High performance PIC plasma simulation with modern GPUs

Alexey V Snytnikov 1 and Alexey A Romanenko 2
1 Institute of Computational Mathematics and Mathematical geophysics SB RAS
2 Novosibirsk State University
E-mail: snytav@ssd.sscc.ru, arom@ccfit.nsu.ru

Abstract. With the recent Nvidia Tesla V100 a performance of 0.5 TFLOPS was achieved for 3D Particle-In-Cell simulation. The paper includes brief description of the simulation algorithm, the detail of GPU implementation and the performance analysis with different GPUs.

1. Introduction
The main idea of this work is to achieve high performance with supercomputers and GPUs in order to conduct high resolution 3D simulations of the fusion plasma in different experimental facilities.

The physical meaning of this work is to investigate the effect of anomalous heat conductivity observed at the GOL-3 fusion 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 during the injection of the powerful relativistic electron beam of a microsecond duration. The effect is the 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. At present, the effect has just a principal theoretical explanation [2].

The goal of theoretical and modelling studies 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 to heat the plasma and also to confine it.

Like other plasma physics problems with non-Maxwellian velocity distribution this problem requires kinetic treatment. It means either direct finite-difference solution of Vlasov equation or Particle-In-Cell[3] method. While PIC method is noisy and time-consuming, still it is in many cases the only method that is able to treat the problem with no non-physical simplifications.

There are a lot of PIC packages, like OSIRIS[4], LCODE[5] or VLPL[6], including those adopted for GPU, e.g. PIConGPU[7], ALaDyn[8], and also some works aimed to particular questions of GPU implementation of PIC method, like [9] and [10], but since they are all problem-specific, we make an implementation on our own.

PIC method consists of two stages: field evaluation and particle pushing. Computational experiments with general purpose computers (e.g. the Lomonosov supercomputer at MSU) show that most time is spent in particle pushing (from 60% to 90%). Since the total time is
big: about a week with 128 Intel Xeon cores for a 3D problem with just $100 \times 4 \times 4$ mesh nodes and 10000 particles for each cell [11], and further study requires meshes of much bigger size, it requires running on GPUs.

2. Basic equations
The basic equations are 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:

$$\frac{\partial f_{i,e}}{\partial t} + \vec{v} \frac{\partial f_{i,e}}{\partial \vec{r}} + \vec{F}_{i,e} \frac{\partial f_{i,e}}{\partial \vec{p}} = 0,$$

$$\vec{F}_{i,e} = q_{i,e} \left( \vec{E} + \frac{1}{c} [\vec{v}, \vec{B}] \right)$$

$$\text{rot} \vec{B} = \frac{4\pi}{c} \vec{j} + \frac{1}{c} \frac{\partial \vec{E}}{\partial t}$$

$$\text{rot} \vec{E} = -\frac{1}{c} \frac{\partial \vec{B}}{\partial t}$$

$$\text{div} \vec{E} = 4\pi \rho$$

$$\text{div} \vec{B} = 0$$

(1)

Here $f_{i,e}$ is the distribution function for ions and electrons, respectively, $t$ is time, $\vec{v}$ is the velocity vector, $\vec{r}$ is the coordinate vector, $\vec{F}$ is the Lorentz force, $\vec{p}$ is the impulse, $\vec{E}$ is the electric field, $\vec{B}$ is the magnetic field, $\rho$ is the charge density.

All the equations below are given in non-dimensional units. The following quantities are used for transition to non-dimensional form:

- velocity of light $c = 3 \times 10^{10}$ cm/s
- mass of electron $m_e = 9.1 \times 10^{-28}$ g
- plasma density $n_0 = 10^{14}$
- time $t = \omega_{pe}^{-1}$, where plasma electron frequency $\omega_{pe} = 5.6 \times 10^{11}$ s$^{-1}$.

The 3D computational domain has the shape of a cube with the following dimensions:

$$0 \leq x \leq L_X, \quad 0 \leq y \leq L_Y, \quad 0 \leq z \leq L_Z$$

Within this domain there is the model plasma that consists from electrons and hydorgen ions. The model plasma particles are distributed uniformly 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. Iniitl distribution of particles by velocities is Maxwellian:

$$f(v) = \frac{1}{\Delta v \sqrt{2\pi}} \exp\left(\frac{-(v - v_0)^2}{2\Delta v^2}\right)$$

here $\Delta v$ - electron velocity dispersion, $v_0$ - average velocity ($v_0 = 0$ for plasma particles. 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 the study.
3. Details on Physical Statement of the Problem
The physical statement of the problem exactly follows the simulations in [11], the difference is
in simulation time which in the present case is by orders of magnitude smaller. As it is stated
in [11] beam relaxation may proceed in three modes, depending on the instability increment $\gamma$
and beam velocity dispersion $\Delta v$: kinetic ($k \Delta v \gg \gamma$), hydrodynamical ($k \Delta v \ll \gamma$) and transit
mode ($k \Delta v \sim \gamma$). Here $k$ is the wavenumber

The goal of the computations is not to simulate the facility as a whole - since no supercomputer
is able to do it now or in the future. Instead, we are going to reveal the underlying instabilities
and reproduce the turbulence observed in experiments.

In such a way, the computational domain corresponds to a very small (characteristic size
$\sim 10^{-3}$ cm) volume of plasma.

4. Description of the GPU implementation
Vlasov equation is solved by the Particle-In-Cell method [3]. In order to understand the ideas
and bottlenecks of GPU implementation of PIC method let us first consider the underlying
numerical techniques.

4.1. Numerical method
The essential of the PIC method is the substitution of Vlasov equation by the set of characteristic
equations. The characteristic equations exactly resemble Newton motion equations for plasma
particles, and due to this reason they are called the motion equations for model particles.

One must note that these model particle are just the mathematical tool for solving Vlasov
equation. This is not a replacement of very big number ($N \sim 10^{23}$) of small particles in real
plasma by a small number ($N \sim 10^{10}$) of big particles in model plasma.

4.1.1. Particle push
In such a way, the equations of motion for model particles are the following:

$$\frac{\partial \vec{p}_{i,e}}{\partial t} = q_{i,e} \left( \vec{E} + \vec{v} \times \vec{B} \right) , \quad \frac{d \vec{r}_{i,e}}{dt} = \vec{v}_{i,e} , \quad \vec{p}_{i,e} = \frac{\vec{v}_{i,e}}{\sqrt{1 - v_{i,e}^2}} \tag{2}$$

here $i,e$ are the indices for ions and electrons, respectively, $\vec{p}$ is impulse, $\vec{r}_{i,e}$ is the coordinate
vector, $\vec{E}$ - electric field, $\vec{v}$ is the velocity and $\vec{B}$ - magnetic field. Equations 2 are solved by the
leapfrog scheme:

$$\frac{p_{i,e}^{m+1/2} - p_{i,e}^{m-1/2}}{\tau} = q_{i,e} \left( \vec{E}^m + \frac{\vec{v}_{i,e}^{m+1/2} - \vec{v}_{i,e}^{m-1/2}}{2} \times \vec{B}^m \right)$$

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

here $\tau$ is the timestep.

4.1.2. Field solver
For the evaluation of electromagnetic field the Langdon-Lasinski [3] method
is employed:

$$\frac{\vec{B}^{m+1/2} - \vec{B}^{m-1/2}}{\tau} = \text{rot}_h \vec{E}^m$$

$$\frac{\vec{E}^{m+1} - \vec{E}^m}{\tau} = \text{rot}_h \vec{B}^{m+1/2} - \vec{j}^{m+1/2}$$
4.2. GPU implementation principles

The following main principles are used to achieve high GPU performance:

- shared memory is used for computations with particles;
- particles are stored in arrays attached to cells in order to use the cache more efficiently;
- field and current values (8 of them for each field and current component) are computed and stored in cells;
- particle reordering is used to speedup particle pusher.

The computational algorithm consists of the following stages performed at each time step:

- particle push, that is the computation of new coordinates and impulses of each particle. At present, the evaluation of current is included in the push, [12, 13];
- electromagnetic field evaluation, [14];
- reordering of particle, which means the transfer of particles to the cells they belong to [15];
- copying of currents from cells to the global arrays [14];
- copying of fields from global arrays to cells [14].

For each of the stages a special kernel is devoted in GPU implementation. The implementation of each kernel is given in more detail in the above-listed papers.

5. Performance measurements with different GPUs

The results of profiling of the code execution is presented in the table 1.

Performance is evaluated in the following way: the computational domain has 6.4 million particles, and near 500 floating point operation are spent for a particle at a time step. The total number of operations is the divided to the measured pushing time. The other parts of the numerical algorithms, like field evaluation of are not considered since they play almost no role both in terms of time and of flops.

Table 1. Particle pushing time, measured by the Nvidia Visual Profiler and the resulting performance in flops for different GPUs, time in microseconds

| Device name     | Pushing time | Performance  |
|-----------------|--------------|--------------|
| Geforce GTX 1050 Ti | 225.4 ms     | 14.1 GFLOPS  |
| Tesla K40m      | 434.4 ms     | 7.3 GFLOPS   |
| Tesla V100      | 6.5 ms       | 0.492 TFLOPS |

A part of the profile for the NVidia Tesla V100 is shown in figure 1. One can see the device name (Tesla V100 SXM2) as well as kernel duration (6.58132 ms). Is can be noticed also that there no big pause between the kernels, so the device is busy most of time.

6. Conclusion

The performance achievement of nearly 0.5 TFLOPS by a single NVidia Tesla V100 for 3D Particle-In-Cell simulation is presented. It means that now it is possible to perform the simulation of the size mentioned in Introduction in 4 hours with one V100 instead of a week with a ordinary cluster of 128 cores (the cluster at Siberian Supercomputer Center, based on rather old Intel Xeon 5540 4-core processors).

The performance shown in the paper is lower though comparable with the top Particle-In-Cell simulation speed of 1 TFLOPS shown in [16] for different architecture (a single Intel Xeon Phi). Future plans are to reduce current accumulation time by eliminating atomic operations and thus shorten pushing time.
Figure 1. A part of the performance profile for V100 GPU. The focus is on the kernel GPU_StepAllCells that performs particle push.

Acknowledgments

This work was supported by RFBR grant 18-07-00364, computational experiments were performed under support of RFBR grant 16-07-00434.

References

[1] Astrelin V T, Burdakov A V and Postupaev V V 1998 Plasma Physics Reports (5)
[2] Burdakov A V, Kotelnikov I A and Erofeev V I 2005 Fusion Science and Technology 47 74–77 (Preprint https://doi.org/10.13182/FST05-A611) URL https://doi.org/10.13182/FST05-A611
[3] Vshivkov V A, Grigoryev Y N and Fedoruk M P 2002 Numerical Particle-in-Cell Methods. Theory and applications (Utrecht-Boston: VSP)
[4] Fonseca R A, Silva L O, Tsung F S, Decyk V K, Lu W, Ren C, Mori W B, Deng S, Lee S, Katsouleas T and Adam J C 2002 Computational Science — ICCS 2002 ed S l o o tPMA ,H o e k s t r aAG ,T a nCJKa n d Dongarra J J (Berlin, Heidelberg: Springer Berlin Heidelberg) pp 342–351 ISBN 978-3-540-47789-1
[5] Lotov 2012 (accessed August 11, 2018) LCODE main site URL http://www.inp.nsk.su/~lotov/lcode/
[6] Tuckmantel T, Pukhov A, Liljo J and Hochbruck M 2010 IEEE Transactions on Plasma Science 38 2383–2389
[7] Burau H, Widera R, Honig W, Juckelang G, Debus A, Kluge T, Schramm U, Cowan T E, Sauerbrey R and Bussmann M 2010 IEEE Transactions on Plasma Science 38 2831–2839
[8] Rossi F, Lendrillo P, S Gavin Al, Sinigardi S and Turchetti G 2012 AIP Conference Proceedings 1507(1) 184–192 (Preprint https://doi.org/10.1063/1.4773692) URL https://doi.org/10.1063/1.4773692
[9] Kong X, C Huang M, Ren C and K Decyk V 2011 230 1676–1685
[10] Rieke M, Trost T and Grauer R 2015 Journal of Computational Physics 283 436–452 (Preprint https://doi.org/10.1061/j.jcp.2014.12.016) URL https://doi.org/10.1061/j.jcp.2014.12.016
[11] Lotov K V, Timofeev I V, Mesyats E, Snytnikov A V and Vshivkov V A 2015 Physics of Plasmas 22(2) (Preprint https://doi.org/10.1063/1.4907223) URL https://doi.org/10.1063/1.4907223
[12] Snytnikov A and Romanenko A 2014 Computer Science 36 79–88
[13] Snytnikov A and Romanenko A 2013 Numerical Analysis 16 81–92
[14] Snytnikov A V, Boronina M A, Mesyats E A and Romanenko A A 2015 Russian Supercomputing Days 2015 (MSU)
[15] Romanenko A A and Snytnikov A V 2018 Vestnik Udmurtskogo Universiteta
[16] Nakashima H 2015 Computers & Electrical Engineering 46