Parallelization of a Vlasov solver by communication overlapping

Eric VIOLARD (Speaker),
LSIIT-ICPS, Université Louis Pasteur
Boulevard S. Brandt, F-67400 Illkirch, France.
violard@icps.u-strasbg.fr, tel (+33) 3 90 24 45 52 / fax : (+33) 3 90 24 45 47

Francis FILBET,
IRMA, Université Louis Pasteur
7 rue René Descartes, F-67084 Strasbourg, France
filbet@math.u-strasbg.fr, tel (+33) 3 90 24 02 06 / fax : (+33) 3 90 24 45 47

Keywords: Parallel MPI application, plasma physics, overlapping, modelisation.

1 Introduction

This paper is devoted to the numerical simulation of problems in Plasma Physics and particle beams propagation. The main interest of this topic is the study of controlled fusion, which seems to be a promising solution for future energy production. Roughly speaking, there exist two approaches to realize controlled fusion. The first one is magnetic confinement, where the plasma or charged particles are contained in a finite region using magnetic fields. Charged particles travel in helical paths around the magnetic field lines and this confines their motion. The second method is inertial confinement, which consists in producing nuclear fusion by shooting at a Deuterium and Tritium target with a particle or laser beam.

A model which can be used in many cases for the study of plasma as well as of beam propagation is the Vlasov equation coupled with the Maxwell or Poisson equations to compute the self consistent fields. It describes the evolution of a system of particles under the effects of external and self-consistent fields. The unknown $f(t, x, v)$, depending on the time t, the position x, and the velocity v, represents the distribution of particles in phase space for each species. The numerical resolution of the Vlasov equation is usually performed by Particle-In-Cell (PIC) methods which approximate the plasma by a finite number of particles. Trajectories of these particles are computed from characteristic curves given by the Vlasov equation, whereas self-consistent fields are computed on a mesh of the physical space. This method yields satisfying results with a relatively small number of particles. However, it is well known that the numerical noise inherent to the particle method becomes, in some cases, too important to get an accurate description of the distribution function. Moreover, the numerical noise only decreases in $1/\sqrt{N}$, when the number of particles N is increased.

To remedy this problem, methods discretizing the Vlasov equation on a mesh of phase space have been proposed. Among them, the finite volume type method (or flux balance method) consists in averaging the distribution function on phase space discrete volumes. These unknowns are updated by considering incoming and outgoing fluxes leading to mass conservation. We will consider the Positive and Flux Conservative method (PFC) [4, 5], which is not only conservative, but also preserves the positivity and the maximum value of the distribution function. The scheme was implemented up to third order accuracy. Thus, discretizations by such methods are really interesting to get a good description of the plasma or particle beam evolution. Unfortunately, these methods require a big amount of memory and computational time, because the unknowns are computed on phase space meshes. Indeed, even if we restrict ourselve to the study of particle beams in the four dimensional transverse plane *i.e.* the orthogonal direction to the beam propagation direction (neglecting variation in (z,v_z)), a reasonable choice to get an accurate approximation of the particle beam is to consider at least 100 points in each direction of the four dimensional

phase space (x,y,v_x,v_y), which means 100^4 points (1 GB of memory). In this case, it is necessary to develop parallel algorithms.

The outline of the paper is as follows : We shall first recall the Vlasov equation and the PFC schemes which is used to approximate the solution. Then, we present a first parallel algorithm in order to reduce the computational cost of the approximation of the Vlasov equation in 4D or 6D. A second algorithm is based on overlapping of communications by calculation, which is enabled to highly reduce the computational time. Finally, we present numerical results illustrating the efficiency of the new parallel algorithm.

2 The Positive and Flux Conservative (PFC) method

The evolution of the density of particles $f(t,x,v)dx\,dv$ in the phase space $(\mathrm{x},\mathrm{v}) \in \mathbb{R}^d \times \mathbb{R}^d$, $d = 1,..,3$, is given by the normalized Vlasov equation,

$$\frac{\partial f}{\partial t} + \mathrm{div_x}(\mathrm{v}\, f) + \mathrm{div_v}(E(t,\mathrm{x})\, f) = 0. \tag{1}$$

where the self electric field E is computed using the Poisson equation, *i.e.*

$$E(t,\mathrm{x}) = -\nabla_{\mathrm{x}}\phi(t,\mathrm{x}), \quad -\Delta_{\mathrm{x}}\phi(t,\mathrm{x}) = \rho(t,\mathrm{x}), \tag{2}$$

where the charge density ρ is defined by

$$\rho(t,\mathrm{x}) = \int_{\mathbb{R}^d} f(t,\mathrm{x},\mathrm{v})d\mathrm{v}. \tag{3}$$

The time discretization of (1) is based on the following splitting algorithm on $\Delta t = [t^n, t^{n+1}]$ knowing an approximate solution f^n at time t^n

1. Solve a free transport equation on Δt

$$\begin{cases} \dfrac{\partial f^{(1)}}{\partial t} + \mathrm{div_x}\left(\mathrm{v}\, f^{(1)}\right) = 0, \\ f^{(1)}(0,\mathrm{x},\mathrm{v}) = f^n(\mathrm{x},\mathrm{v}). \end{cases} \tag{4}$$

2. Compute the electric field $E(t^{n+1/2},\mathrm{x})$ at time $t^{n+1/2}$ by substituting $f^{(1)}(\Delta t,\mathrm{x},\mathrm{v})$ in the Poisson equation and in (3).

3. Solve on Δt the equation

$$\begin{cases} \dfrac{\partial f^{(2)}}{\partial t} + \mathrm{div}_v\left(E(t^{n+1/2},\mathrm{x})\, f^{(2)}\right) = 0, \\ f^{(2)}(0,\mathrm{x},\mathrm{v}) = f^{(1)}(\Delta t,\mathrm{x},\mathrm{v}). \end{cases} \tag{5}$$

   and set $f(t^{n+1},\mathrm{x},\mathrm{v}) = f^{(2)}(\Delta t,\mathrm{x},\mathrm{v})$.

Using this procedure the algorithm boils down to solve equations (4) and (5) on a phase space mesh $(\mathrm{x},\mathrm{v}) \in \mathbb{R}^4$. To this aim, we introduce a finite set of mesh points $(\mathrm{x}_i = (x_i, y_i))_{i\in\{0,..,n_x\}}$ and $(\mathrm{v}_j = (v_{xj}, v_{yj}))_{j\in\{0,..,n_v\}}$ of the computational domain. We will denote by $\Delta x = x_{i+1} - x_i = y_{i+1} - y_i$ the space step, $\Delta v = v_{xj+1} - v_{xj} = v_{yj+1} - v_{yj}$ the velocity step and by $C_{i,j} = [\mathrm{x}_i, \mathrm{x}_{i+1}] \times [\mathrm{v}_j, \mathrm{v}_{j+1}]$ the control volume. Assume the values of the distribution function f, stored in matrix $(F^n_{i,j})_{i,j}$, are known at time $t^n = n\,\Delta t$. We find the new values at time t^{n+1} by successively solving (4) and (5) on each control volume $C_{i,j}$ from time t^n to time t^{n+1}. Using the PFC scheme described in [4], we get a first approximation $F^{(1)}_{i,j}$ of equation (4) from the values $F^n_{i,j}$

$$F^{(1)}_{i,j} = Q^{(1)}_{\Delta x,\Delta v}(F^n_{0,j}, F^n_{1,j}, .., F^n_{n_x-1,j}). \tag{6}$$

Let us mention that the computation of $F^{(1)}_{i,j}$ only depends on row values of F^n.
From these new values, we approximate the electric field on the physical space mesh $(\mathrm{x}_i)_{i\in\{0,..,n_x\}}$ from the discrete space charge

$$\rho_i^{n+1/2} = \Delta v \sum_j F^{(1)}_{i,j}$$

using a Fast Fourier Transform (FFT). Finally, we get the solution at time t^{n+1} by approximating equation (5),

$$F^{n+1}_{i,j} = Q^{(2)}_{\Delta x,\Delta v}(F^{(1)}_{i,0}, F^{(1)}_{i,1}, .., F^{(1)}_{i,n_v-1}). \tag{7}$$

In this case, the approximation of $F^{n+1}_{i,j}$ only requires column values of $F^{(1)}$.
We refer to [4] for more details about discrete operators $Q^{(1)}_{\Delta x,\Delta v}$ and $Q^{(2)}_{\Delta x,\Delta v}$, which respectively approximate equations (4) and (5) with a very good accuracy.

2.1 The parallel algorithm

This part is devoted to designing parallel algorithms to solve the Vlasov equation for each time step using the numerical method presented before. The aim is construct a parallel algorithm to reduce the computational time. One method only consists in optimizing communications, we refer for example to [6, 3], where authors describe an efficient implementation of the transposition. Another approach, which seems to be more efficient when we treat large data size, is to perform communications and computations at the same time. It is referred as computation/communication overlapping [1].

2.2 A first parallel algorithm

Let us denote by F^n the matrix $(F^n_{i,j})_{i,j}$, where i represents the index of the physical space mesh and j the index of the velocity space. We note that the big amount of work is performed during steps (6) and (7). Therefore, we will focus on the parallelization of these computations. To do this we observe, as mentionned before, that the operator $Q^{(1)}$ (resp. $Q^{(2)}$) only acts on rows of $(F^n_{i,j})$ (resp. on columns of $(F^{(1)}_{i,j})$). Thus, for the first step if data are distributed by row on each processor, then no communications are required. Similarly, for the second step a storage by column does not involve any communication at all. The link between two steps is done through a transposition of $F^{(1)}$.

Following this idea, we first present a simple parallel algorithm [2], starting with one dimensional band distribution F^n along v-direction.

Algorithm 1:

For each time step Δt

1. Compute the matrix $F^{(1)}$ using $Q^{(1)}_{\Delta x,\Delta v}$ from values F^n. The storage by rows avoids local communications for this step.

2. Distribute the matrix $F^{(1)}$ into the matrix transposed ${F^{(1)}}^T$ in order to get a one dimensional band distribution along x-direction.

3. Integrate ${F^{(1)}}^T$ to get the discrete charge density ρ and compute the electrical field E

4. Apply $Q^{(2)}_{\Delta x,\Delta v}$ to ${F^{(1)}}^T$ for solving equation (7) over a time step of Δt. The storage by columns avoids local communications for this step.

5. Redistribute matrices ${F^{n+1}}^T$ into matrix F^{n+1} in order to get a one dimensional band distribution along v-direction.

End.

In this case, we notice that with our parallel algorithm, communications only occur during the two data redistributions phases. In other words, the algorithm falls into phases of computations without any communication and phases of communications without any computation.

Improving this algorithm is particularly interesting because communication phases are very time-consuming due to the very large amount of data, which are transmitted. Indeed, communications consist in a transposition that is a global exchange of data among all processors. It is well known that this operation is one of the most demanding communications tasks.

Figure 1 illustrates the communications required during the first redistribution phase on a 4-processors architecture. Each processor holds a band of matrix F^n and assuming the band is subdivised into 4 blocks, it must send one block of data to each other processor and receive one block of data from each other processor. This transposition task is costly. It is important to improve our algorithm.

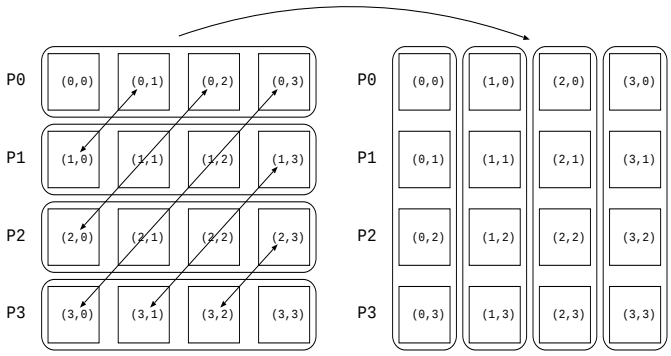

Figure 1: The data exchange pattern to transpose a data matrix on a four processors system.

2.3 Algorithm improvement

As seen previously, an interesting feature of the PFC method is that it induces a simple parallel algorithm where communications and computations are performed in well distinct phases. It makes clear that the only mechanism that must be used to significantly improve our algorithm, consists in performing some computations while some data are communicated.

Let us describe the overlapping method from a simple example, where the successive tasks are carried out by two processors P_0 and P_1. On the one hand, processor P_0 first sends some data, say D_{01}, to processor P_1 and then performs some computations, say C_0. On the other hand, processor P_1 first performs some computations, say C_1, and then waits for receiving data D_{01}. Therefore, communication of D_{01} can be clearly achieved by the system while P_0 computes C_0 and P_1 computes C_1.

This mechanism can be used during steps (1)-(2) and (4)-(5) of **Algorithm 1** in order to exchange blocks of data while others are computed.

Let us first focus on steps (1)-(2) of **Algorithm 1**, we will denote by p the number of processors which are used. Then, we subdivide matrix $F^{(1)}$ into $p \times p$ blocks $(B_{k,l})_{k,l}$ (see Fig. 2.3). Processor k first computes blocks $B_{k,l}$, $l = 0, .., p-1$ and then sends block $B_{k,l}$ to processor l, for all $l \in \{0, .., p-1\} \setminus \{k\}$. These tasks can be performed in any order provided a block is computed before it is send. The overlapping of communications depends on the order in which these tasks are carried out. To get the best overlapping, the send of a block must be initiated just after its computation and computation of block $B_{k,k}$ be performed in last. Finally, steps (4)-(5) can be achieved by following the same idea.

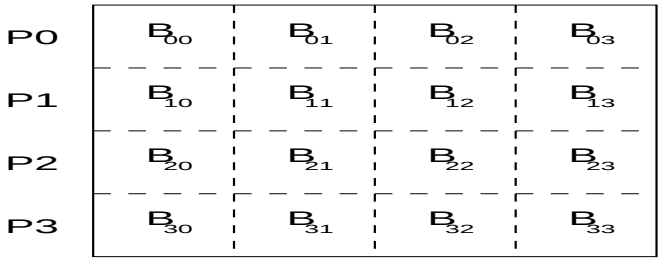

Figure 2: Block subdivision of the data matrix $F^{(1)}$.

Algorithm 2:

For each time step Δt; for each processor k

1. For each $l = 1, .., p-1$,

   - compute the block $B_{k,(k+l)\%p}$ corresponding to matrix $F^{(1)}$ using $Q^{(1)}_{\Delta x,\Delta v}$ from values F^n.
   - send the block $B_{k,(k+l)\%p}$ to processor $(k+l)\%p$. This communication will be overlapped by the computation of the next block $B_{k,(k+l+1)\%p}$.
   - initialize the reception of block $B_{(k-l)\%p,k}$ from processor $(k-l)\%p$.

   End.
   Compute block $B_{k,k}$ corresponding to matrix $F^{(1)}$.

2. Received blocks are stored in matrix ${F^{(1)}}^T$, which is integrated to get the discrete charge density ρ. The electrical field E is computed from ρ.

3. For each $l = 1, .., p-1$,

   - compute the block $B_{k,(k+l)\%p}$ corresponding to matrix ${F^{n+1}}^T$ using $Q^{(2)}_{\Delta x,\Delta v}$ from values ${F^{(1)}}^T$.
   - send the block $B_{k,(k+l)\%p}$ to processor $(k+l)\%p$. This communication will be overlapped by the computation of the next block $B_{k,(k+l+1)\%p}$.
   - initialize the reception of block $B_{(k-l)\%p,k}$ from processor $(k-l)\%p$.

   End.
   Compute block $B_{k,k}$ corresponding to matrix ${F^{n+1}}^T$.

End.

We used MPI to implement algorithms, presented before, where non blocking sends and receives are done by functions `MPI_Issend` and `MPI_Irecv`. We choose the synchronous communication mode in order to avoid temporary copy of data.

3 Numerical results

This part is devoted to comparison of the two parallel algorithms for the Vlasov equation. We consider the numerical simulation in the transverse plane of beam propagation using three different meshes of phase space. Calculations are done on a SGI O2K computer with 64 processors. First, we only use 16 points in each direction, which means that the total number of points of the matrix F^n is 16^4. We compute the solution with 2, 4, 8 and 16 processors. Then, we consider the same

physical problem using more refined meshes of 32^4 and 64^4 total number of points. In Table 1, we present the total computational time using **Algorithm 1**, whereas results obtained by the **Algorithm 2** are listed in Table 2.

nbprocs	16^4 points	32^4 points	64^4 points
2	14	221	4154
4	8.41	94	2040
8	5.75	50	1011
16	2.43	32	531
32	X	20	283

Table 1: Total time of computation using **Algorithm 1**.

nb procs	16^4 points	32^4 points	64^4 points
2	8.4	123	4124
4	5.75	62.5	1528
8	2.81	35.5	558
16	2.06	26	308
32	X	18	207

Table 2: Total time of computation using **Algorithm 2**.

From these results, we can observe that the second algorithm based on overlaping of communications by calculation always reduces the computational cost. Moreover, when the data size is increasing, the best efficiency is reached for larger and larger number of processors. For example, with 32^4 points, the second algorithm is the most efficient with only 4 processors, but using 32^4, the maximum efficiency is reached for 8, etc. However, for a fixed number of unknowns, the performance of two algorithms is very closed when we use a very large number of processors. Indeed, for large number of processors, many messages of small size are sent, which induces high communication cost.

4 Conclusion

In this paper, we proposed a new algorithm based on communication overlapping by calculation. Numerical results show the efficiency of this method compared to a basic algorithm. Nevertheless, due to the high number of exchanges, the efficiency highly depends on data size and on number of processors. In fact, we can conjecture that when the number of processors is very large, the first algorithm without overlapping is more efficient, because few messages are sent by combining small messages into larger ones. Thus, both algorithm can be used depending on data size and number of processors, which are available.

References

[1] L. Colombet, Parallelization of applications for homogeneous and heterogeneous processors network *PhD thesis IMAG, Grenoble* (1994).

[2] O. Coulaud, E. Sonnendrücker, E. Dillon, P. Bertrand and A. Ghizzo, Parallelisation of Semi-Lagrangian Vlasov Codes *J. Plasma Phys.* 61 (1999).

[3] C. Christara and X. Ding and K. R. Jackson An Efficient Transposition Algorithm for Distributed Memory Computers. *13th Annual International Symposium on High Performance Computing Systems and Applications (HPCS'99)* (1999), pp 435–448.

[4] F. Filbet, E. Sonnendrücker, P. Bertrand, Conservative Numerical schemes for the Vlasov equation *J. Comput. Phys.* 172 (2001), pp 166–187.

[5] E. Sonnendrücker, J. Roche, P. Bertrand and A. Ghizzo The Semi-Lagrangian Method for the Numerical Resolution of Vlasov Equations. *J. Comput. Phys.* 149 (1998), pp 201–220.

[6] P. N. Swarztrauber Transposing Arrays on Multicomputers Using de Bruijn Sequences. *J. Parallel Distributed Computing.* 53 (1998), pp 63-77.