance interaction of a relativistic electron beam with plasma. The beam interacts with the plasma through the electric field (similar to the Landau damping), thus it is necessary to simultaneously observe two different scales. The first is the Debye plasma length and the second is the beam-plasma interaction wavelength, which is 10 or 100 times longer than the Debye length. Since one should provide at least 8 grid cells with the Debye length, it is possible to estimate the size of the grid.

It is also necessary to provide a large number of particles for each cell of the grid for the simulation of turbulence. The level of non-physical statistical fluctuations is inversely proportional to the number of particles per cell. So if there are too few particles, all the physical plasma waves and oscillations will be suppressed by non-physical noise.

2. Model description. The basic equations

This section is aimed at displaying the equations that are solved in all plasma physics problems. They are the suitable equations for the template implementation of the PIC method.

The mathematical model employed for solving the problem of 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 usual notation have the following form:

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

$$\operatorname{rot}\vec{B} = \frac{4\pi}{c}\vec{j} + \frac{1}{c}\frac{\partial \vec{E}}{\partial t}, \qquad \operatorname{div}\vec{B} = 0,$$

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

In our case, this equation system is solved by the method described in [8]. All the equations will be further given in the dimensionless form. The following basic quantities are used for the transition to this form:

- characteristic velocity is the 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 particles. The quantities with the subscript i are related to ions and those with the subscript e — to electrons:

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

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

The leapfrog scheme is employed to solve these equations:

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

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

Here τ is a time step. 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 [8]. The scheme gives second order of approximation with respect to space and time.

3. Parallel implementation

This section deals with showing the parallelization strategy that must be supported by the PIC implementation template, since it is quite clear that the plasma physics problems of interest will never fit into a single CPU of GPU memory.

The program was parallelized by the domain decomposition method. The computational domain is divided into parts along the direction orthogonal to the direction of the beam (along the axis Y, the beam moving along the axis). The computational grid in the whole domain is divided into equal parts (subdomains) along the axis Y. Each subdomain is assigned to a group of processors (in the case of a multicore system a single core would be called a processor, since no hybrid parallelization like MPI+OpenMP is employed, just mere MPI). Furthermore, the particles of each subdomain are distributed uniformly between processors of the group with no regard to their position, as is shown in Figure 1.

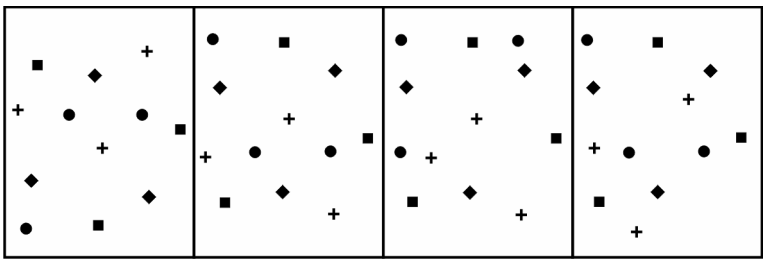

Figure 1. The scheme of domain decomposition. The computational domain is divided into 4 subdomains. The particles of each subdomain are uniformly distributed among four processors with no regard to their position. Different symbols (a circle, a square, a diamond, a star) denote particles belonging to different processors in the same subdomain

Every processor in the group solves the Maxwell equations in the whole subdomain and exchanges boundary values of the fields with processors assigned to the adjacent subdomains. Then the equations of motion for the particles are solved, and the 3D matrix of the current density and the charge density are evaluated by each processor. But since the processor has only a part of the particles located inside the subdomain, it is necessary to sum the matrices through all the processors of the group in order to obtain the whole current density matrix in the subdomain. Interprocessor data exchange is performed by the MPI subroutines.

Parallelization efficiency. A parallel program was primarily developed for the simulation of the beam interaction with plasma on large computational grids and with large numbers of particles. That is why the parallelization efficiency was computed in the following way:

$$k = \frac{T_2}{T_1} \cdot \frac{N_1}{N_2} \cdot \frac{S_2}{S_1} \cdot 100\ \%.$$

Here T_1 is the computation time with N_1 processors, T_2 is the computation time with N_2 processors, S is the characteristic size of the problem in each case. Here the characteristic size is the grid size along the axis X. In this section, the characteristic size S is proportional to the number of processors N. It means that the workload of a single processor is constant. The purpose of such a definition of efficiency is to find out what the communication overhead is when the number of processors is increased with the constant workload for each processor. In the ideal case, the computation time must remain the same (the ideal $k = 100\ \%$).

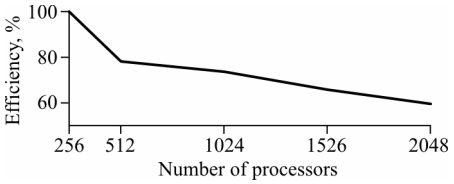

Figure 2. The parallelization efficiency measured with MVS-100K cluster. The grid size along Y and Z is 64 nodes, the grid size along X is equal to the number of processors; 150 particles per cell for all the cases

In the computations dealing with the efficiency evaluation, the axis X of the only grid size was increasing, all the other parameters remaining constant. The results obtained are shown in Figure 2.

4. The GPU implementation

The implementation of the above PIC algorithm for the GPUs is quite standard. The field evaluation method are ported to the GPU almost without any change. The computation speed id high enough even without optimization. The field arrays are stored in the GPU global memory.

The bottleneck of the PIC codes is the particle push. With the CPUs, it takes up to 90 % of runtime. So, first the particles are distributed among cells. This step only reduces the push time twice with the CPUs. With the GPUs it is even more important since it enables the use of texture memory (texture memory is limited and the whole particle array will never fit). The second step is keeping the field values related to the cell (and, also, to the adjacent cells) in the cell itself. This is important since each particle needs 6 field values and writes 12 current values into the grid nodes, and now all is done within a small amount of memory (the cell) without addressing the global field or current arrays that contain the whole domain. Then the evaluated currents from all cells are added to the global current array.

This gives speedup of about 10 for Tesla 2,070 as compared to a single Xeon core. It is not so much, but any sophisticated optimization has not been applied yet.

5. The template implementation of the PIC method

In order to attain the objective (the tool for the fast development of the new problem-oriented PIC codes for GPUs) it is necessary to:

1. Develop an optimized GPU implementation of the PIC method for a particular problem.
2. Create a set of diagnostics tools to facilitate the analysis of results by a physicist.
3. Provide an option to replace the problem-specific parts of a computational algorithm.

In order to achieve the latter point, the C++ templates are being used. It means that the "computation domain" class is implemented that contains "cell" class objects. The "cell" class contains "particle" class objects. Here the "computation domain" class is a template class with "cell" class as a parameter, "cell" being a template itself, with "particle" as a parameter.

For a wide variety of the PIC method implementations, most of the operations of a cell with its particles are absolutely the same (adding/removing a particle, particle push, the evaluation of a particle contribution to the current). Only the gridless particles method of gyrokinetic codes might be an exception. And even them fit the proposed scheme since they just do not need any operations, they do not introduce anything new. This fact gives a hope that these operations once implemented as a template will be efficient for a number of problems solved with the PIC method.

Also, the operations of the computation domain with cells are absolutely the same. The things that differ are the initial distribution and the boundary conditions. Thus, these operations must be implemented as virtual functions.

Since particle attributes and operations with particles are similar in most cases, it is possible to create a basic implementation of the "particle" class containing a particle position, momentum,charge and mass. Then, if for some new physical problem the "particle" needs new attributes, then a derived class is implemented, and this new class "derived particle" is used as a parameter to the "cell" template class.

At present, there are object-oriented implemtations of the PIC method (e.g. OOPIC library), and also the template libraries for PIC method [9, 10], however for the CPU-based supercomputers, but not for hybrid ones.

Development of the PIC template for GPUs. The porting of the implemented PIC method template to the GPU was carried out in the following way:

On the basis of the PIC method template (class Plasma), a derived class was created (class GPUPlasma), which is also a template. In the GPUPlasma the following methods were added:

- copying the domain to the GPU,
- comparison of the CPU and the GPU results, and
- invocation of the GPU kernels for field evaluation.

The implementation of the "cell" for the GPU (GPUCell class) was inherited from Cell class. It is important that particle storage within a cell must be optimized in terms of the GPU memory, thus the structure or class arrays are not suitable. The GPUCell class also includes the copying to and from a device and comparison of the GPU and the CPU cells. In this case, "particle" implementation for the GPU is exactly the same as for CPU. Here for debugging it is necessary that computation methods are implemented just once for both CPU and GPU.

6. Electron heat conductivity in computational experiments

This section displays a scheme of the anomalous heat conductivity effect. The exact physics needs much higher numbers of particles and grid sizes.

In order to simulate the interaction of the electron beam with plasma, the following values of the main physical parameters were set:

- electron temperature of 1 KeV,
- the mass of ion 1836 electron masses (hydrogene ions),
- plasma density of 10^{17} cm^{-3},
- the ratio of beam density to plasma density of 10^{-3},
- beam energy of 1 MeV,
- the size of the domain of $L_X = 0.065$ cm and $L_Y = L_Z = 0.008$ cm,
- the grid size of $512 \times 64 \times 64$ nodes, 150 particles per cell.

Density modulation was observed in the computational experiments. The amplitude of the modulation is 220 % of the initial density value. Modulation in this case means the emergence of regions with a very high or a very low density in the previously uniform-density plasma as is shown in Figure 3. It is seen that the density becomes non-uniform not only along the direction of the beam (the axis X), but also along the axis Y. Thus the density is modulated not along X, that seems quite natural, but also along Y and Z. This corresponds to the physics of the process well, because it is known that the waves propagating in plasma due to beam relaxation have all the three components of wave-vector as non-zeros.

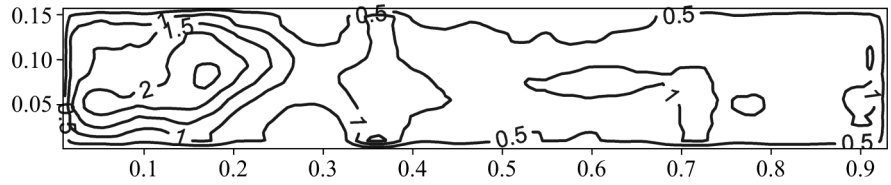

Figure 3. Electron density contours in the plane XY, $z = L_Z/2$, the instant time $t = 91.7$ (in terms of the plasma period). The density is given in terms of the initial values of density

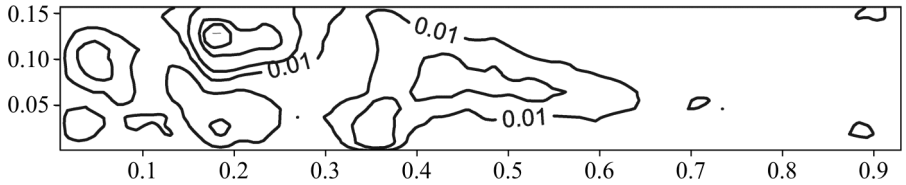

Figure 4. Electron heat flux contours in the plane XY, $z = L_Z/2$, the instant time $t = 91.7$ (in terms of the plasma period). The flux is given in terms of the initial values of the electron heat flux

Moreover, it was found out that the movement of beam electrons becomes eddy as a result of the beam interaction with plasma. At the initial moment of instant time all the beam electrons have the same velocity strictly along the axis X. This results in the eddy structure of the electron heat flux

$$q(x, y) = \left|T_e(x, y, L_Z/2)\vec{v_e}(x, y, L_Z/2)\right|.$$

The electron heat flux also gains modulations along X and Y. Moreover, there are regions with a very low value of the electron heat flux as is shown in Figure 4 (less than 1 % of the initial electron heat flux). It means that only in small and isolated regions the value of electron heat flux is close to initial value, but generally in the computational domain the electron heat flux is very low. Thus the domain as a whole has very low heat conductivity after the beam relaxation.

At the present time, with the GPU computations the exact values of two-stream instability increments were obtained, corresponding to the values of [11].

7. Conclusion

The parallel implementation of the numerical model of the interaction of the electron beam with plasma is described, and the parallel efficiency is discussed. The worktime of the program has been measured and analyzed on various clusters. The scheme of the anomalous heat conductivity effect is shown. The exact physics needs much higher numbers of particles and grid

sizes. The GPU computations resulting in the exact values of two-stream instability increments were obtained, corresponding to the exact theoretical value.

The development of the PIC method template for the GPU is described with a preliminary speedup of the order of magnitude compared to a single Xeon core. The template enables us to create a GPU program for the new physical problem from rather a wide field of plasma physics and beam physics just by adding the new physical features as particle and cell attributes.

References

[1] Kong X., Huang C., Ren Ch., Decyk V. Particle-in-cell simulations with charge-conserving current deposition on graphic processing units // J. Comp. Phys. — 2011. — Vol. 230, Iss. 4. — P. 1676–1685.

[2] Rossi F., Londrillo P., Sgattoni A., et al. Towards robust algorithms for current deposition and dynamic load-balancing in a GPU particle in cell code // 15th Advanced Accelerator Concepts Workshop, June 10–15, 2012, Austin, Texas, USA. — 2012. — P. 184. — (AIP Conf. Proc.; 1507); http://dx.doi.org/10.1063/1.4773692.

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

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

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

[6] 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. — 2008. — Vol. 83, Iss. 5, 6. — P. 759–765.

[7] Wright J.C., Bonoli P.T., D'Azevedo E., Brambilla M. Ultrahigh resolution simulations of mode converted ion cyclotron waves and lower hybrid waves // Computer Physics Communications. — 2004. — Vol. 164, Iss. 1–3. — P. 330–335.

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

[9] Decyk V. Skeleton PIC codes for parallel computers // Computer Physics Communications. — 1995. — Vol. 87, Iss. 1–2. — P. 87–94.

[10] Malyshkin V.E., Tsigulin A.A. Generator of parallel programs implementing numerical models // Avtometriya.— 2003.— No. 3.— P. 124–135 (In Russian).

[11] Timofeev I.V., Lotov K.V. Relaxation of a relativistic electron beam in plasma in the trapping regime // Phys. Plasmas— 2006.— Vol. 13, No. 6.— 062312.