Annales UMCS Informatica AI 5 (2006) 201-209

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Parallelization of the 'Particle-in-Cell' (PIC) density calculations in plasma computer simulations

Marcin Brzuszek[1*], Marcin Turek[2], Juliusz Sielanko[1]

[1]Institute of Computer Science, [2]Institute of Physics, Maria Curie Sklodowska University, Pl. M. Curie-Skłodowskiej 1, 20-031 Lublin, Poland

Abstract
The paper deals with the problem of parallelized Particle-In-Cell charge density calculations used in computer plasma simulation. The dependencies between the execution time and the simulation parameters such as the number of 'macro particles', plasma density, particular charge distribution technique and the number of processing units are presented. The local computer cluster and MPI standard have been used in order to parallelize calculations.

1. Introduction

Computer simulations of plasma behaviour in ion sources or TOKAMAKs are still a great challenge for programmers, despite the constantly growing processing power of available computer hardware. This is due to the complexity of physical processes to be taken into consideration when making simulation models.

One of the methods employed e.g. for computing the trajectories of charged particles in the electromagnetic field in plasma simulations is the Particle-In-Cell (PIC) method. The essence of PIC method [1] is using the computational particles (so called 'macro particles' or simply 'particles' in the next part of the paper) representing a large number of real particles of the same kind (electrons or ions). The charge of such a macro particle has to be distributed among spatial mesh representing computational area. There are many factors in simulations to be taken into account, these are: plasma density, number of computational particles, the mesh cell size, the boundary conditions and so one. All the factors have an influence on the results of calculations, but also affect the time of calculations [2].

However, despite the simplifying assumptions of PIC method, one needs to follow – in the case of 3-D calculations the trajectories of even hundreds of

[*]Corresponding author: e-mail address: Marcin.Brzuszek@umcs.lublin.pl

millions macro particles. The consequence of using such a huge number of computational particles is long duration time of those simulations (up to weeks), even if the fastest available computing units are used. For that reason, the Particle-In-Cell method is usually run in parallel mode. The division of calculations into subsets of tasks is done either by domain (spatial) decomposition or by particle decomposition, depending on the problem characteristics and the number and 'grid topology' of available computing units. Sometimes the mixed approach is implemented. Some parts of simulated process are parallelized more efficiently by particles decomposition, whilst in other case the spatial decomposition technique is more adequate.

In order to determine particle trajectories affected by spatial charge it is necessary to carry out charge density calculation procedure every time step. The scheme of that procedure is shown in figure 1. In the entry state of that procedure the spatial coordinates of every particle as well as its kind are given. As a result the three dimensional array containing calculated charge density is returned.

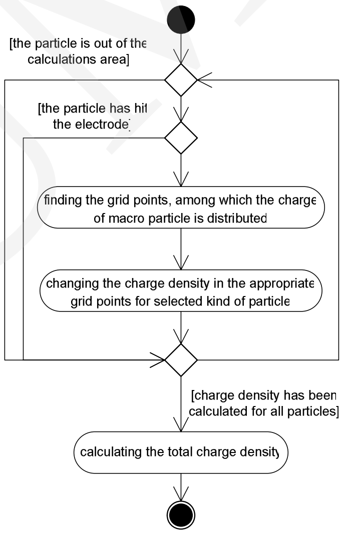

[the particle is out of the calculations area]
[the particle has hit the electrode]
finding the grid points, among which the charge of macro particle is distributed
changing the charge density in the appropriate grid points for selected kind of particle
[charge density has been calculated for all particles]
calculating the total charge density

Fig. 1. The diagram for the density of charge calculation procedure [3]

The heart of that procedure is particle charge distribution among appropriate spatial grid points. The charge assigned to a single macro particle is given by:

$$Q_p = N_{rp} \cdot e\,,$$

where N_{rp} is the number of real particles represented by one macro particle and e is the electron charge. It depends on the plasma concentration and may be calculated from the equation:

$$N_{rp} = \frac{n \cdot V_t}{N_m},$$

where n is the plasma density, ranging within 10^9-10^{11} cm^{-3}, N_m is the number of macro particles in the volume V_t.

The so called *effective charge,* which is used in the Nearest Grid Point method for the charge density distribution calculation is:

$$Q_{cf} = \frac{N_{rp} \cdot e}{V_{cell}} \text{ [C/cm}^3\text{]},$$

where V_{cell} is the volume of single cell of the grid system.

Increasing the plasma concentration entails increasing the value of Q_{cf}. Using an insufficient number of macro-particles may cause significant irregularities of charge density distribution. This may lead up to unwanted effect of non-physical electrostatic potential fluctuations. One way to avoid this is the increasing number of computational particles (decreasing Q_{cf}), the other is using better approximation of charge density distribution scheme.

As it has been said before, the paper presents the results of measurements of the execution time (CPU) for different charge density distribution methods (Nearest Grid Point, Cloud-In-Cell), plasma concentration, the number of macro particles and the number of processing units.

The simulation code is written in Fortran 90 and parallelised using Message Passing Interface (MPI), which is a standard tool for parallel programming with distributed memory, based on message passing [4]. The MPI library is a set of functions and procedures that enable sending and receiving messages by processes running on different processing units (usually one process per one processor). This is the way to synchronize processes and exchange data.

All presented results of the calculation were obtained using local (built in Lublin) PC-cluster, a part of CLUSTERIX (National Cluster of Linux Systems) project (new generation meta-cluster based on Globus Toolkit 3.2 software with own data management and task scheduling solutions). The cluster in Lublin consist of 12 dual-processor SMP nodes: 64-bit Intel Itanium 2; 4GB RAM; with Gigabit Ethernet connection; Debian Sarge operating system; Intel Fortran Compiler for IA64 architecture; MPICH 1.2.6 implementation of MPI and OpenPBS batch job queuing system [5].

2. Density calculations in parallel mode

For parallel calculation of charge density distribution, the particles decomposition approach is used [6]. Every process has its own copies of charge distributions stored in three-dimensional arrays corresponding to a spatial mesh. The charge density distribution procedure returns several arrays: one for every kind of particle and additional one for total charge density. However, every process is only responsible for its subset of particles. One of the advantages of this approach is even load balancing. The disadvantage, however, is that the copies of the data (charge density distributions) have to be stored separately for each processor, and it is the cause of high memory consumption. Another advantage of this approach is that it is the simplest way to recode existing sequential programs into a parallel form, as can be seen in figure 2. In the entry state the set of macro particles is divided into *N* subsets, where *N* is the number of processors. Every process has information about the kind and position, only for the particles from its own subset.

Fixing the grid points, among which the charge of macro particle is distributed depends on the selected scheme. The Nearest Grid Point (NGP) method assumes that the whole charge value of particle is assigned to the nearest grid point – see Fig. 3a. In the Cloud-In-Cell(2) method the charge value of single macro particle is distributed among all 8 grid points (in the case of 3D calculations). In Fig.3b – for simplifications – such scheme of the particle distribution is shown for the 2D approximation. The Cloud-In-Cell(3) method assumes even better charge distribution. The charge value is distributed among 27 grid points.

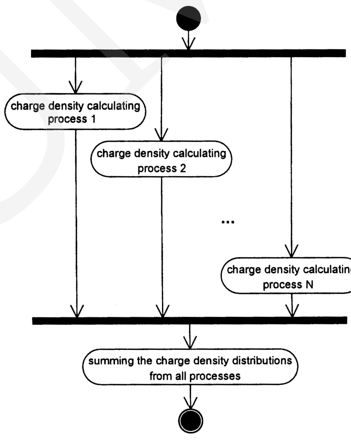

charge density calculating process 1
charge density calculating process 2
...
charge density calculating process N
summing the charge density distributions from all processes

Fig. 2. The diagram of the parallel form of charge density distribution calculation procedure

In both Cloud-In-Cell methods, the division is described by weight coefficients that depend on the distance between the particle and the considered grid points. The details of the CIC(3) algorithm could be found in [7]. In Fig.3b the scheme of such distribution simplified to only 2-D case is shown. It should be mentioned that in every case weight coefficients are normalized to Q_{cf} value.

After the charge density calculations are finished by all processes, the collective communication routine MPI_REDUCE with the parameter MPI_SUM is called. In other words, the charge densities are summed and the result is gathered to one (say 'master') process. This is the only one moment in the whole procedure of charge density distribution calculation, where processes have to communicate with each other and exchange data.

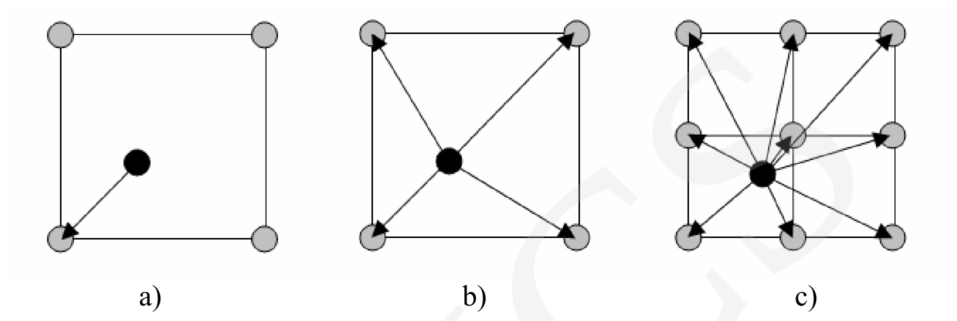
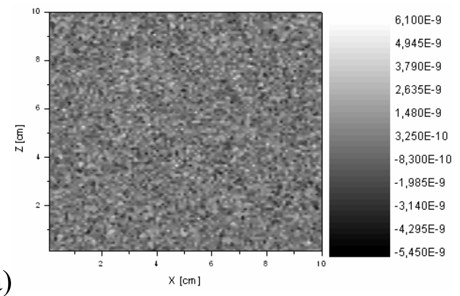


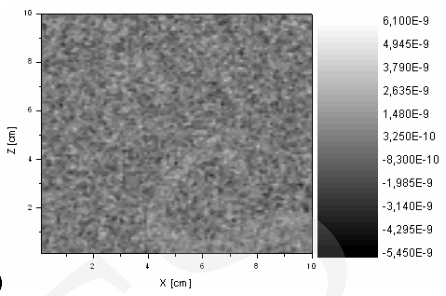
Fig. 3. The simplified schemes of charge distribution among grid points for NGP – *a*, CIC(2) – *b* and CIC(3) – *c* methods

3. Results of calculations

The charge density distribution simulations were carried out for the cubic computational area of 10 cm side, represented by 100×100×100 grid points. The initial positions of particles were produced by a random number generator. It should be mentioned that the standard random number generator function of Fortran 90 compiler was used all over the code. The kind of macro particles was also determined using a random number generator. However, the assumption was made that there is approximately the same number of positive and negative charged particles. Under such assumption, the Q_{cf} values of oppositely charged particles assigned to one grid point compensate each other to a large degree. The draw of the random initial coordinates and the kind of particles result in particular space distribution. The homogeneity and deviations of the distribution from zero (which is an expected value) depends on the number of macro particles and charge distribution method. Having in mind random number generator imperfections, the situation, when the deviations from zero are as small as possible is expected. In the next two figures, the cross-sections of selected the charge density distributions (in C/cm^3), as an arithmetic mean from five middle layers of mesh, are shown. The picture 4 illustrates differences in distribution homogeneity depending on the selected method, and picture 5 presents the same, depending on the number of macro particles.

As can be seen, the use of both more accurate charge distribution methods and a greater number of macro particles give more homogeneous charge density distribution. However, the presented figures suggest that using a larger number of macro particles influences the homogeneity stronger than using a more accurate charge assignment scheme. The maximum deviations of averaged (over five middle grid layers) charge density versus the number of macro particles for three methods of charge distribution are shown in Fig. 6.


a)


b)

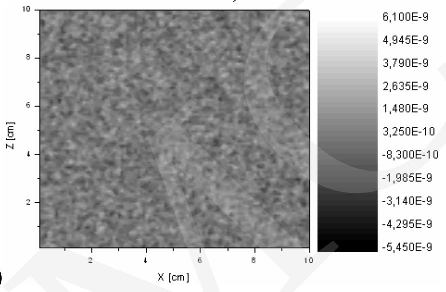
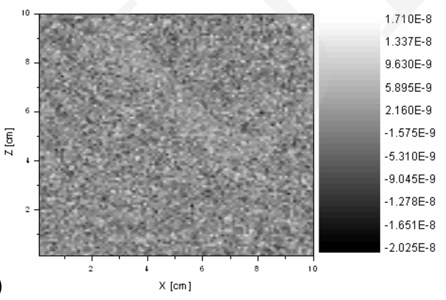
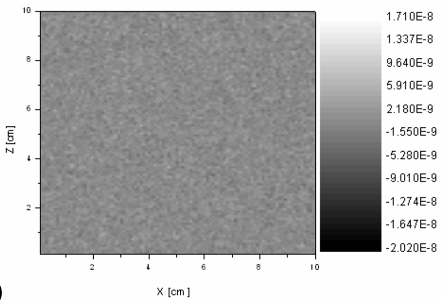

c)

Fig. 4. The cross-sections of charge density distributions (in C/cm^3) calculated by different methods for 20 mln of macro particles: a) NGP, b) CIC (2), c) CIC (3)


a)


b)

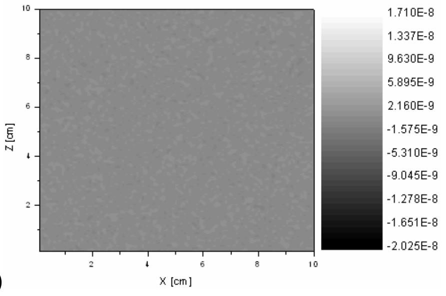

c)

Fig. 5. The cross-sections of charge density distributions (in C/cm^3) calculated by the NGP method for different number of macro particles: a) 2 mln, b) 20 mln, c) 200 mln

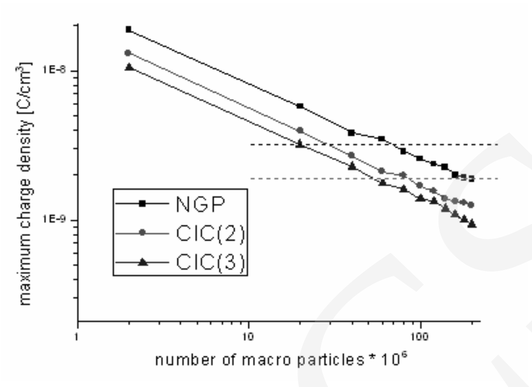


Fig. 6. Maximum deviations from the zero of charge density averaged over five middle grid layers versus the number of macro particles for three methods of charge distribution

One can see that using the CIC(2) instead of the NGP method increases the homogeneity by about 29-39%, in the whole range of macro particles number. Using the CIC(3) method provides even better results, namely 40 to 50%. Comparing graphs in figure 6 with the dependencies of charge density calculations execution time on the number of processors (Fig.7 and 8), and taking into special account time of calculations, one can conclude that the strategy of using the NGP method and a greater number of macro particles is the most effective.

For example, using NGP and 100 mln of macro particles gives better results (i.e. more homogeneous distributions) in less time than using the CIC(3) method and 20 mln of macro particles.

Analyzing the graphs of time of charge density calculations by NPG method versus the number of processors illustrated in Fig.8, one can conclude that using a double number of processors reduces execution time by approximately the factor of 2. This is especially easy to see for a larger number of macro particles, but even for 2 mln of macro particles some kind of regular behaviour can be observed. When the processor number equals 2, 4, 8 or 16 (powers of 2) the execution times are much less than those for the neighbouring processor numbers. This is probably due to the fact that collective communication algorithms (e.g. used in MPI_REDUCE) are more effective for such processor numbers.

Figs. 4, 5 and 6 present the results for the plasma concentration equal to 10^{11}[cm^{-3}]. For a comparison, Fig. 9 shows maximum deviations for charge density averaged over five middle layers for the two plasma concentration values – 10^{10}[cm^{-3}] and 10^{11}[cm^{-3}].

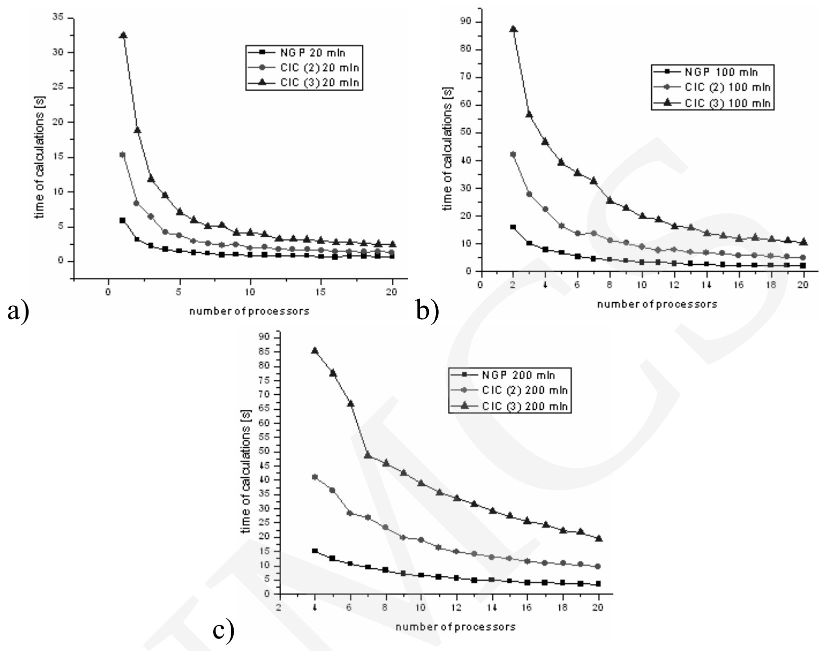


Fig. 7. Execution time of charge density calculations as a function of the number of processors used for the parallel run for different methods of charge distribution and different numbers of macro particles: a) 20 mln, b) 100mln, c) 200 mln

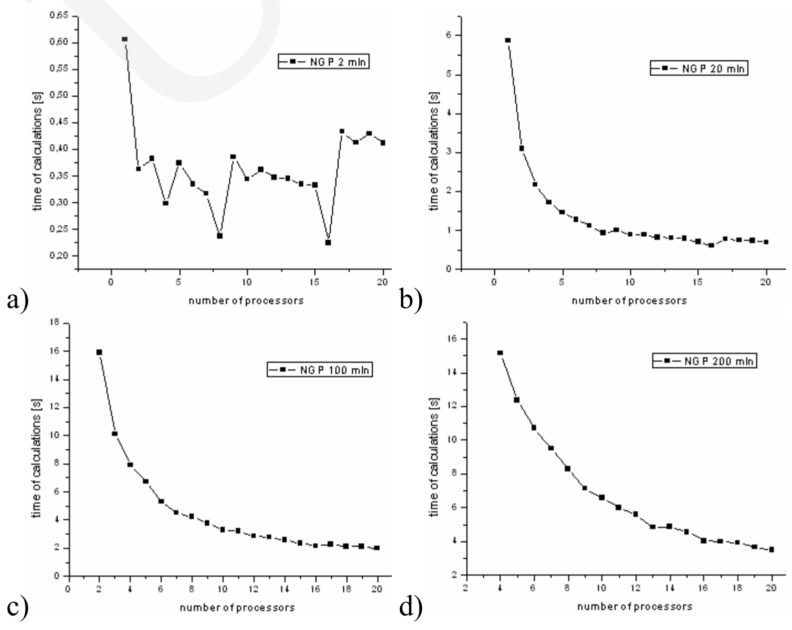


Fig. 8. Time of charge density calculations versus the number of processors used for the parallel run, using the NGP method and a different number of macro particles: a) 2 mln, b) 20 mln, c) 100 mln, d) 200 mln

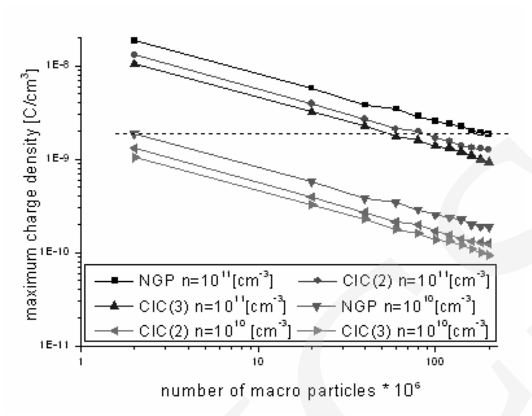


Fig. 9. Comparison of maximum deviations of charge density averaged over five middle grid layers for the two plasma concentration values (10^{10}[cm^{-3}] and 10^{11}[cm^{-3}])

It can be easily seen, that the average deviation typical of the case of 10^{10} [cm^{-3}] and 2 mln of macro particles is achieved for 10^{11}[cm^{-3}] plasma concentration when using 200 mln of particles. The influence of plasma concentration on the execution time may be considered for 10^{11}[cm^{-3}] plasma concentration simulation using 20 processors and 200 mln of macro particles. For this case the CPU time is about 6 times longer than one-processor simulation for the same plasma concentration and using only 2 mln of macro particles.

Conclusions

Computer simulations of such complex physical processes as plasma behaviour in ion sources require careful analysis of each program module. Mutual dependencies between those modules have to be recognized, in order to reduce execution time of the whole code as much as possible. One of those modules is the Particle-In-Cell charge density distribution procedure. The main parameter, that strongly influences execution time, is plasma concentration. This paper shows that the best solution is using the simplest charge density distribution technique and a large number of macro particle. Peculiarity of the problem allows easy parallelization and reduction of simulation execution time.

References

[1] Hockney R., Eastwood J., *Computer Simulation Using Particles*, Moscow, Mir, (1987).
[2] Sielanko J., Turek M., Tanga A., Annales UMCS-Informatica, 2 (2004) 251.
[3] Sielanko J., Muszyński M., Electron Technology, 30(4) (1997) 352.
[4] http://www-unix.mcs.anl.gov/mpi/
[5] http://clusterix.pcz.pl/
[6] Foster I, *Designing and Building Parallel Programs*, Addison-Wesley, (1996).
[7] Decyk V.K., Karmesin S.R., de Boer A., Liewer P.C., *Optimization of Particle-in-Cell Codes on RISC Processors*, Computers in Physics, 10 (1996) 290.