A Multi-threaded Particle-in-cell Approach for Kinetic Plasma Simulations

Marius Joldoş[1]
marius.joldos@cs.utcluj.ro
Anca Hangan[1]
anca.hangan@cs.utcluj.ro

Gabriel Voitcu[2]
gabi@spacescience.ro
Marius Echim[2]
echim@spacescience.ro

Alin Suciu[1]
alin,suciu@cs.utcluj.ro
Anca Marginean[1]
anca.marginean@cs.utcluj.ro

[1]Technical university of Cluj-Napoca
Computer Science Department

[2]Institute of Space Science, Magurele, Romania

Abstract—**Particle-in-cell (PIC) simulations for plasma physics model self-consistently plasma phenomena at kinetic scales. Such simulations focus on the individual trajectories of a very large number of particles in self-consistent and external electric and magnetic fields, and are of great importance in astrophysics and space sciences. We developed a multi-threaded application to simulate such phenomena. This article describes the parallel performance of our application on a high-performance multicore system**

Keywords—plasma simulation, parallel computing, multithreading.

I. Introduction

Particle-in-cell (PIC) simulations provide a useful tool to investigate self-consistently plasma processes at microscopic (kinetic) scales [1] [2]. In PIC simulations, the dynamics of plasma is studied by following the trajectories of a significantly large number of particles in their self-consistent and external electric and magnetic fields. Among various approaches, the 3D electromagnetic PIC method provides a full description of plasma by considering both self-consistent electrostatic and electromagnetic effects at microscopic level. Recently, this method has been used by [3] [4] [5] to investigate the interaction of high-speed plasma jets with non-uniform magnetic fields in a simplified magnetopause-like configuration typical for a northward interplanetary magnetic field. This is an important and active research topic in space plasma physics that is highly relevant for the magnetospheric environment of planet Earth (e.g. [6]), but also for understanding the interaction of planetary plasmas with solar and stellar winds or the dynamics of astrophysical relativistic jets (e.g. [7] [8]).

The 3D electromagnetic PIC approach is the most suitable tool to address the issue of high-speed plasma jets interaction with magnetic boundaries/discontinuities since it allows for the simultaneous investigation of key physical effects as self-polarization, finite Larmor radius effects and electromagnetic processes along all relevant directions, namely convection, tangential and parallel directions with respect to the magnetic field orientation (e.g. [5]). This kind of simulations are computationally demanding processes due to the highly restrictive requirements imposed by the discretization of space and time that can lead to unpractical execution runtimes (as explained in [2]). Therefore, in order to achieve reduced execution runtimes and extend their applicability towards more complex magnetospheric configurations, parallelization is required.

Over time, various parallelized applications have been developed for particle-in-cell simulation of plasmas. In [9] , Di Martino et al. used the particle decomposition technique with High Performance Fortran to parallelize a hybrid fluid-PIC code developed for simulations of turbulence in magnetically confined plasmas. In [10] and [11] Qiang et al. applied the particle-field and domain decomposition approaches to a 3D electrostatic PIC code used for beam dynamics simulations. In [12] and [13] Barsamian et al. developed multicore algorithms for 2D and 3D electrostatic PIC simulations. All these implementations and many others are built for specific problems and research areas. The applicability domain for our PIC algorithm – finite-size 3D plasma structures interaction with non-uniform transverse magnetic fields, requires a 3d3v full-electromagnetic model [5]. We consider here an implementation based on the particle decomposition technique with a multi-threaded approach. The use of multithreading has the advantage that communication of the information changed is not needed, as all threads have access to the whole program memory. This results in faster execution times.

The paper is organized as follows: Section II briefly presents the stages of the algorithm used, the kind of simulations it can do and how it was parallelized; Section III describes the experiments carried out using the implementation and the results obtained; Section IV presents the conclusions and gives information on future work.

II. Simulation Description

A. Algorithm stages

We use here an explicit and relativistic 3d3v electromagnetic PIC algorithm developed for three-dimensional kinetic simulation of fully-ionized collisionless magnetized plasmas [14] [15] The PIC simulation proceeds as a repeated cycle of four main stages. First, the Newton-Lorentz equation of motion for each simulated particle is integrated explicitly in the total (internal + external) electric and magnetic fields by using the Buneman-Boris leap-frog method [2]. Second, the

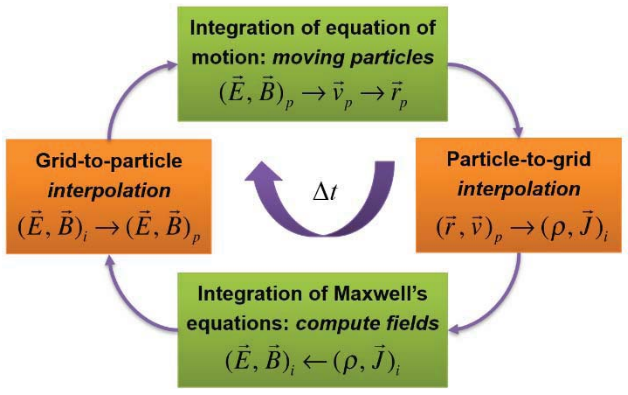
Integration of equation of motion: *moving particles*
$(\vec{E},\vec{B})_p \rightarrow \vec{v}_p \rightarrow \vec{r}_p$
Grid-to-particle *interpolation*
$(\vec{E},\vec{B})_i \rightarrow (\vec{E},\vec{B})_p$
Δt
Particle-to-grid *interpolation*
$(\vec{r},\vec{v})_p \rightarrow (\rho,\vec{J})_i$
Integration of Maxwell's equations: *compute fields*
$(\vec{E},\vec{B})_i \leftarrow (\rho,\vec{J})_i$

Fig. 1. The stages of the PIC simulation

current density is computed in the spatial nodes of the simulation grid by using the current deposition of particles technique [16]. To compute the charge density, the first order weighting scheme (linear interpolation) is used [2]. Third, Ampère and Faraday's equations are integrated explicitly using finite-differences in both space and time to obtain the self-consistent electric and magnetic fields in the location of each grid-cell defined by the Yee lattice [17]. Fourth, the self-consistent electric and magnetic fields are interpolated from the grid-cells back to the location of each particle by using the same weighting scheme as before. In our implementation, the last stage is directly coupled with the first one into a single module. Fig. 1 illustrates the stages of the 3d3v PIC simulation.

Stages 1 (particle mover) and 2 (current computation) of the PIC cycle are the most time-consuming tasks performed at each time-step of the simulation. Both stages require multiple resource-intensive operations that put a heavy work on the processors. Moreover, in order to reduce the intrinsic statistical noise and to limit the possible undesired effects of the aliasing that could arise due to the discretization of configuration space, a smoothing procedure is applied to the current density (following [2]). For more details on the 3D PIC algorithm used here, see [15].

B. *Types of Simulations*

Four simulation configurations were chosen and implemented to study the interaction of localized high-speed plasma jets with magnetic discontinuities. These configurations have been specifically selected to overcome the physical limitations of the previous studies on this topic [5] [3] [4] and to advance the understanding of the mass, momentum and energy transfer at the interface between the solar wind and planetary
magnetospheres. Here are the four configurations:

- A - Plasma jet in vacuum interacting with a prescribed tangential discontinuity with magnetic shear characterized by:
    - A1 - non-zero magnetic field gradient;
    - A2 - constant magnetic field magnitude.
- B - Plasma jet in background plasma environment interacting with a uniform transverse magnetic field.
- C - Plasma jet in background plasma environment interacting with a self-consistent tangential discontinuity.

Configurations A1 and A2 have been selected to study the effects produced by the rotation of the background magnetic field on the dynamics of plasma jets streaming in vacuum across a prescribed tangential discontinuity. Configuration B shall be used to investigate the interaction between a background plasma population and high-speed plasma jets moving across uniform transverse magnetic fields. In the last configuration (C), we consider the simultaneous interaction of the high-speed plasma jets with the self-consistent tangential discontinuity and the background plasma environment.

C. *A Parallel Approach*

The parallel code is based on a sequential version developed by Gabriel Voitcu and Marius Echim that embraces the methodology discussed in section II.A [15] and the simulation configurations presented in section II.B. Fig. 2 illustrates the main steps of the parallel algorithm.
The single thread parts are depicted in blue, and are executed only by one thread, which we will call the main thread. The main thread has its share of the particle domain to handle. The parallel parts, executed by all threads, are shown in orange: move particles (3) and compute current densities (5) which belong to the simulation loop. They correspond to stages 1 and 2 mentioned before. The computation of the current densities is performed using separate memory areas by each thread. These is the cause that step (5) includes a phase in which the main thread sums up the currents computed by the other threads.
The horizontal red dashed lines represent barriers (synchronization points).
In what follows every step is explained:

- In the initialization step:
    - Configuration data is read from a file, particle positions and velocities are set,
    - initial current densities and fields are computed, and
    - The data is saved to files.

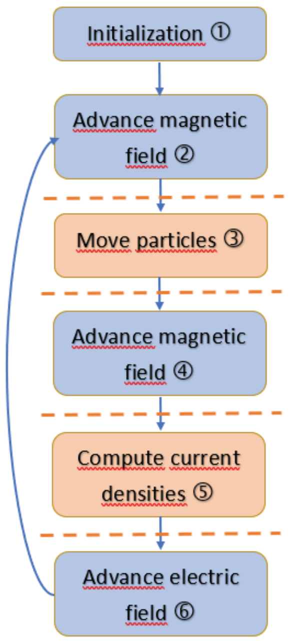

Fig. 2. The main steps of the parallel algorithm

Each thread is assigned an equal share of the total number of particles; the work assigned to each thread is proportional to the number of particles it will handle.

- In the advance magnetic field step, new values for magnetic field are calculated by the *main* thread.
- In the move particles step, *every* thread moves its assigned particles to new positions
- The compute current step has two parts:
  - In the first one *every* thread computes the currents generated by its particles
  - In the second part, the *main* thread collects the results, sums them up, and performs a smoothing operation on the values.
- In the last step, new values for the electric field are computed by the *main* thread

The multithread approach allows all threads to share memory, so the fields do not need to be copied and sent to them.

TABLE 1 summarizes the main areas used for storing data: what thread has access and how it uses it.

TABLE 1. MEMORY LAYOUT

Data stored	Remarks
Particle positions and velocities	Each thread works on a disjoint area
Magnetic field	Changed only by main thread
Electric field	Changed only by main thread
Current densities	Each thread has its own area where it collects currents generated by the move of the particles it manages

III. EXPERIMENTS AND RESULTS

A series of tests were performed on a NVidia DGX-1 server using only the CPUs. The server is equipped with 2 Intel Broadwell-EP 2200MHz processors (40 cores each) and 16 32G DDR4 2133 modules resulting in 511GB RAM available and runs Ubuntu Linux version 18.04 LTS. The machine also has GPU accelerators, but they were not used.

The program was written in C++ and compiles with g++ versions 4.8.5 and above. It was packed in a docker [18] container, as this is the standard way to run on Nvidia DGX server. Among the program runs executed, we present below the ones which involve a larger number of particles (tenths of million).

A. *A type A simulation execution*

Simulations of type A were carried out with 85,737,500 ion-electron pairs and 1 million cells.

Fig. 3 shows the evolution of the simulation time taken to execute 100 iterations of a Type A simulation.

From the graph one can see that the time decrease with increasing number of threads, but this increase is much smaller past 8 threads for this problem size. That occurs because the parallel part is reduced, and the sequential parts dominate the time.

B. *A type B simulation execution*

First, simulations of type B were carried out with 85,737,500 ion-electron pairs and 1 million cells. The number of threads used was from 1(sequential) to 64 – they were running on 1 to 64 cores of the physical machine. Fig. 4 shows the time taken for executing 5 iterations. The increased time at 50 and 100 iterations is owed to saving the data in files. These show that the save operation – which is not carried out in parallel, and involves heavy operating system I/O, as files are large – are costly. (The particle data needs about 8GB to store). Fig. 5 shows the evolution of the simulation time taken to execute 100 iterations of a Type B simulation

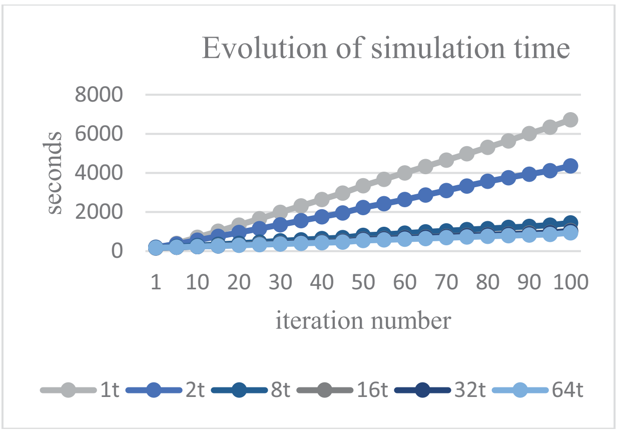

Fig. 3. The evolution of the simulation time for a Type A simulation with 85,737,500 ion-electron pairs and 1 million cells.

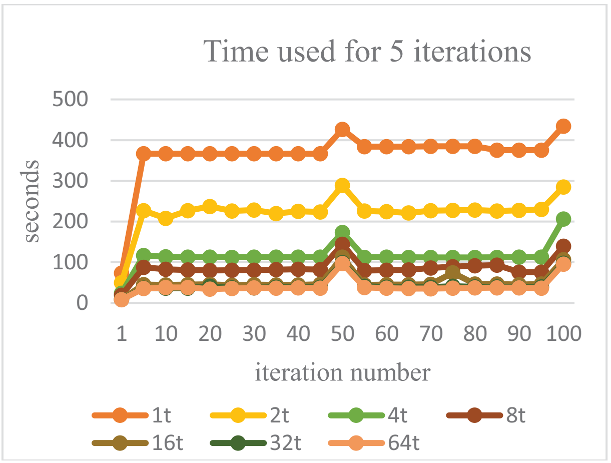

Fig. 4. The time consumed for 5 iterations for a Type B simulation with 85,737,500 ion-electron pairs and 1 million cells

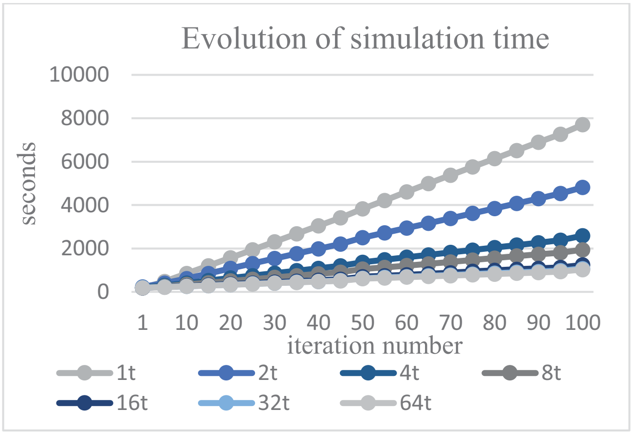

Fig. 5. The evolution of the simulation time for a Type B simulation with 85,737,500 ion-electron pairs and 1 million cells

One can notice that the increase in performance is much smaller past 8 threads, for the reason stated in section III.A.

The average speedup is shown in table TABLE 2 for a type A simulation and in TABLE 3 for a Type B simulation: 85,737,500 ion-electron pairs and 1 million cells.

TABLE 2. AVERAGE SPEEDUP FOR A TYPE A SIMULATION

# threads	2	8	16	32	64
Speedup	1.19	1.5	2.41	3.53	5.68

TABLE 3. AVERAGE SPEEDUP FOR A TYPE B SIMULATION

# threads	2	4	8	16	32	64
Speedup	1.52	2.73	3.54	5.35	6	6.2

By comparing the two results one may think that there is poor performance. Actually, the workload is too small for the program to behave well. The next section demonstrates it.

C. A type B simulation with increased number of particles

In order to investigate the behavior of the multithreaded implementation for higher number of particles, we used a Type B simulation with 16 threads. The results are

TABLE 4. TIME INCREASE VS NUMBER OF PARTICLES

summarized in TABLE 4.

Increase factor	**Number of ion-electron pairs**	**Average time multiplication factor**
4	43,897,600	1.192857105
8	87,795,200	1.495103226
12	175,590,400	2.407220146
16	351,180,800	3.52546859
32	702,361,700	5.684873724

Increase factor represents the ratio between the number of ion-electron pairs in the simulation and the number of used in the base simulation (10,974,400). One can notice that the application scales well, a 16 times increase in the work resulting in a less than four times increase in the total processing time. The cause of this is that the balance between the sequential part and the parallel part of the program improves. corresponds to an increase factor of 4, x8 to increase factor of 8 etc.

Fig. 6 shows how the time taken for 5 iterations evolves during a simulation with increased number of particles(given in TABLE 4). The x4 in the legend.

Fig. 7 shows the evolution of the simulation time, sampled every 5 iterations. There are variations caused by the workload of the machine (something else ran in parallel)

We found one implementation inspired by the TRISTAN code, in High Performance Fortran, from 2008, described in [19]. The authors used from 1,200,00 to 3,500,00 ion-electron pairs and 781,625 cells. Their benchmark results look better than ours, but the number of particles is much smaller, and it is not clear how that application would scale for large number of particles (billions and further).

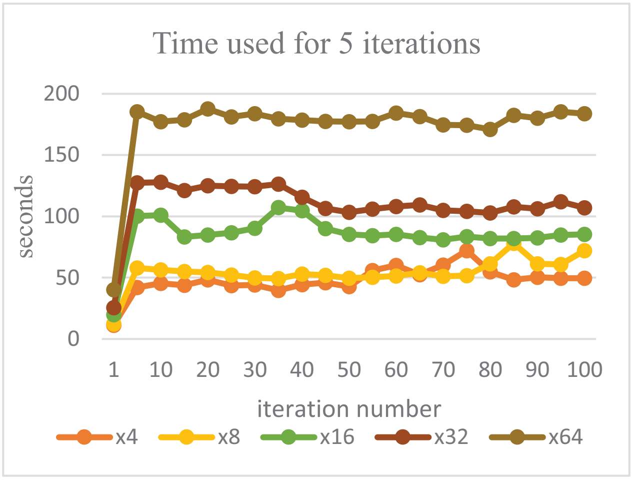

Fig. 6. Time for 5 iterations in a Type B simulation with increased number of particles (cf. TABLE 4)

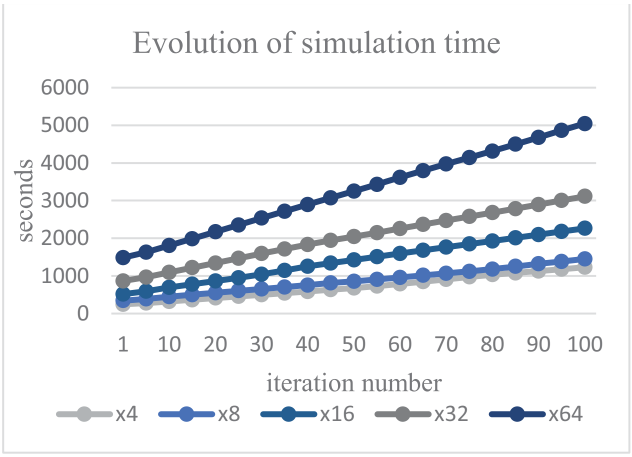

Fig. 7. The evolution of the simulation time used in a Type B simulation with increasing number of particles (cf. TABLE 4)

IV. CONCLUSIONS AND FUTURE WORK

A multithreaded solution for the kinetic model for PIC plasma simulation was successfully developed. The parallel performance of the implementation is good, considering the basic algorithm it relies on. It scales well for larger particle amounts.

A small improvement in the running time could be obtained by allowing the advance magnetic field (1) step of Fig. 2 to be executed by the first thread which reaches that point of the execution, removing the barrier before it, and synchronize after that step.

The main disadvantage of our approach is that the size of the simulation is limited by the available memory on the single machine which runs it.

To overcome this limitation, a message-passing system interface (MPI) implementation is under development. Due to the use of communication among processes, this will add, unfortunately, a supplemental time, depending on the size of communicating data.

ACKNOWLEDGMENT

This work was supported by the Romanian Ministry of Research and Innovation - PCCDI grant, Project no. 18 PCCDI/2018. Gabriel Voitcu and Marius Echim also acknowledge support from the Romanian Ministry of Education and Research through the Core Program LAPLAS VI/2020.

References

[1] R. W. Hockney and J. W. Eastwood, Computer simulation using particles, Bristol: Hilger, 1988, 1988.

[2] C. K. Birdsall and A. B. Langdon, Plasma physics via computer simulation, New York: Taylor and Francis, 1991.

[3] G. Voitcu and M. Echim, "Tangential deflection and formation of counterstreaming flows at the impact of a plasma jet on a tangential discontinuity," *Geophysical Research Letters,* vol. 44, p. 5920–5927, 6 2017.

[4] G. Voitcu and M. Echim, "Crescent-shaped electron velocity distribution functions formed at the edges of plasma jets interacting with a tangential discontinuity," *Annales Geophysicae,* vol. 36, pp. 1521-1535, 11 2018.

[5] G. Voitcu and M. Echim, "Transport and entry of plasma clouds/jets across transverse magnetic discontinuities: Three-dimensional electromagnetic particle-in-cell simulations," *Journal of Geophysical Research: Space Physics,* vol. 121, p. 4343–4361, 5 2016.

[6] F. Plaschke, H. Hietala, M. Archer, X. Blanco-Cano, P. Kajdič, T. Karlsson, S. H. Lee, N. Omidi, M. Palmroth, V. Roytershteyn, D. Schmid, V. Sergeev and D. Sibeck, "Jets Downstream of Collisionless Shocks," *Space Science Reviews,* vol. 214, p. 81, 2018.

[7] T. Karlsson, E. Liljeblad, A. Kullen, J. Raines, J. Slavin and T. Sundberg, "Isolated magnetic field structures in Mercury's magnetosheath as possible analogues for terrestrial magnetosheath plasmoids and jets," *Planetary and Space Science,* vol. 129, 6 2016.

[8] K.-I. Nishikawa, J. T. Frederiksen, Å. Nordlund, Y. Mizuno, P. E. Hardee, J. Niemiec, J. L. Gómez, A. Pe'er, I. Duțan, A. Meli and et al., "Evolution of Global Relativistic Jets: Collimations and Expansion with kKHI and the Weibel Instability," *The Astrophysical Journal,* vol. 820, p. 94, 3 2016.

[9] B. Di Martino, S. Briguglio, G. Vlad and P. Sguazzero, "Parallel PIC plasma simulation through particle decomposition techniques," *Parallel Computing,* vol. 27, no. 3, pp. 295-314, 2001.

[10] J. Qiang, M. A. Furman and R. D. Ryne, "A parallel particle-in-cell model for beam–beam interaction in high energy ring colliders," *Journal of Computational Physics,* vol. 198, no. 1, pp. 278-294, 2004.

[11] J. Qiang and X. Li, "Particle-field decomposition and domain decomposition in parallel Particle-field decomposition and domain decomposition in parallel," *Computer Physics Communications,* vol. 181, no. 12, pp. 2024-2034, 2010.

[12] Y. Barsamian, S. Hirstoaga and É. Violard, "Efficient Data Layouts for a Three-Dimensional Electrostatic Particle-in-Cell Code," *Journal of Computational Science,* vol. 27, pp. 345-356, 2018.

[13] Y. Barsamian, Charguéraud, A. and A. Ketterlin, "A Space and Bandwidth Efficient Multicore Algorithm for the Particle-in-Cell Method," in *Parallel Processing and Applied Mathematics: 12th International Conference (PPAM)*, Cham, 2018.

[14] O. Buneman, "Computer Space Plasma Physics, Simulation Techniques and Software," H. Matsumoto and Y. OKamura, Eds., Terra Scientific, 1993, p. 67.

[15] G. Voitcu, "Kinetic simulations of plasma dynamics across magnetic fields and applications to the physics of planetary magnetospheres," Bucharest, 2014.

[16] J. Villasenor and O. Buneman, "Rigorous charge conservation for local electromagnetic field solvers," *Computer Physics Communications,* vol. 69, pp. 306-316, 3 1992.

[17] K. S. Yee, "Numerical solution of initial boundary value problems involving Maxwell'sequations in isotropic media," *IEEE Transactions on Antennas and Propagation,* Vols. AP-14, pp. 302-307, 1966.

[18] D. Merkel, "Docker: Lightweight Linux Containers for Consistent Development and Deployment," *Linux Journal,* vol. 2014, 3 2014.

[19] D. Cai, Y. Li, K.-I. Nishikawa, C. Xiao, X. Yan and Z. Pu, "Parallel 3-D Electromagnetic Particle Code Using High Performance FORTRAN: Parallel TRISTAN," 2008, pp. 25-53.