2011 IEEE International Parallel & Distributed Processing Symposium

# Asynchronous peer-to-peer distributed computing for financial applications

Thierry GARCIA
Université de Toulouse
INP – ENSEEIHT – IRIT
BP 7122, 2 Rue Camichel
F-31071 Toulouse Cedex, France
e-mail: thierry.garcia@enseeiht.fr

Ming CHAU
Advanced Solutions Accelerator
199 Rue de l'Oppidum
F-34170 Castelnau le Lez, France
e-mail: mchau@advancedsolutionsaccelerator.com

The Tung Nguyen, Didier El-Baz
CNRS - LAAS 7 avenue du colonel Roche
F-31077 Toulouse, France
Université de Toulouse; UPS, INSA, INP, ISAE.
e-mail: ttnguyen@laas.fr elbaz@laas.fr

Pierre SPITERI
Université de Toulouse
INP – ENSEEIHT – IRIT
BP 7122, 2 Rue Camichel
F-31071 Toulouse Cedex, France
e-mail: pierre.spiteri@enseeiht.fr

*Abstract*— **This paper deals with the numerical solution of financial applications, more specifically the computation of American and European options derivatives modelled by boundary values problems. In such applications we have to solve large-scale algebraic linear systems. We concentrate on synchronous and asynchronous parallel iterative algorithms carried out on peer-to-peer networks. The properties of the operators arising in the discretized problem ensure the convergence of the parallel iterative synchronous and asynchronous algorithms. Computational experiments performed on peer-to-peer networks are presented and analyzed.**

***Keywords-component; parallel asynchronous algorithms; distributed computing; iterative numerical methods; peer-to-peer networks; European options derivatives; American options derivatives.***

## I. INTRODUCTION

For the last years financial applications have known great developments. In particular, European and American options derivatives modelled by the classical Black and Scholes equation have known great interest [1]. The problem consists of solving a time dependant boundary value problem defined on an unbounded domain generally included in the three dimensional space. A classical artifice consists of solving the Black and Scholes equation on a bounded domain and to increase the size of the domain for ensuring convergence to the exact solution [2]. So, we have to solve boundary value problems numerically. Taking into account the size of the algebraic systems derived from the discretization process of the Black and Scholes equation, parallel iterative algorithms are prefered.

In this paper we concentrate on asynchronous and synchronous iterative methods implemented on peer-to-peer (P2P) architectures. Recent advances in microprocessors architecture and networks permit one to consider new applications like High Performance Computing (HPC). Therefore, we can identify a real stake at developing new protocols and P2P environments for HPC since this can lead to economic and attractive solutions. Task parallel models and distributed iterative methods for large scale parallel numerical simulation on P2P networks gives rise to numerous challenges like communication management, scalability, heterogeneity and peer volatility on the network (see [3]). Some performance issues can be addressed via distributed asynchronous iterative algorithms (see [4] to [8]). The reader is referred to [5] for the solution of the stationary obstacle problem and to [9] for contribution to the development of a new protocol and an environment for P2P high performance computing. The approach presented in [9] is different from MPICH Madeleine [10] in allowing the modification of internal transport protocol mechanism in addition to switch between networks. Recently, middleware like BOINC [11] or OurGrid [12] have been developed in order to exploit the CPU cycles of computers connected to the network. Those systems are generally dedicated to applications where tasks are independent and direct communications between machines are not needed.

In [5], we have considered a stationary problem solved by a Richardson like method. In the present study we consider an evolutive problem closer and related to the studied financial application, more precisely European and American options derivatives in which each stationary problem derived from the numerical time marching scheme is solved by a parallel block relaxation algorithm, which is more efficient than the Richardson's one. This kind of algorithm corresponds in fact to a subdomain method without overlapping. In addition, due to the overhead of synchronizations the use of distributed asynchronous iterative algorithms [6], [7] and [8] seems well suited to P2P computing.

1530-2075/11 $26.00 © 2011 IEEE
DOI 10.1109/IPDPS.2011.292

The present paper is structured as follows: section II deals with the presentation of the financial application (European and American options derivatives). In section III, we present parallel synchronous, and more generally parallel asynchronous iterative schemes. In section IV, environment for peer-to-peer high performance computing is described. In section V, implementation of parallel iterative algorithms is detailed. Computational experiments are displayed and analysed in section VI, in particular the pertinence to apply such parallel asynchronous iterative methods is presented for the solution of the model problems. Finally concluding remarks are presented in section VII.

## II. THE PROBLEM OF OPTIONS DERIVATIVES

The classical Black and Scholes equation is a boundary value problem describing the evolution of call or put options in the field of mathematics of financial derivatives [1]. Among the many descriptions of financial option contracts, we can consider:

- the European option, which may only be exercised at expiry, i.e. when the time τ takes the value T of the final expiry date,
- the American option, which may be exercised at any time prior to expiry.

These two kinds of options are modelled by retrograde time dependent convection – diffusion equations. From a mathematical point of view, the main difference between them consists in the fact that European option is defined on a normed vectorial space while the American option is defined on a closed convex set included in a normed vectorial space; consequently, the determination of American option needs the projection of the computed values on the convex set.

Classically, an European option is modeled by the following time dependent linear equation:

$$\begin{cases} \dfrac{\partial v(\tau,x)}{\partial \tau} + (r - \dfrac{\sigma^2}{2}).\nabla v + \dfrac{\sigma^2}{2}\Delta v - rv = 0, \text{ everywhere in } [0,\mathrm{T}]xR^n \\ v(T,x) = \psi(x) \end{cases}$$

where ψ(x)= Max(x-K,0) (in the case of call option) or ψ(x) = Max(K-x,0) (in the case of put option); in the above equations, K denotes the exercise price – called classically strike, v denotes the value of the considered option, i.e. a “call” or a “put” option; v = v(τ,x) is a function of the current value of the underlying asset x and of the time τ. Note also that the considered option also depends on the following parameters:

- r the interest rate,
- σ the volatility of the underlying asset, σ being in fact the instantaneous standard deviation of the price with respect to K fixed beforehand; in fact σ characterizes the uncertainly of the behavior of the option.

Using the same previous notations, the American option is modeled by the following nonlinear equation:

$$\begin{cases} \dfrac{\partial v(\tau,x)}{\partial \tau} + (r - \dfrac{\sigma^2}{2}).\nabla v + \dfrac{\sigma^2}{2}\Delta v - rv \geq 0, \ v(\tau,x) \geq \psi(x), \text{ everywhere in } [0,\mathrm{T}]xR^n \\ (\dfrac{\partial v(\tau,x)}{\partial \tau} + (r - \dfrac{\sigma^2}{2}).\nabla v + \dfrac{\sigma^2}{2}\Delta v - rv)(v(\tau,x) - \psi(x)) = 0, \text{ everywhere in } [0,\mathrm{T}]xR^n \\ v(T,x) = \psi(x) \end{cases}$$

One of the main difficulty lies on the fact that the Black and Scholes equation is not defined on a bounded domain, but is defined on the unbounded domain $R^n$, n ≥ 1. This difficulty overcome by considering the problem defined on a bounded domain $\Omega \subseteq R^n$ with appropriate boundary conditions. Then it can be proved that the solution of the equation defined on the bounded domain Ω converges to the solution of the model problem when the measure of Ω tends to infinity [2]. Another particularity of the problem to solve, is that the value of the option is not known at the initial time τ = 0; only the final value v(T,x) is known. So the problem consists of computing v(0,x).

These previous two particularities can be treated, first by considering problems defined on a bounded large domain Ω and secondly by a change of variable, such that the variable τ is replaced by a variable t = T - τ. Thus, the model problem of the European option is replaced by the following linear problem:

$$\begin{cases} \dfrac{\partial v(t,x)}{\partial t} - (r - \dfrac{\sigma^2}{2}).\nabla v - \dfrac{\sigma^2}{2}\Delta v + rv = 0, \text{ everywhere in } [0,\mathrm{T}]x\ \Omega \\ v(0,x) = \psi(x) \\ B.C. \text{ of v(t,x) defined on } \partial\Omega \end{cases}$$

where B.C. describes the boundary conditions on the boundary ∂Ω of the domain Ω. Practically, the Dirichlet condition where v is fixed on ∂Ω or the Neumann condition where the normal derivative of v is fixed on ∂Ω is classically considered.

Concerning the American option, by the same way, we have to solve the following nonlinear problem:

$$\begin{cases} \dfrac{\partial v(t,x)}{\partial t} - (r - \dfrac{\sigma^2}{2}).\nabla v - \dfrac{\sigma^2}{2}\Delta v + rv \geq 0, \text{ v(t,x)} \geq \psi(x), \text{ everywhere in } [0,\mathrm{T}]x\Omega \\ (\dfrac{\partial v(t,x)}{\partial t} - (r - \dfrac{\sigma^2}{2}).\nabla v - \dfrac{\sigma^2}{2}\Delta v + rv)(v(t,x) - \psi(x)) = 0, \text{ everywhere in } [0,\mathrm{T}]x\Omega \\ v(0,x) = \psi(x) \\ B.C of\ v(t,x) \text{ defined on } \partial\Omega \end{cases}$$

## III. Numerical Solution of the Problems of Options Derivatives

For the numerical solution of the financial problems, we consider that the temporal part is discretized by implicit or semi – implicit scheme. While, for the spatial part of the problem, the bounded domain $\Omega \subseteq R^3$ is discretized with an uniform mesh. Note also that the spatial differential operators are discretized by appropriate schemes as follows:
- the Laplacian is discretized by using the classical seven points scheme,
- the first derivatives, corresponding to the convection phenomenon, are discretized by using decentered scheme; more precisely, if the coefficient of the considered first derivative is strictly positive, then the forward-difference formula is used, otherwise backward–difference formula is considered.

Thus, if good accuracy is wanted, then the full discretization of the Black and Scholes problem leads to the solution of a very large linear algebraic system at each time step.

Let A denote the discretization matrix of the model problem. It follows from the considered appropriate discretization of the Black and Scholes equations, that the diagonal entries of the matrix A are strictly positive and that the off-diagonal entries are nonpositive. Moreover, the matrix A is strictly diagonally dominant. Thus, A is an M – matrix [13]. This very interesting property of the matrices to invert, at each time step, has a consequence regarding to the behaviour of the parallel iterative algorithms considered in the sequel; indeed the convergence of the iterative scheme is ensured (see [4] and [6] to [8]).

Note also, that due to the large size of the linear algebraic system to invert at each time step, it is necessary to use iterative methods. More precisely, we consider in the present study iterative parallel synchronous and asynchronous block relaxation algorithms studied in [6], [7] and [8], implemented in the present study on peer-to-peer distributed architecture. In the sequel, let us recall the formulation of parallel synchronous and more generally asynchronous block relaxation algorithms for the solution of a large linear algebraic system.

Let us consider the solution of the linear algebraic system associated, for example to the solution of the discretized European option

$$A.V=F$$

where V and F are respectively, a vector whose components approximate the values of the exact solution and the right hand side of the partial differential equation 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:

$$V_i = A_{i,i}^{-1}(F_i - \sum_{j\neq i} A_{i,j}V_j) = \Phi_i(V),\ i = 1,\cdots,m,$$

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 the problem of European option.

For the solution of the discretized American option problem, we have to consider now the projection on a convex set (see [1]) as follows:

$$V_i = \text{Proj}(A_{i,i}^{-1}(F_i - \sum_{j\neq i} A_{i,j}V_j)) = \Phi_i(V),\ i = 1,\cdots,m.$$

Then, by considering the appropriate operator as well as, the European option than the American option, the problems consist of solving the following fixed point problem:

$$\begin{cases} \mathit{Find}\ V^* \text{ such that} \\ V^* = \Phi(V^*) \end{cases} \tag{1}$$

where $V \to \Phi(V)$ is a fixed point mapping defined in a finite dimensional space. For all V, consider the following block-decomposition of the mapping $\Phi$ associated to the parallel distributed implementation:

$$\Phi(V) = (\Phi_1(V), \ldots, \Phi_m(V))$$

We consider the distributed solution of the fixed point problem (1) via a parallel asynchronous block relaxation method defined as follows (see [6] to [8]): let the initial guess $V^{(0)}$ be given and for every $p \in N$ assume that we can get $V^{(1)}, \ldots\ldots, V^{(p)}$; then $V^{(p+1)}$ is defined recursively by:

$$V_i^{(p+1)} = \begin{cases} V_i^{(p)} \text{ if } i \notin J(p) \\ \Phi_i(\ldots, V_j^{(s_j(p))}, \ldots) \text{ if } i \in J(p) \end{cases} \tag{2}$$

where $\mathbf{J} = \{J(p)\}$, $p \in N$, is a sequence of nonempty sets such that:

$$\begin{cases} J(p) \subset \{1,\ldots,m\},\ J(p) \neq \emptyset,\ \forall p \in N, \\ \forall i \in \{1,\ldots,m\},\ \text{Card}(\{p \in N \mid i \in J(p)\}) = +\infty \end{cases} \tag{3}$$

and

$$\begin{cases} \forall p \in N,\ \forall j \in \{1,\ldots,m\},\ s_j(p) \in N,\ 0 \le s_j(p) \le p, \\ \forall p \in N,\ s_i(p) = p \text{ if } i \in J(p), \\ \forall p \in N,\ \forall j \in \{1,\ldots,m\},\ \lim_{p\to\infty}(s_j(p)) = +\infty. \end{cases} \tag{4}$$

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 peers go at their own pace according to their intrinsic characteristics and computational load. The parallelism between the peers is well described by J since J(p) contains the number of components relaxed by each processor on a parallel way while the use of delayed components in (2) 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 algorithm (2) – (4) describes a computational method where the communications between peers can be synchronous or asynchronous. Among them parallel synchronous methods, when $s(p) \equiv p, \forall\, p \in N$ ; moreover if $J(p) = \{1,\ldots, m\}$ and $s(p)=p, \forall\, p \in N$, then (2) – (4) describes the sequential block Jacobi method while if $J(p) = p.\mathrm{mod}(m) + 1$ and $s(p)=p, \forall\, p \in N$, then (2) – (4) 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 evolution Black and Scholes equations, a numerical time marching scheme is implemented and, at each time step, we have to solve a large scale algebraic system by using either parallel synchronous or asynchronous algorithms.

Then, for the solution of stationary problems derived from European and American options derivatives, the convergence of classical synchronous or asynchronous relaxation algorithms has been established by various ways, using contraction techniques (see [6] and [7]) or partial ordering techniques (see [8]). To summarize, as previously said, since the discretization matrix is an M-matrix then thanks to various results established in [6], [7] and [8], the iterative process described by (2) – (4) converges to $V^*$, for every initial guess $V^{(0)}$. The reader is referred to these previous references for more details. Moreover assume that the algebraic system is split into q blocks, $q \leq m$, corresponding to a coarser subdomain decomposition without overlapping; then using results in [7], it can be shown by using the same arguments, that the parallel asynchronous block relaxation methods converge for this coarser decomposition. Furthermore, if the subdomain decomposition associated with m blocks is a point decomposition, then classical parallel asynchronous block relaxation methods converge for every subdomain coarser decomposition and for every numbering (lexicographical or red-black) of the blocks.

## IV. Environment for peer to peer high performance computing

P2PDC [5] is an environment for peer to peer high performance computing that was developed at LAAS-CNRS. Contrarily to existing solutions, P2PDC environment allows directed communications between peers by using the P2PSAP Peer To Peer Self Adaptive communication Protocol (see [9]). The P2PSAP protocol chooses dynamically the most appropriate communication mode between any peers according to schemes of computation and elements of context like topology. In the present study, note

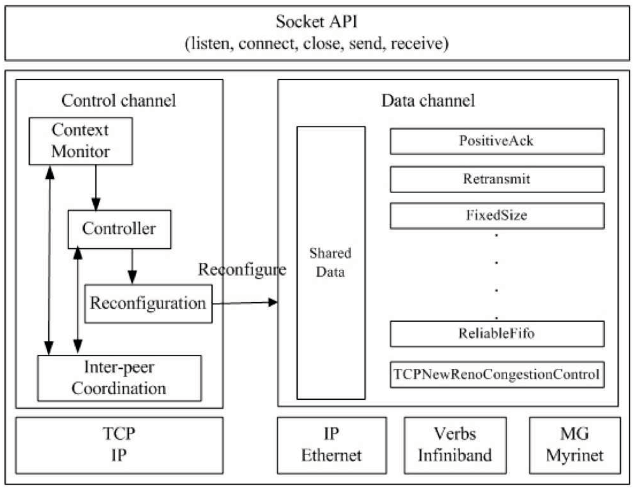

Figure 1. P2PSAP Protocol architecture

that the self adaptive capability is not used. In the following, we shall briefly present the P2PSAP protocol and the P2PDC environment.

### *A. Protocol P2PSAP*

The P2PSAP protocol is based on the Cactus framework (see [14] and [15]) and makes use of micro-protocols. Figure 1 shows the architecture of P2PSAP; the protocol consists of a Socket interface and two channels: a control channel and a data channel.

A **Socket API** is placed on the top of the protocol in order to facilitate programming. Session management commands like listen, open, close, setsockoption and getsockoption are directed to Control channel; while data exchange commands, i.e. send and receive commands are directed to Data channel.

The **Data channel** based on Cactus framework, transfers data packets between peers. It has two levels: the physical layer and the transport layer. The physical layer is encompassed to support communications on different networks, i.e. Ethernet, InfiniBand and Myrinet; the data channel can be triggered between the different types of networks. The transport layer is made of several micro-protocols; it can be reconfigured by substituting or removing and adding micro-protocols. The behavior of the data channel is triggered by the control channel.

The **Control channel** manages session opening, closure and data channel configuration. The TCP/IP protocol is used to exchange control messages since those messages must not be lost. The control channel has four main components. The *Context monitor* collects context data like schemes of computation, peers location or machine loads. The *Controller* manages session opening and end through TCP connection opening and closure; it also combines and analyzes context information so as to choose the configuration (at session opening) or to take reconfiguration decision (during session operation) for **Data channel**.

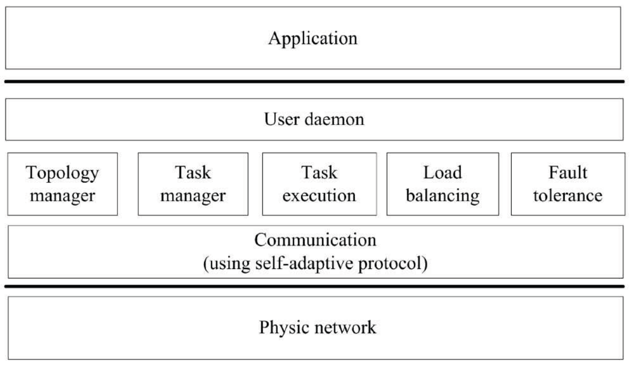

Figure 2. Environment architecture

Reconfiguration of Data channel is carried out via the dedicated Cactus functions by substituting or removing and adding micro-protocols.

## B. *Environment P2PDC*

Figure 2 illustrates the global architecture of the environment P2PDC. We describe now its main components
- *User daemon* is the interaction interface between the application and the environment; it allows users to submit their tasks and retrieve final results.
- *Topology manager* organizes connected peers into clusters and maintains links between clusters and peers.
- *Task manager* is responsible for subtasks distribution and results collection.
- *Task execution* executes subtasks and exchanges intermediate results.
- *Load balancing* estimates peer workload and migrates a part of work from overloaded peer to non-loaded peer.
- *Fault tolerance* ensures the integrity of the calculation in case of peer or link failure.
- *Communication* provides support for directed data exchange between peers using protocol P2PSAP.

In order to allow programmers to develop their own application easily, P2PDC proposes a programming model with a reduced set of communication operations and some deployment activities carried out automatically by the environment.

*Communication operations*: There are only three classical operations: *P2P_Send*, *P2P_Receive* and *P2P_Wait*. The idea is to facilitate programming of large-scale P2P applications and to hide the complexity of communication management as much as possible. The operation *P2P_Wait* is particular and used in order to wait for the arrival of a message from another peer.

Messages exchanged between peers can be decomposed into two classes: data messages and control messages. Data messages are used to exchange boundary values of block components at interfaces between peers; while control messages are used to exchange computational state like local termination criteria, termination command, etc. A *flags* parameter is added to communication operations in order to distinguish two classes of messages: the *CTRL_FLAG* indicates control messages and the *DATA_FLAG* indicates data messages. The control channel of P2PSAP exchanges not only messages for protocol configuration but also messages from P2PDC application with the so-called *CTRL_FLAG* flag. Then messages from P2PDC application with *DATA_FLAG* will be handled by data channel and messages from P2PDC application with *CTRL_FLAG* will be handled by control channel. Thus, the communication mode for data exchange is chosen according to the context of the user choice; while note that the communication mode for exchange of control messages is always asynchronous and reliable.

*Application programming model*: In order to develop an application, programmers have to write code for only three functions: *Problem_Definition()*, *Calculate()* and *Results_Aggregation()*. Others deployment activities like subtasks distribution or results collection are carried out automatically by the environment. In the *Problem_Definition()* function, programmers define the problem in indicating the number of subtasks and subtask data, computational scheme and number of peers necessary. In the *Calculate()* function, programmers write subtasks code; they can use *P2P_Send()* and *P2P_Receive()* to send or receive updates. Finally, programmers define how subtasks results are aggregated and the type of output, i.e. a console or a file, in the *Results_Aggregation()* function.

For further details about P2PSAP protocol and P2PDC environment, reference is made to [9] and [5] respectively.

## V. Implementation

The implementation of the considered financial application is carried out with the P2PDC environment taking into account the computational scheme on one hand and the stopping criterion of the iterative algorithm, on the other hand. The computation scheme (synchronous, asynchronous) is chosen at the beginning of the solution. Each node updates its assigned sub-blocks of components of the iterate vector in the lexicographical order. Then, the transmission of the boundary blocks assigned to each computational node to the contiguous blocks is delayed so as to reduce the waiting time in the case of the use of synchronous method. The parallel algorithm is based on the SPMD paradigm, which is commonly used for parallel and distributed application. In our case, each process initializes its own data set. The principle of the implementation of the parallel algorithm is summarized below; in this flow chart, r is the time step index, $V_0$ is the initial condition of the time dependant problem and $F_{r+1}$ is the right hand side computed at time r+1.

In the *Calculate()* function of P2PDC, *P2P_Send()* and *P2P_Receive()* operations will not have the same property according to the computation scheme used. The operation *P2P_Wait()* is used in order to synchronize the peers at each time step before the computation of the right-hand side. In the synchronous (resp. asynchronous) scheme, the *P2P_Send()* and *P2P_Receive()* operations are blocking (resp. not blocking). Note that this implementation of the parallel asynchronous method simplifies the exchange of boundary blocks in the block relaxation method but it is not adapted to the synchronized part at each time step. The P2PDC communication API (Application Programming Interface) does not offer a real synchronization barrier but only a function, which wait for the arrival of a message from another peer.

We specify that in the implementation of the application concerning the computation of European and American options, we use the classical block-relaxation method in which the blocks are numbered using the lexicographical ordering. More precisely, in the considered implementation several adjacent tridiagonal blocks are gathered; so, this kind of method can be viewed as a subdomain method without overlapping between the subdomains. Note also that each tridiagonal algebraic subsystem is solved using the TDMA method well adapted to the solution of tridiagonal subsystems. Moreover, the considered implementation allows having a multiplicative subdomain method. Finally, in the considered implementation of the subdomain relaxation method without overlapping the data are split into regular polygons.

**Algorithm: on computing peers**

```
// Initial condition $V_0$
Input $V_0$
For each time step r do
  // synchronized part
  Compute $\boxed{\delta t*F_{r+1} + V_0}$
  While no convergence do
    // by synchronous or asynchronous algorithm
    Solve by block relaxation method $\boxed{A.V=\delta t*F_{r+1} + V_0}$
    // Project in the case of American options
    $\boxed{V_0 \leftarrow V}$
    Send local convergence to convergence manager peer
    Receive global convergence from convergence manager peer
  End while
  If on barrier manager peer Then
    Wait all peers
    Send notification restart to computing peer
  Else
    Send notification message to barrier manager peer
    Wait notification restart from barrier manager peer
  End If
End for
Send final notification message to convergence manager peer.
```

For the implementation of the time marching scheme, the parallel synchronous algorithm is easy to implement, since the *P2P_Send* and the *P2P_Receive* are blocking operations. In the case of the asynchronous scheme, the implementation is more difficult. Due to the specificity of the implicit time marching scheme and to P2PDC conception, it is necessary to implement an additional synchronisation barrier when a new time step is considered. At the end of a time step, all the peers are synchronized thanks to the *P2P_Wait* operation and send a message to a barrier manager peer dedicated among all the computing peers. Then this manager barrier peer restarts each computing peer. An additional level of synchronization is performed only when the first relaxation is started.

With respect to [5], the Richardson solver is replaced by the subdomain without overlapping solver and a new efficient stopping criterion is implemented using a supplementary convergence manager peer only devoted to this task. Indeed, we have compared the efficiency of the stopping criterion proposed by [16] based on the use of a decentralized method and the centralized one presented in [17]. The comparison of the two stopping criteria has shown that the number of relaxations are not the same and we have finally chosen the stopping criterion proposed by [17] since more relaxations are performed and better numerical quality of the computed solution is obtained. For a current time step, when a local convergence is reached by a computing peer, a message is sent to the convergence manager peer; a local convergence corresponds to the fact that the uniform norm of the difference between two successive updates is less than a given threshold, fixed here to $10^{-6}$. This last manager peer stops the iterative process when all the computing peers have reached convergence then the next time step is performed.

When all time steps are performed, the computation is ended; in this case, a final notification message is sent by the computing peers to the convergence manager peer. Note that the principle of the centralized stopping criterion is the same as well as for the synchronous algorithm than the asynchronous one; nevertheless, the only difference between synchronous and asynchronous convergence detection lies in the way that the convergence manager answers to computing peers. In the synchronous case, the convergence manager waits for the convergence message of all computing peers at each relaxation; while in the asynchronous case, the answer is sent dynamically after receipt of any convergence message.

## VI. PEER-TO-PEER EXPERIMENTS

Computational schemes have been implemented in C language as a parallel algorithm using the P2PDC environment. Computational experiments have been carried out on the Grid5000 platform. This French grid platform is composed of 2970 processors with a total of 6906 cores distributed over 9 sites in France. 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 site of Grid5000 have several clusters with different performances.

Table I displays the characteristics of the machine used in the computational experiments.

| Site | Cluster | Processors Type | Speed GHz | CPU | Core | RAM |
|---|---|---|---|---|---|---|
| Lille | ChinqChint | Intel Xeon E5440 QC | 2,83 | 2 | 8 | 8 |
| Orsay | Gdx | AMD Opteron250 | 2,4 | 2 | 2 | 2 |

TABLE I. CHARACTERISTICS OF MACHINES ON EACH SITE.

We have considered a cubic domain Ω contained in the 3D space, Ω being discretized with $S = s^3$ points where s=256 denotes the number of points considered on each edge of the cube. The iterate vector is decomposed into $m = s^2$ sub-blocks with s components. A set of sub-blocks is assigned to each node and is updated using a sequential Gauss-Seidel block relaxation like algorithm performed in the lexicographical order. Concerning the time marching scheme, only three time steps are considered.

The results of the sequential computational experiments are summarized in Table II.

| European Options | | | | American Options | | | |
|---|---|---|---|---|---|---|---|
| Lille | | Orsay | | Lille | | Orsay | |
| Time | Relax | Time | Relax | Time | Relax | Time | Relax |
| 7669 | 1004 | 12668 | 1004 | 8534 | 1108 | 14362 | 1108 |

TABLE II. ELAPSED TIME AND RELAXATIONS WITH SEQUENTIAL ALGORITHM ON EACH SITE.

The parallel computational experiments are summarized in Table III, IV, V and VI in which 2, 4, 8, 16, 32, 64 and 128 computing machines are used (not including the convergence manager peer). In Table III and in Table V the number of relaxations is mentioned; note that for only the asynchronous experiments, Min, Max and Average number of relaxations are taken over the processors.

| | Asynchronous | | | | Synchronous | |
|---|---|---|---|---|---|---|
| Peers | Time/s | Relaxations | | | Time/s | Relaxations |
| | | Min | Max | Average | | |
| 2 | 5492 | 996 | 1439 | 1217 | 4904 | 1004 |
| 4 | 3158 | 1019 | 1695 | 1353 | 3158 | 1015 |
| 8 | 1289 | 995 | 1572 | 1309 | 1370 | 1015 |
| 16 | 515 | 1010 | 2363 | 1502 | 589 | 1030 |
| 32 | 195 | 822 | 1866 | 1486 | 297 | 1035 |
| 64 | 90 | 1074 | 1823 | 1478 | 292 | 1053 |
| 128 | 65 | 949 | 2316 | 1380 | 303 | 1091 |

TABLE III. ELAPSED TIME AND AVERAGE RELAXATIONS FOR EUROPEAN OPTIONS.

| | Asynchronous | | Synchronous | |
|---|---|---|---|---|
| Peers | Speedup | Efficiency | Speedup | Efficiency |
| 2 | 1.40 | 0.70 | 1.56 | 0.78 |
| 4 | 2.43 | 0.61 | 2.43 | 0.61 |
| 8 | 5.95 | 0.74 | 5.60 | 0.70 |
| 16 | 14.89 | 0.93 | 13.02 | 0.81 |
| 32 | 39.39 | 1.23 | 25.82 | 0.81 |
| 64 | 85.21 | 1.33 | 26.26 | 0.41 |
| 128 | 117.90 | 0.92 | 25.31 | 0.20 |

TABLE IV. SPEEDUP AND EFFICIENCY FOR EUROPEAN OPTIONS.

For the considered application and architecture used, when the number of machines is greater than 8 for European options and 4 for American options, the asynchronous scheme of computation, scales better than the synchronous one. For example, with 128 peers, the asynchronous computation scheme clearly performs about five times (European options) and about three times (American options) better than the synchronous one. This feature shows that asynchronous algorithms are less sensitive to granularity network latency. Moreover the asynchronous algorithms do not generate idle time. Therefore load balancing is not necessary for decreasing the elapsed time.

| | Asynchronous | | | | Synchronous | |
|---|---|---|---|---|---|---|
| Peers | Time/s | Relaxations | | | Time/s | Relaxations |
| | | Min | Max | Average | | |
| 2 | 7169 | 1106 | 1862 | 1484 | 6526 | 1123 |
| 4 | 3209 | 1161 | 1709 | 1431 | 3681 | 1138 |
| 8 | 1464 | 1096 | 1720 | 1400 | 1659 | 1138 |
| 16 | 581 | 1153 | 2574 | 1748 | 626 | 1138 |
| 32 | 285 | 1069 | 2157 | 1712 | 361 | 1145 |
| 64 | 102 | 495 | 1469 | 1049 | 318 | 1165 |
| 128 | 109 | 1284 | 2685 | 1704 | 337 | 1208 |

TABLE V. ELAPSED TIME AND AVERAGE RELAXATIONS FOR AMERICAN OPTIONS.

| | Asynchronous | | Synchronous | |
|---|---|---|---|---|
| Peers | Speedup | Efficiency | Speedup | Efficiency |
| 2 | 1.19 | 0.59 | 1.31 | 0.65 |
| 4 | 2.66 | 0.66 | 2.32 | 0.58 |
| 8 | 5.83 | 0.73 | 5.14 | 0.64 |
| 16 | 14.67 | 0.92 | 13.63 | 0.85 |
| 32 | 29.94 | 0.94 | 23.64 | 0.74 |
| 64 | 83.72 | 1.31 | 26.84 | 0.42 |
| 128 | 78.29 | 0.61 | 25.32 | 0.20 |

TABLE VI. SPEEDUP AND EFFICIENCY FOR AMERICAN OPTIONS.

In both cases of synchronous or asynchronous scheme, the speedup of computation is calculated using the shortest sequential elapsed time (see Table II); note that on a heterogeneous architecture the notion of speedup and efficiency is inappropriate, but this is a good indicator of performance. Nevertheless, note that the obtained values of speedup are high, particularly for the asynchronous scheme with large number of machine. Note also that super-linear acceleration has been obtained for asynchronous schemes of computation but not in the synchronous case. This feature shows that the communication overhead induced by the parallelisation is small enough, so that positive effects of parallelism on memory management can be seen. Indeed, the more peers are used, the less data is involved in the computation on each single peer. Therefore, the proportion of data that fits in cache memory is higher when granularity is fine.

The efficiency of asynchronous scheme of computation is better compared to the efficiency of synchronous scheme of computation. Note that for the considered size of the algebraic system to solve, 128 peers are sufficient.

## VII. CONCLUSION

In the presented study, for the solution of the evolution European and American options derivatives, we have studied the use and the implementation of parallel synchronous and asynchronous iterative algorithm [18] on an environment of peer-to-peer architecture. It follows from the computational experiments that the choice of communication mode (synchronous or asynchronous) has important impact on the efficiency of the distributed methods. The computational results show that the use of P2PDC carried out on the Grid5000 platform permits one to obtain good efficiency, particularly when the number of processors is large except for the synchronous method when the number of processors is greater or equal to 64. Moreover, using parallel iterative asynchronous methods, compared to the synchronous ones, is well adapted to the use of heterogeneous and distant distributed large number of machines. Finally, the difference of performance is mainly due to the weight of synchronisations between the processors.

## VIII. ACKNOWLEDGMENT

Part of this study has been made possible by ANR grant: ANR-07-CIS7-011 and support of Grid5000.

## REFERENCES

[1] Paul Wilmott, Jeff Dewyne, Sam Howison, Option pricing - mathematical models and computation - Oxford financial press – 1993.

[2] P. Jaillet, D. Lamberton, B. Lapeyre, “Variational inequalities and the pricing of american options”, Acta Applicandae Mathematicae, vol. 21, pp. 263 – 289, 1990.

[3] D. El Baz, G. Jourjon, “Some solutions for Peer to Peer Global Computing”, in *Proceedings of 13th Euromicro conference on Parallel, Distributed and Network-Base Processing*, 2005, pp. 49-58.

[4] P. Spitéri, M. Chau, “Parallel asynchronous Richardson method for the solution of obstacle problem” in *Proceedings of the 16th Annual International Symposium on High Performance Computing Systems and Applications*, pp. 133-138, 2002.

[5] T.T. Nguyen, D. El Baz, P. Spiteri, G. Jourjon, M. Chau, “High Performance Peer-to-Peer Distributed Computing with Application to Obstacle Problem”, in *Proceedings of IEEE IPDPS 2010 Conference,* Atlanta, 2010.

[6] L. Giraud, P. Spiteri, “Parallel resolution of non-linear boundary value problems”, Mathematical Modeling and Numerical Analysis, vol. 25, n° 5, pp. 579-606, 1991.

[7] J.C. Miellou. P. Spiteri, “A criterion of convergence for general fixed point methods”, Mathematical Modeling and Numerical Analysis, vol. 19, pp. 645-669, 1985.

[8] J.C. Miellou, D. El Baz, P. Spiteri, “A new class of asynchronous iterative algorithms with order interval”, Mathematics of Computation, vol. 67, n° 221, pp. 237-255, 1998.

[9] D. El Baz, T.T. Nguyen, “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.

[10] Aumage, G. Mercier, "MPICH/Madeleine: a True Multi-Protocol MPI for High Performance Networks", *15th International Parallel and Distributed Processing Symposium* (IPDPS'01), 2001.

[11] David P. Anderson, “BOINC: A System for Public-Resource Computing and Storage”, *5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, USA, 2004.

[12] N. Andrade, W. Cirne, F. Brasileiro, P. Roisenberg, “OurGrid: An approach to easily assemble grids with equitable resource sharing”, in

*Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 61-86, 2003.

[13] J.M. Ortega, W. Rheinboldt, Iterative solution of nonlinear equations in several variables, Academic press, 1970.

[14] Matti A. Hiltunen, “The Cactus Approach to Building Configurable Middleware Services”, in *DSMGC2000*, Nuremberg, Germany, 2000.

[15] G.T Wong, M.A Hiltunen, R.D Schlichting, “A configurable and extensible transport protocol” in *Proceedings of IEEE INFOCOM ’01*, Anchorage, Alaska, pp. 319–328, 2001.

[16] D.P. Bertsekas, J.N. Tsitsiklis, Parallel and distributed computation: numerical methods, in Prentice Hall, Englewood Cliffs, N.J., 1987.

[17] J.M. Bahi, S. Contassot-Vivier, R. Couturier, Parallel iterative algorithms: from sequential to grid computing, in Chapman & Hall/CRC, 2007.

[18] Pierre Spiteri, "Parallel optimization and financial mathematics", in *Proceedings of COSI'09 and Microsoft summer school*, Annaba, pp. 257 - 270, 2009.