Proteins Separation in Distributed Environment Computation

Ming Chau[1], Thierry Garcia[2], and Pierre Spiteri[2]

[1] Advanced Solutions Accelerator 199 Rue de l'Oppidum F-34170 Castelnau le Lez, France
[2] INP – ENSEEIHT – IRIT BP 7122, 2 Rue Camichel F-31071 Toulouse Cedex, France
mchau@advancedsolutionsaccelerator.com,
{thierry.garcia,pierre.spiteri}@enseeiht.fr

Abstract. In the present study we consider the parallel simulation on a peer-to-peer demonstrator for the computation of a 3D differential equations, which model the behavior of the continuous-flow electrophoresis problem. The physical model considered is constituted by the Navier-Stokes equation coupled with a convection-diffusion equation and a Laplacian equation. By using appropriate discretization of the coupled boundary value problems, we can prove the convergence of synchronous and more generally asynchronous relaxation methods. Finally parallel experiments on the peer-to-peer demonstrator are presented and analyzed.

Keywords: Parallel computing, distributed computing, continuous-flow electrophoresis, asynchronous iteration, PISO algorithm.

1 Introduction

The continuous-flow electrophoresis problem plays a major role in many fields such as medicine, agronomy and more generally in biology. The electrophoresis is used to separate biological molecules, especially proteins, based on their movement in a colloidal suspension while under the influence of an electric field. Each particle moves toward the electrode of opposite electrical polarity. The solution of this method is determined by the migration distance at the collection plane and the fineness of the filament occupied by each protein species. The numerical simulation can bring useful informations but needs a lot of high performance computation.

So the goal of the present study is then to implement on distributed architectures the solution of the 3D continuous-flow electrophoresis problem by parallel computations. Thus we consider in the sequel a mathematical problem constituted by the coupling of the Navier-Stokes equation with a potential equation describing the electric field and as many evolution convection-diffusion equations than species to separate (see [1] and [2]); these boundary value problems are defined in the three-dimensional space. Then after discretization, we have several large scale algebraic systems to solve; consequently parallel computations seem well adapted in order to reduce the elapsed time of computation. Moreover, since parallel asynchronous algorithms reduce the idle time due to synchronizations between the processors (see [3], [4] and [5]), we consider in the sequel the implementation of such methods

© Springer-Verlag Berlin Heidelberg 2011

compared to the synchronous version of the same algorithm on a distributed peer-to-peer computational environment. Using appropriate discretization schemes for the boundary value problems to solve, it can be proved by various techniques that the parallel synchronous or more generally asynchronous relaxation methods converge (see [3], [4] and [5]). In fact, by gathering several adjacent blocks, these kinds of methods can be viewed like a subdomain method without overlapping between the subdomains. Note that the considered approach is different to the one considered in [14] and [15] where the parallel Schwarz alternating method has been implemented, using a mesh with fewer points; in this case, the subdomains overlap each other. Moreover, in [14] and [15], the experiments are performed on a SMP supercomputer while, in the present study, they have been implemented on a peer-to-peer platform.

Furthermore today, the development of parallel applications on distributed heterogeneous architectures already exists since about fifteen years. It can be also observed that for some time past, the size of the problem to solve and the global complexity of the applications considered increase fast. So the distributed scientific programming is a promising solution since this distributed computations performed on supercomputers may now run on a variety of heterogeneous resources geographically distributed. Thus the development of parallel applications on platforms is a promising challenge since it allows using heterogeneous architectures; furthermore the operating of computational codes on such architecture permits ones to a great flexibility of running.

In the present study, in the context of an ANR grant, we will use the P2PDC platform developed by the LAAS Laboratory of Toulouse (France). The reader is referred to [6], [7] and [8] for more details concerning this environment. In [7] a distinct application, the obstacle problem, is solved on this platform using a projected Richardson's method; this method is different to the one considered here, i.e. the block relaxation method. Note that the concerned approach is different from MPICH Madeleine [9], middleware like BOINC [10] or OurGrid [11]. Then, the goal of the present study is to consider continuous-flow electrophoresis problem solved by a subdomain method without overlapping and performed on this demonstrator platform. So we will also compare the performance of such parallel asynchronous algorithms with the synchronous ones.

The paper is organized as follows. In section 2, we present the application target with more details. The section 3 is devoted to the numerical solution of the 3D continuous-flow electrophoresis problem; consequently appropriate discretizations of the boundary value problems involving convergence of the parallel asynchronous and synchronous algorithms are presented. In section 4 we describe the implementation of the parallel asynchronous and synchronous algorithms on the P2PDC platform. The section 5 is devoted to the presentation of the parallel experiments.

2 The Physical Problem and the Associated Model

2.1 The Physical Problem

Continuous flow electrophoresis is a process for separating protein mixture. Currently used for biological analysis, its extension to a preparative scale is difficult. Its resolution is determined by the migration distance at the collection plane and the fineness of the filament occupied by each protein species. Filament undergoes spreading due to a number of different phenomena, among which electrokinetic's and electro hydrodynamics are known to be important. On one hand, differences in migration velocity between the ionic species give rise to local variations in electrical conductivity near the protein filament. On the other hand, the local change in electrical conductivity distorts the electrical field, thus including shear stress in the liquid and creating a local flow pattern. More precisely density coupling phenomena involving thermal and solutal connection are likely induce strong instability effects. Numerical simulations can bring useful information concerning the nature of expected effects.

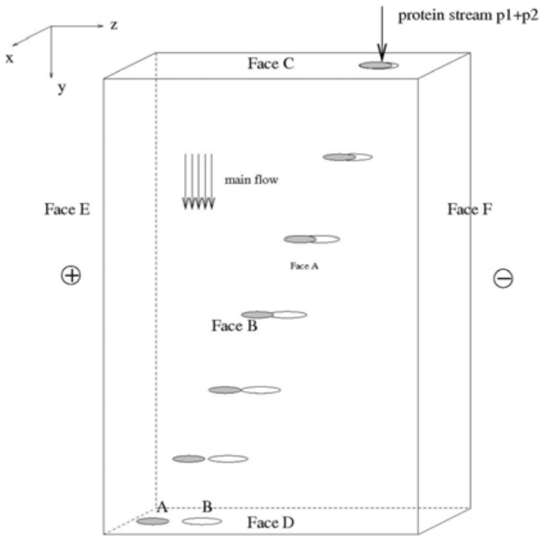

Fig. 1. The electrophoresis cell

2.2 The Physical Model

A physical model has been developed (see [1], [2]) to describe these phenomena when two or several proteins are being separated. This model consists in the coupling of three boundary value problems defined in a bounded domain included in the three dimensional space. According to the usual shape of the electrophoresis chamber, we will consider in the sequel that the domain is a parallelepiped, as we can see on Figure 1.

The coupled equations describing the considered phenomena are:

- in order to describe the flow, the Navier-Stokes equations

$$\begin{cases} \frac{\partial \vec{U}}{\partial t} + \vec{U}.\nabla\vec{U} - \eta.\Delta\vec{U} + \frac{1}{\rho}\nabla p = \varepsilon F, \Omega \times [0,T] \\ div(\vec{U}) = 0, \Omega \times [0,T] \\ C.L. \\ C.I. \end{cases} \tag{1}$$

where Ω represents the domain of the cell, η is the kinematics viscosity of the fluid, ε is the dielectric permittivity, $\vec{U} = (u, v, w)$ is the fluid velocity, p is the pressure, ρ is the volumetric mass, F is the external dielectric forces given by F = (F_i), with $F_i = \nabla(E_i\vec{E})$, i = 1,2,3, C.I. and C.L. denotes respectively the initial and the boundary conditions, the Dirichlet boundary conditions (i.e. the values of the velocity on the boundary is known) are preponderant; more precisely the values of the components of $\vec{U}$ on the faces A, B and C are given and so Dirichlet boundary conditions fulfill; on faces E and F Dirichlet boundary conditions are considered for the components u and w, while on these previous faces the normal derivative of v is null. Lastly on the face D the normal derivatives of the components of $\vec{U}$ are null ;

- in order to describe the transport of one protein, the following time dependant convection-diffusion equation is considered

$$\begin{cases} \dfrac{\partial c}{\partial t} + (\vec{U} + \mu\vec{E}).\nabla c - D\Delta c = 0, \Omega \times [0,T] \\ C.L. \\ C.I. \end{cases} \tag{2}$$

where c denotes the concentration, μ the electrophoretic mobility, D is the diffusion coefficient, and $\vec{E}$ is the electrical field. On the face A the concentration is known and then non homogeneous Dirichlet boundary conditions fulfill while on the other faces the normal derivative of c is null;

- in order to describe the electrical phenomena, a potential equation which corresponds to a generalized Laplacian is considered

$$\begin{cases} -div(K\nabla\Phi) = \Delta Q, \Omega \times [0,T] \\ C.L. \end{cases} \tag{3}$$

where K is the electrical conductivity defined by K = K_0 + c and Q is the quantity of electrical current defined by Q = Q_0 + R.θ.c, where θ is the temperature, K_0 and R are physical constants and Φ is the electrical potential related to the electrical field by the classical relation $\vec{E} = -\nabla\Phi$. The boundary conditions correspond to the homogeneous or non homogeneous Dirichlet ones.

Finally we have to solve a coupled system of non linear boundary value problems.

3 The Numerical Solution

The numerical solution of the electrophoresis problem allows two distinct steps: the discretization step and the solution of the algebraic systems derived from the discretization step. In fact after appropriate discretizations we obtain very large sparse discretization matrices arising in the algebraic systems to solve; so parallel iterative methods are appropriate for the solution of each large algebraic system to solve. More specifically, we will solve these previous large algebraic systems by parallel asynchronous and synchronous relaxation methods, which corresponds in fact to a subdomain method without overlapping.

3.1 Discretization of the Boundary Values Problems

The first step consists in the discretization of the global problem. Due to the specificity of each kind of boundary value problem to solve, various discretization techniques will be considered.

Using an implicit time marching scheme, the Navier-Stokes equation is discretized on staggered grids by the finite volume method [12]. This time marching scheme is a predictor-corrector algorithm based on the splitting of the solution of velocity equations and pressure equations; more precisely, for the global Navier-Stokes problem, the P.I.S.O. algorithm (Pressure-Implicit Splitting of Operators) is carried out [13]. At each time step, P.I.S.O. method is constituted by a sequence of one predictor step and two corrector steps and then the arithmetic operations on the velocity are decoupled from those of the pressure. Let (u^n, v^n, w^n) and p^n the values of the velocity and the pressure at the n^{th} time step. Then the predictor step consists in computing one time step for the Navier-Stokes equation by leaving the pressure unchanged. An intermediate velocity ($u^{n+1/3}$, $v^{n+1/3}$, $w^{n+1/3}$) is obtained by solving three algebraic systems in which the discretization matrices A_u, A_v, A_w are obtained by the finite volume method. Then, each corrector step consists in

1) solving the mass conservation equation $div(\vec{U}) = 0, \Omega \times [0,T]$, where the pressure's gradient is substituted by the velocity,
2) correcting the former approximate velocity with the computed pressure.

Then the velocities ($u^{n+2/3}$, $v^{n+2/3}$, $w^{n+2/3}$), (u^{n+1}, v^{n+1}, w^{n+1}) and the pressures $p^{n+1/2}$, p^{n+1} are obtained.

This method leads to the solution of five large sparse linear systems at each time step. Moreover, in each corrector step, we have to solve an algebraic linear system in which it is necessary to invert a matrix A_p. In fact these four previous matrices have a common property to be irreducible or strictly diagonal dominant and have strictly positive diagonal entries and non positive off-diagonal entries. So, classically the matrices A_u, A_v, A_w and A_p are M-matrices (see [14], [15] and [16]) and the reader is referred to [15] for the values of the entries of the matrices A_u, A_v, A_w and A_p. Finally it was shown that two corrector steps are sufficient to obtain a suitable accuracy (see [13]), compatible with the discretization scheme and round off error propagation.

The time dependant convection-diffusion transport equation is very easy to discretize. The evolutive part of this equation is discretized by using an implicit time marching scheme; for the spatial part of this equation, the classical seven points scheme is used for the Laplacian and decentered schemes for the convection terms. More precisely, for the convection terms a forward difference scheme is used if the convection coefficient is strictly positive and a backward difference scheme otherwise. Then, since the diagonal entries are strictly positive and the off-diagonal entries are non positive, and moreover due to the fact that the resulting discretization matrix is strictly (irreducible) diagonally dominant, finally the discretization matrix of the time dependant convection-diffusion equation is an M-matrix.

Finally, since the conductivity K arising in the potential diffusion equation is not constant, it is necessary to derive appropriate finite difference scheme. This scheme is obtained by considering a forward-backward scheme and a backward-forward scheme (see [14] and [15]); then by combining these two previous schemes, more precisely by taking their mean we obtain the discretization matrix. This matrix have also strictly positive diagonal entries and non positive off-diagonal entries and is irreducible diagonally dominant and consequently is an M-matrix.

Consequently, it can be also noted that for the transport equation and the potential equation we obtain two additional large sparse linear systems to solve. Reference is made to [15] for more details concerning the values of the entries of all the discretization matrices.

3.2 Parallel Asynchronous Subdomain Relaxation Algorithms

On the mathematical point of view and at each time step, all the linear systems derived from appropriate discretization of the linear and nonlinear boundary value problems have a common property which, according to theoretical results, ensure the convergence of the parallel synchronous and asynchronous relaxation methods studied in previous works (see [3], [4] and [5] for example). Indeed since at each time step these matrices are M-matrices then the considered iterative parallel method, corresponding in fact to a subdomain method without overlapping, converge for every subdomain decomposition (see [4]). For clarity, we recall hereafter the formulation of parallel synchronous and more generally asynchronous relaxation methods. Let us consider the solution of the following linear algebraic system

$$\mathrm{A.X} = \mathrm{B} \tag{4}$$

where X and B are respectively, a vector whose components approximate the values of the exact solution and the right hand side of the system respectively, at each point of the mesh. We consider now a block decomposition from the previous linear algebraic system and associate the following fixed-point mapping

$$X_i = A_{i,i}^{-1}(B_i - \sum_{j \neq i} A_{i,j} X_j) = F_i(X),\ \mathrm{i} = 1, \cdots, \mathrm{m}, \tag{5}$$

where m is an integer denoting the number of blocks. This kind of fixed point problem can be considered, at each time step, for the solution of each discretized boundary value problem which models the electrophoresis problem. Then, the problem consists in solving the following fixed point problem

$$\begin{cases} \mathit{Find}\, \mathrm{X}^* \text{ such that} \\ \mathrm{X}^* = F(\mathrm{X}^*) \end{cases} \tag{6}$$

where $X \to F(X)$ is a fixed point mapping defined in a finite dimensional space. For all X, consider the following block-decomposition of the mapping F associated to the block decomposition

$$\mathrm{F}\,(\mathrm{X}) = (\mathrm{F}_1\,(\mathrm{X})\,, \ldots., \mathrm{F}_m\,(\mathrm{X})\,). \tag{7}$$

We consider the distributed solution of the fixed point problem via a parallel asynchronous block relaxation method defined as follows (see [3] to [5]): let the initial guess $X^{(0)}$ be given and for every $q \in N$ assume that we can get $X^{(1)}$,……, $X^{(q)}$; then $X^{(q+1)}$ is defined recursively by

$$X_i^{(q+1)} = \begin{cases} X_i^{(q)} \text{ if } i \notin J(q) \\ F_i(\ldots, X_j^{(s_j(q))}, \ldots) \text{ if } i \in J(q) \end{cases} \tag{8}$$

where J = {J(q)} , q ∈ N, is a sequence of nonempty sets such that

$$\begin{cases} J(q) \subset \{1,\ldots,m\},\ J(q) \neq \emptyset,\ \forall q \in \mathrm{N}, \\ \forall i \in \{1,\ldots,m\},\ \mathrm{Card}(\{q \in \mathrm{N} \mid i \in J(q)\}) = +\infty \end{cases} \tag{9}$$

and

$$\begin{cases} \forall q \in N,\ \forall j \in \{1,\ldots,m\},\ s_j(q) \in N,\ 0 \leq s_j(q) \leq q, \\ \forall q \in N,\ s_i(q) = q \text{ if } i \in J(q), \\ \forall q \in N,\ \forall j \in \{1,\ldots,m\},\ \lim_{q \to \infty}(s_j(q)) = +\infty. \end{cases} \tag{10}$$

The previous asynchronous iterative scheme models computations that are carried out in parallel without order nor synchronization and describes in fact a subdomain method without overlapping. Particularly, it permits one to consider distributed computations whereby processors go at their own pace according to their intrinsic characteristics and computational load. The parallelism between the processors is well described by J since J(q) contains the number of components relaxed by each processor on a parallel way while the use of delayed components permits one to model nondeterministic behavior and does not imply inefficiency of the considered distributed scheme of computation. Note that, theoretically, each component of the vector must be relaxed an infinite number of time. The choice of the relaxed components may be guided by any criterion, and, in particular, a natural criterion is to pick-up the most recently available values of the components computed by the processors.

Remark. The previous algorithm describes a computational method where the communications between processors can be synchronous or asynchronous. Among them parallel synchronous methods, when $s(q) \equiv q$, $\forall\ q \in \mathrm{N}$; moreover if $J(q) = \{1,\ldots, m\}$ and $s(q)=q$, $\forall\ p \in \mathrm{N}$, then the previous algorithm describes the sequential block Jacobi method while if $J(q) = q.\mathrm{mod}(m) + 1$ and $s(q)=q$, $\forall\ q \in \mathrm{N}$, then the previous algorithm models the sequential block Gauss – Seidel method. So, the previous model of parallel asynchronous algorithm appears like a general model.

For the solution of the electrophoresis problem, a numerical time marching scheme is implemented and, at each time step, we have to solve seven large scale algebraic system by using either parallel synchronous or asynchronous algorithms.

Then, for the solution of each stationary problems derived from the discretization, the convergence of classical synchronous or asynchronous relaxation algorithms has been established by various ways, using contraction techniques (see [3] and [4]) or partial ordering techniques (see [5]). To summarize, as previously said, since the discretization matrix is an M-matrix then thanks to various results established in [3], [4] and [5], the iterative process described previously converge to X^*, for every initial guess $X^{(0)}$. The reader is referred to these previous references for more details. Moreover assume that the algebraic system is split into r blocks, $r \leq m$, corresponding to a coarser subdomain decomposition without overlapping; then using results in [4], it can be shown that the parallel classical asynchronous block relaxation methods converge for this coarser decomposition. Furthermore, if the subdomain decomposition associated with m blocks is a point decomposition, then the parallel asynchronous block relaxation methods converge for every subdomain coarser decomposition and for every numbering (lexicographical or red-black) of the blocks.

Remark. Note also that, in the parallel experiments presented in section 5, we have solved each submatrix constituted by gathering adjacent blocks by a symmetric block Gauss – Seidel method.

4 Implementation

4.1 Environment for Distributed High Performance Computing

P2PDC [7] is a peer-to-peer demonstrator for distributed high performance computing that was developed at LAAS (see [8]). There are only three classical functions: P2P_Send, P2P_Receive and P2P_Wait. The operation P2P_Wait is particular and used in order to wait for the arrival of a message from another machine. For further details about P2PDC environment, reference is made to [8].

4.2 Parallel Implementation of Proteins Separation

The implementation is based on master/slave paradigm, which is a commonly used approach for parallel and distributed applications. In our case, a master process controls the distribution of work to a set of slave processes. The following process is used for the creation and the solution of all linear systems that arise in the numerical simulation of electrophoresis phenomenon. Matrix and right hand side creation have been implemented sequentially since this part of computation is not very intensive. In this way, the master process can be seen as a matrix and a vector filler that feeds a parallel linear system solver. In other words, only intensive computations have been parallelized. This programming method greatly simplifies the solution of boundary value problems.

A numerical solution has been previously implemented in Fortran 90 as a parallel synchronous and asynchronous algorithm using MPI (Message Passing Interface) and experiments using the parallel Schwarz domain decomposition method with asynchronous and synchronous communications has been compared in the context of distributed environment (see [14] and [15]). In the present study the parallel experimentation is different since we consider here subdomain methods without overlapping performed on a demonstrator of peer-to-peer platform. In order to use again some Fortran 90 parts of the code adapting to P2PDC platform, it is necessary to interface the code in Fortran 90 language in C language and conversely. The Figure 2 shows the process of Fortran 90 implementation of the parallel subdomain decomposition method.

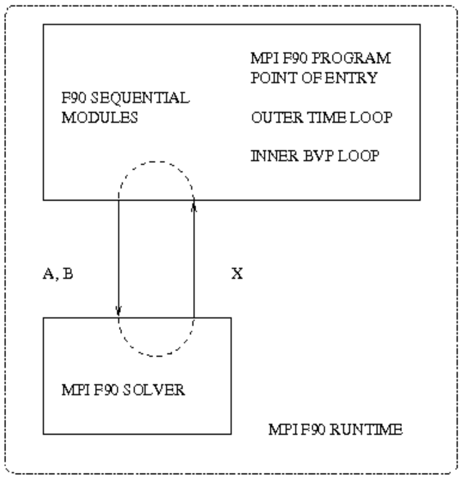

Fig. 2. Process of preview Fortran 90 implementation

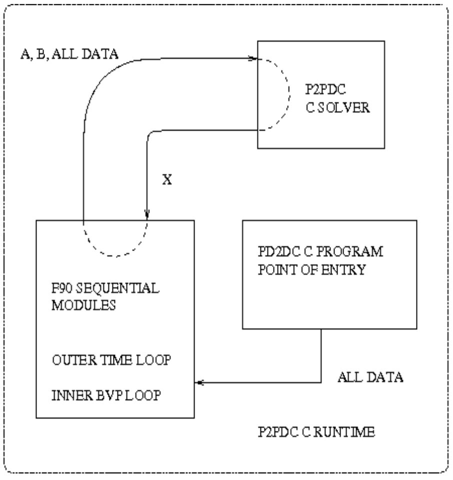

Fig. 3. Implementation of subdomain method without overlapping on P2PDC

The platform P2PDC presents only an interface in C language. A function called *Calculate* permits one to distribute on each peer the computation task and is the point of entry of the program. The Figure 3 shows the implementation of the solution by subdomain method without overlapping performed on P2PDC platform. All P2PDC data structures are passed from the point of entry through the Fortran 90 subroutines, in order to reach the C parallel solver written with P2PDC communication API.

In the same way of [17], due to the specificity of the implicit time marching scheme and the P2PDC conception, we implement an additional synchronization barrier using P2P_Wait function when a new time step is considered and also an efficient stopping criterion using a supplementary convergence manager peer only devoted to this task.

The algorithms can be summarized as follows:

```
Algorithm on P2PDC Calculate
for each time step do
         call Fortran 90 program
end for

Algorithm on Fortran part
for the seven linear system to solve
         compute A,B with Fortran 90 sequential modules
         solve algebraic system
end for

Algorithm for one algebraic system
while no convergence do
         if (master) then
```

```
          **compute** discretization matrix and right hand
          side of the system to solve
          **send** input data to slaves
          **compute** SPMD parallel algorithm to solve the
          linear system
          **receive** output data from slaves
        **else**
          **receive** input data from master
          **compute** SPMD parallel algorithm to solve the
          linear system
          **send** output data to master
        **endif**
**end while**
```

5 Parallel Experiments

In the considered numerical simulations, the size of the domain Ω is 30 x 0.5 x 10 cm. A regular mesh is defined on this domain; this mesh is constituted by 624 x 234 x 31 = 4 526 496 mesh points. The domain is split into as many subdomains as the number of processors; the subdomains are numbered using a lexicographical ordering. In the following, we consider a migration for only one protein injected on the face C of the cell. The electrical field is equal to 950 m^{-1}, the Reynolds number is equal to 250, the electrophoretic mobility is equal to 526 x 10^{-8} m^2 v^{-1} s^{-1} and the filament radius is equal to 1.5 mm. Figure 4 visualizes the computed concentration and the protein filament.

The parallel experiments have been performed using on one hand the homogeneous cluster located in Sophia Antipolis of the Grid'5000 platform as a support of P2PDC, the French grid platform composed of 2970 processors with a total of 6906 cores distributed over 9 sites and on the other hand heterogeneous machines located on two distinct sites located in Sophia Antipolis and in Nancy. Most of them have at least a Gigabit Ethernet network for local machines. Nodes between the different sites range from 2.5 Gflops up to 10 Gflops. Several sites of Grid'5000 have several clusters with different performances. Table 1 displays the characteristics of the machines used in the computational experiments.

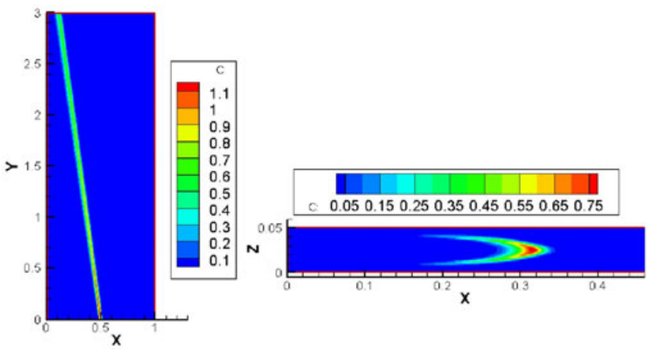

Fig. 4. Computed concentration and protein filament

Table 1. Characteristics of machines on each site

Site	Cluster	Processors Type	Speed GHz	CPU	Core	RAM
Sophia	suno	Intel Quad Xeon E5520	2,26	2	4	32
Nancy	griffon	Intel Quad Xeon L5420	2,5	2	4	16

These machines use Intel processor-based running Linux 64 bits. Note that the volume of data to be stored on each core is very large. It's the reason why, due to the fact that seven linear algebraic systems must be solved at each time step, the number of discretization grid points is not chosen too high. Taking into account of the size of the electrophoresis cell, this number of grid points does not degrade the accuracy of the numerical simulation. Moreover two successive time steps are considered in the implicit time marching scheme. Consequently very large memory size is needed, in fact greater than 8 GB. Thus, the computational machines are selected in order to satisfy this constraint. Note that, in the studied application, at each time step the heterogeneous values of the entries of the matrices and of the right-hand side must be stored, and consequently, the previous constraint must be satisfied; so, only machines having large memory could be used. The number of machines used has been limited up to 32 according to the duration of the exploitation of the computational codes in a dedicated mode and to constraints of exploitation. Moreover Table 5 and Figure 6 show clearly, that if more than 32 distant machines are used, then the elapsed time of synchronous parallel algorithm increases. Thus, for the considered size of the algebraic systems, which are not very large, since the overhead damages the performances of the parallel synchronous algorithms, additional machines are not really necessary. Consequently, the number of machine is also limited to 32 in asynchronous exploitation. In the preliminary experiments, the sequential, synchronous and asynchronous parallel implementations of the algorithms are compared. From a numerical point of view, the global electrophoresis problem is very hard to solve. Indeed, among the seven linear algebraic systems to solve, three of them are very ill-conditioned; it is particularly true for the two linear algebraic systems corresponding to the two corrector steps for the solution of the Navier-Stokes equation and also for the solution of the potential equation. Consequently, the asymptotic convergence rate is low and supplementary machines are not suitable. In the implementation of the subdomain method without overlapping, several adjacent blocks are assigned to each processor; thus, on each machine, this assignation allows a behavior of the parallel algorithm as close as possible of a symmetric block Gauss-Seidel method. Thus from a practical point of view, parallel synchronous and asynchronous subdomain method without overlapping are more efficient.

Table 2 shows the sequential elapsed time obtained on a cluster machine of each site. The results of the experiments are summarized in Tables 3 to 6 and Figures 5 to 6.

Table 2. Elapsed time and relaxations with sequential algorithm on each site

Sophia		Nancy	
Time	Relax	Time	Relax
12825	15250	15929	15250

Table 3. Elapsed time and relaxations on the cluster suno (Sophia)

	Asynchronous				Synchronous	
Peers	Time/s	Relaxations			Time/s	Relaxations
		Min	Max	Average		
2	8493	19564	20600	20082	8797	16574
4	3959	21995	23193	22565	8121	16582
8	2810	27831	31581	29255	5653	18294
16	1423	22929	28453	24702	2641	20602
32	636	13004	18986	14459	1596	20840

Table 4. Speedup and efficiency on the cluster suno (Sophia)

	Asynchronous		Synchronous	
Peers	Speedup	Efficiency	Speedup	Efficiency
2	1.51	0.75	1.46	0.73
4	3.24	0.81	1.58	0.39
8	4.57	0.57	2.27	0.28
16	9.02	0.56	4.86	0.30
32	20.18	0.63	8.04	0.35

Table 5. Elapsed time and relaxations on distant clusters suno (Sophia) and griffon (Nancy)

	Asynchronous				Synchronous	
Peers	Time/s	Relaxations			Time/s	Relaxations
		Min	Max	Average		
2	11271	21077	26273	23675	12148	16574
4	5743	22008	31204	28197	11786	16582
8	3961	28158	37301	33185	7151	18294
16	2177	27562	32287	29980	5849	20602
32	1196	15288	21410	18392	9585	20840

Table 6. Speedup and efficiency on two distant clusters suno (Sophia) and griffon (Nancy)

	Asynchronous		Synchronous	
Peers	Speedup	Efficiency	Speedup	Efficiency
2	1.14	0.57	1.06	0.53
4	2.23	0.56	1.09	0.27
8	3.24	0.40	1.79	0.22
16	5.89	0.37	2.19	0.14
32	10.72	0.34	1.34	0.04

Tables 3 to 6 and Figures 5 and 6, clearly show the benefit of using parallel algorithms, particularly when the number of processors increases.

In Tables 3 and 5, the comparison of elapsed time between synchronous and asynchronous methods is presented; in Tables 4 and 6, the values of speedup and the efficiency are indicated. In Table 6, these values are pointed out only for information; indeed, in the case of computation on a heterogeneous architecture the notion of speed – up and efficiency has no sense. Nevertheless, note that here the speed – up and the efficiency are computed with respect of the lower sequential elapsed time.

In the implementation of the parallel method, it has to be noted that the computation of the entries of the matrices and of the right-hand sides are sequential. A parallelization of such data is not necessary, since the corresponding computation is about 3 % of the global elapsed time of the parallel version.

In Table 3 only the cluster Suno located at Sophia Antipolis is used. Clearly, when the number of processors is greater than two, it appears that the asynchronous parallel subdomain method without overlapping has better performance than the corresponding synchronous algorithm. Figure 5 shows clearly that no more additional machines are necessary, due to the size of each algebraic system to solve at each time step. Speedup and efficiency given in Table 4 show the interest of parallel asynchronous algorithms.

Table 5 presents the results of parallel experiments when two distant clusters are used. Note that the Suno cluster located in Sophia Antipolis give better elapsed time than the one obtained on griffon cluster located in Nancy. Similarly to the case of use of only one cluster, the asynchronous parallel subdomain method without overlapping has better performance than the corresponding synchronous parallel algorithm.

In the two cases, since there is no idle times, better performances of asynchronous methods follows from the fact that the weight of synchronizations do not penalize mainly the performance.

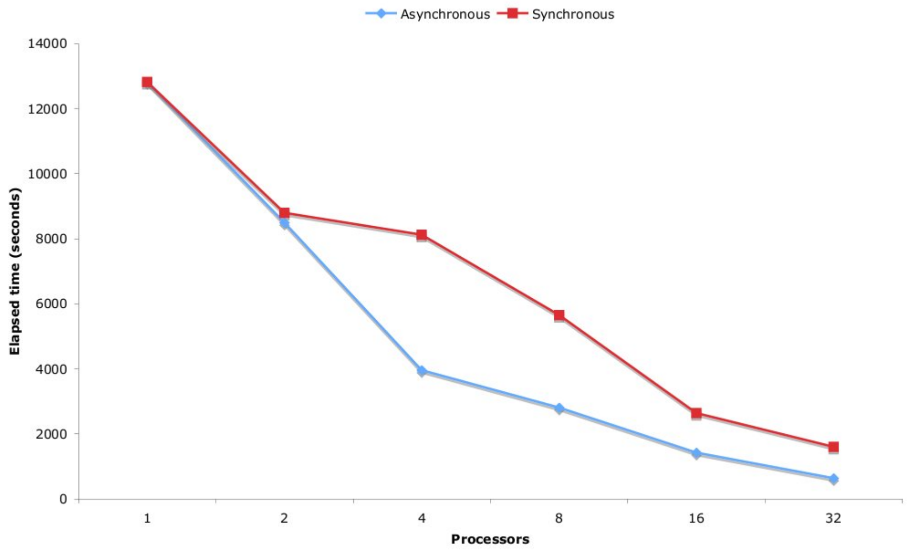

Fig. 5. Elapsed time for synchronous and asynchronous scheme on the cluster suno (Sophia)

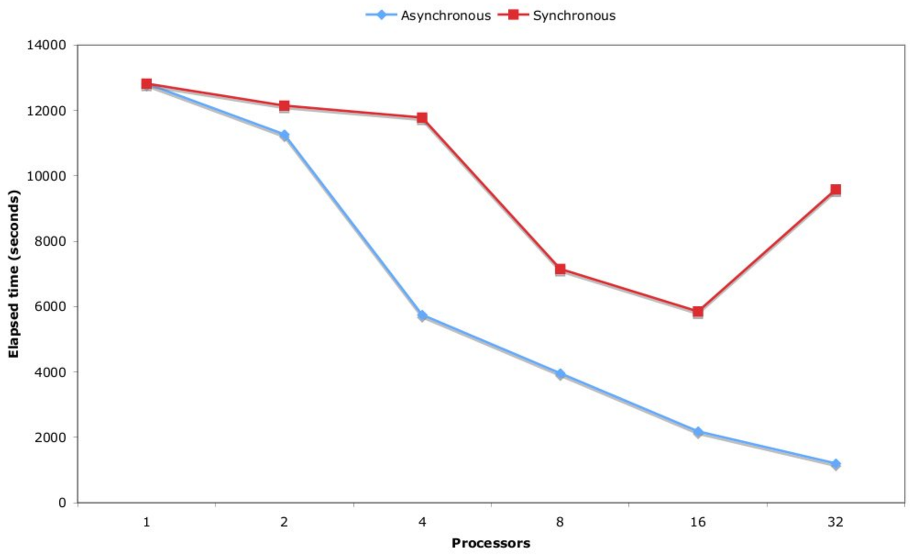

Fig. 6. Elapsed time for synchronous and asynchronous scheme on clusters suno (Sophia) and griffon (Nancy)

The performances obtained by asynchronous methods when we use only one cluster are better compared to the case of using two distant clusters. This is due to the fact that, in the latter case, the distant communications penalize the performances since the latency times between two distant nodes are more large; on the other hand when using distant nodes, the asymptotic convergence rate decreases due to the frequency of updates.

In Tables 3 and 5 we note that, in asynchronous parallel experiments, the number of relaxations necessary to reach convergence is slightly greater than the ones necessary when parallel synchronous schemes are used; as previously said this is particularly true when distant peers are used. This is a well-known drawback of asynchronous iterative schemes, due to the use of possibly delayed components. It turns out that the overhead generated by additional relaxations in the case of asynchronous algorithms is smaller than the synchronization overhead combined with processor idle time of parallel schemes of computation. Note that, when the number of processors increases, the number of relaxations necessary to reach convergence increases too. Nevertheless these additional relaxations improve the solution's accuracy and the numerical quality of the computations. Furthermore, despite higher number of relaxations, elapsed time of asynchronous scheme is less than the one obtained when synchronous scheme is used. This asynchronism is an efficient way to deal with communication overhead and load unbalance. Moreover, the efficiency of the synchronous algorithm decreases faster than the efficiency of the asynchronous algorithm when the number of processors increases. The lack of synchronization points and the use of current values of the components of the iterate vector induce better performances for the asynchronous parallel algorithm when several processors are used.

5 Conclusion

In this work, we have implemented the parallel numerical solution of the electrophoresis problem with the experimental P2PDC platform. Due to the fact that the algebraic systems involved in the parallel simulation process are very ill – conditioned, the solution of this problem is out of reach for sequential algorithms in a reasonable elapsed time. Parallel algorithms implemented with the P2PDC environment have been experimented using the distributed and heterogeneous Grid'5000 computing platform. Due to the weight of synchronizations, which is very high on this kind of platform, parallel synchronous and asynchronous algorithms have been tested successfully. Experiments show clearly that the asynchronous approach performs better than the synchronous one and that some acceleration can be obtained from parallel implementations of subdomain method on the considered platform. The use of an asynchronous parallel solver in a distributed peer-to-peer environment is a good approach.

Acknowledgment

This study has been made possible by ANR grant: ANR-07-CIS7-011 and support of Grid'5000. We thank gratefully M. The Tung NGUYEN (LAAS –Toulouse) for providing us the P2PDC framework.

References

[1] Clifton, M.J., Roux-de-Balman, H., Sanchez, V.: Electro hydrodynamic deformation of the sample stream in continuous-flow electrophoresis with an AC electric field. The Canadian Journal of Chemical Engineering 70, 1055–1062 (1992)
[2] Clifton, M.J.: Numerical simulation of protein separation by continuous-flow electrophoresis. Electrophoresis 14, 1284–1291 (1993)
[3] Giraud, L., Spiteri, P.: Parallel resolution of non-linear boundary value problems. Mathematical Modelling and Numerical Analysis 25(5), 579–606 (1991)
[4] Miellou, J.C., Spiteri, P.: "A criterion of convergence for general fixed point methods'. Mathematical Modelling and Numerical Analysis 19, 645–669 (1985)
[5] Miellou, J.C., El Baz, D., Spiteri, P.: A new class of asynchronous iterative algorithms with order interval. Mathematics of Computation 67(221), 237–255 (1998)
[6] El Baz, D., Jourjon, G.: Some solutions for Peer to Peer Global Computing. In: Proceedings of 13th Euromicro Conference on Parallel, Distributed and Network-Base Processing, pp. 49–58 (2005)
[7] Nguyen, T.T., El Baz, D., Spiteri, P., Jourjon, G., Chau, M.: High Performance Peer-to-Peer Distributed Computing with Application to Obstacle Problem. In: Proceedings of IEEE IPDPS 2010 Conference, Atlanta (2010)
[8] El Baz, D., Nguyen, T.T.: A self-adaptive communication protocol with application to high performance peer to peer distributed computing. In: Proceedings of the 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, Pisa, pp. 327–333 (2010)

[9] Aumage, G.M.: MPICH/Madeleine: a True Multi-Protocol MPI for High Performance Networks. In: 15th International Parallel and Distributed Processing Symposium, IPDPS 2001 (2001)

[10] Anderson, D.P.: BOINC: A System for Public-Resource Computing and Storage. In: 5th IEEE/ACM International Workshop on Grid Computing, Pittsburgh, USA (2004)

[11] Andrade, N., Cirne, W., Brasileiro, F., Roisenberg, P.: OurGrid: An approach to easily assemble grids with equitable resource sharing. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2003. LNCS, vol. 2862, pp. 61–86. Springer, Heidelberg (2003)

[12] Patankar, S.V.: Numerical heat transfer and fluid flow. Mc. Graw Hill, New York (1980)

[13] Issa, R.I.: Solution of the implicitly discretized fluid flow equation by operator splitting. Journal of Computational Physic 62, 40–65 (1986)

[14] Chau, M., Spiteri, P., Boisson, H.C.: Parallel numerical simulation for the coupled problem of continuous-flow electrophoresis. International Journal for Numerical Methods in Fluids 55, 945–963 (2007)

[15] Chau, M., Spiteri, P., Guivarch, R., Boisson, H.C.: Parallel asynchronous iterations for thesolution of a 3D continuous-flow electrophoresis problem. Computers and Fluids 37(9), 1126–1137 (2008)

[16] Ortega, J.M., Rheinboldt, W.: Iterative solution of nonlinear equations in several variables. Academic press, London (1970)

[17] Garcia, T., Chau, M., Nguyen, T., El Baz, D., Spiteri, P.: Asynchronous peer-to-peer distributed computing for financial applications. In: 12th IEEE International Workshop on Parallel and Distributed Scientific and Engineering Computing (IEEE IPDPS/PDSEC 2011), Anchorage, Alaska, USA, May 16-20 (2011)