An Equilibrium Pricing Model for Large Scale Computational Markets

Lilia Chourou, Mohamed Jemni  
LaTICE laboratory  
Higher School of Sciences and Techniques of Tunis  
University of Tunis  
Tunis, Tunisia  
lilia.gherir@planet.tn  
mohamed.jemni@fst.rmu.tn  

Ahmed Elleuch  
CRISTAL laboratory  
National School of Computer Sciences  
University of Manouba  
Manouba, Tunisia  
Ahmed.Elleuch@ensi.rnu.tn

Abstract—Load balancing is one of the most challenging problems facing large-scale computational systems. This paper introduces a competitive pricing model to achieve a general equilibrium between the global supply and demand in large scale computational commodity markets. Prices are adjusted according to a tâtonnement like process. A fully distributed pricing model is proposed based on an existing partially distributed version. While in this last version, the price of each commodity is computed by only one auctioneer, in our algorithm, a scalable number of auctioneers is used to support a massive number of suppliers and consumers.

Keywords: pricing, competitive equilibrium, scalability, computational commodity markets.

1. INTRODUCTION

The broad deployment of the internet has led to a massive number of internetworked computers. These computers represent a big underutilized reserve of heterogeneous computing resources. Many internet computing systems exploit these unused resources to reach high computing throughputs. It is rather common that volunteers donate these unused computational resources. It is through the introduction of economic incentives that it would be possible to scale-up much more the number of suppliers as well as the number of consumers, by embedding a strategy for welfare.

Buyya and al. have described different economic models for the management of resources in a grid [4] [5]. Among these models, we have adopted the commodity market model where, in contrast with the other models, prices are continuously adjusted to bring the global supply and demand back into equilibrium.

However, existing pricing algorithms are not suited for large scale computational markets. To overcome this drawback, we propose a scalable pricing model. Our model uses a partially distributed solution proposed by Cheng and Wellman [6] which is founded on the principle of the centralized tâtonnement process of L. Walras [12]. In [6], the price of each commodity is computed by only one auctioneer. There is one-to-one correspondence between markets and commodities. However, this distributed solution is not suited when there is a massive number of suppliers and consumers. On the other hand, our fully distributed solution uses a variable number of auctioneers per commodity. To each auctioneer is associated a limited number of suppliers and consumers. A proximity relationship is used, each auctioneer communicates with the suppliers and consumers of its region. And to build the global supply and demand, it communicates with the other auctioneers associated to the same commodity and situated in other regions. The scalability of the pricing algorithm is a fundamental property ensured by our algorithm.

To evaluate the effectiveness of our algorithm, two modelling and simulation case studies have been carried out. In the first one, the behaviour of the actors is modelled by utility functions commonly used in micro-economics and in several works implementing market mechanisms. These functions fulfill the convexity of the preferences and the gross substitutability assumptions needed to ensure the convergence of the tâtonnement process [2] [3]. For this first case study, the simulation results show that our distributed algorithm converges. Furthermore, it accelerates remarkably the convergence to a general equilibrium. In the second case study, the convexity of preferences is relaxed and the behaviour of the actors is modelled by more realistic utility functions. As the preferences are not convex, the demand functions are discontinuous. Consequently, if the equilibrium price does not exist, our pricing algorithm is able only to determine the two nearest prices leading to a change of the sign of the excess demand. The simulations results show that these prices will have very close values. Consequently we may consider these prices as equilibrium prices. Like in the first case study, the simulation results show that our distributed algorithm converges as well.

The remainder of this paper is organized as follows. Section 2 describes the most representative pricing algorithms and shows their limits in a large-scale system. Section 3 outlines the proposed computational market model that supports our scalable pricing algorithm. This algorithm is then presented in section 4. Section 5 provides an empirical evaluation of the proposed algorithm. Finally, section 6 presents the conclusions and future work.

II. RELATED WORK

The WALRAS algorithm proposed by Cheng and Wellman [6] is based on the tâtonnement process of Walras. However, they associate to each market one auctioneer. Another important feature of the WALRAS algorithm is that the announcement of offers and demands is asynchronous. Suppliers and consumers are
not necessarily negotiating the same goods at the same time using the same price information. Cheng and Wellman [1] investigated under the hypothesis of convex preferences and gross substitutability between goods, that the pricing process generated by their algorithm converges to a unique equilibrium. However, when each good is considered separately, the price adjustment process remains centralized and is performed by a dedicated auctioneer. The auctioneer is then overloaded if the number of suppliers and consumers is massive.

In [13], C. Weng et al. proposed to associate one auctioneer to a group of resources which their prices are strongly correlated. They use the tâtonnement process to reach equilibrium. The authors show experimentally that their group pricing algorithm converges faster than the WALRAS algorithm. Besides, the authors propose to organize the resources of the grid into autonomous administrative domains. Each domain is governed by an agent responsible for gathering information associated to its domain and submitting the excess demand information to the auctioneer. The organisation into domains addresses the scalability problem but it resolves only the aggregation of the excess demand, the price adjustment process is still centralized.

Wolski et al. proposed the system "First bank of G" [14]. They use an adaptation of Smale’s method [10] that aims to determine a trajectory for prices to find the equilibrium. To address the scalability problem due to the periodical interrogation of the suppliers and consumers, Wolski et al. propose to approximate each excess demand function by a polynomial used when the prices get closer to equilibrium. Wolski et al. consider complementary resources (like CPUs and disk storage). So the gross substitute property is not verified and then the uniqueness of equilibrium is not guaranteed. Besides, this system is still based on a centralised approach which is not scalable.

Stuer et al. propose a pricing algorithm of substitutable resources [11]. They limit themselves to two categories of substitutable resources: slow and high speed CPUs. A consumer determines for each CPU category a factor of preference according to its speedup, its price and a weight given by him to this category. This factor is then used to fix demand. The Smale method is also used and adapted to substitutable goods context. The simulations show that the proposed algorithm converges. But the scalability problem was not addressed and the pricing algorithm remains centralized.

In [17], X. Zhao and al. propose a distributed resource pricing mechanism. It allows each provider to adjust his resource prices according to a future demand prediction. The chains of Markov are used to predict future demands. The strategy of providers is to balance the resource load and to maximize its profits. The proposed pricing model is scalable, but this approach provides incentives for providers more than the consumers. The convergence of the price-adjusting mechanism toward a general equilibrium is not guaranteed.

A hierarchical approach is used in the distributed pricing algorithm called COTREE (COMBINATORIC TREE) [1]. The different actors in the system, called agents, are organized according to a logical tree. These agents are classified into two categories: the suppliers and the consumers considered as leafs of the tree, and the auctioneers. Each auctioneer is responsible for the aggregation of requests of its children (suppliers, consumers and possibly auctioneers). The simulation results show that COTREE requires significantly less messages to exchange. Besides, the pricing algorithm is scalable, but a failure of tree root is fatal.

Our algorithm uses the work of Cheng and Wellman [6] which is based on the basic concepts of general equilibrium theory. The proposed asynchronous bidding protocol is retained. To resolve the scalability problem, we consider a variable number of auctioneers for each market. The demand is aggregated, like in the COTREE system, according to a distributed model rather than a hierarchical model. Each auctioneer builds then a demand function which is used to determine the equilibrium. To avoid a frequent polling that would slow down the pricing process, we approximate, like in [14], each demand function by a polynomial. Besides, to accelerate the price convergence, the search space is divided among the auctioneers.

III. THE DISTRIBUTED COMPUTATIONAL MARKET MODEL

To find equilibrium between supply and demand in commodity markets, the economy must fulfill some hypotheses and specially those of pure and perfect competition, which are typically not satisfied by conventional markets. As we have described in [7], for computational markets, these hypotheses can be established in an artificial way. Besides, K. Arrow et al. [2] provide a proof of general equilibrium stability when all goods are gross substitutes. This property is verified if an increase in the price of a good increases the demand for another good.

However, pricing is achieved by a central auctioneer in a commodity market model. To resolve the scalability problem, for each commodity, we use a variable number of auctioneers called super-peers. A proximity relationship between a super-peer and its peers must exist to reduce the communication delays. The set of super-peers that belong to the same region is denoted \( S^l \) \( (1 \leq l \leq R) \). The total number of good types is denoted \( K \), and \( R \) is the total number of grid regions. The set of super-peers related to the same market, and so to the same type of good \( g \), is denoted \( S^g \). \( sp^g \) is the super-peer in the region \( l \) associated with the good \( g \). To a super-peer is associated a limited number of suppliers and consumers that we call peers. The set of peers associated to the super-peer \( sp^g \) is denoted \( PE^g \).

Figure 1 illustrates the proposed model with only two types of goods distributed into three regions. For the good \( 1 \), \( S^1 \) includes then three super-peers: \( sp^1 \), \( sp^2 \), and \( sp^3 \). For reasons of clarity, only the communications made by the super peer \( sp^g \) are depicted.

A partial equilibrium is reached when a super-peer \( sp^g \) finds a clearing price of a good \( g \), then it announces...
this price to all the other super-peers that belong to $SP_{g}$. But the tâtonnement process can exhibit a slow convergence to equilibrium [15]. To speed up the price adjustment, the searching space is divided into contiguous subspaces explored separately and in parallel by the super-peers of $SP_{g}$. Each super-peer collects and aggregates the preference information, for an announced price, from its peers and also from super-peers associated to the same good and located in other regions. The general equilibrium is reached when prices no longer change, so each super-peer needs also to continuously receive, from the other super-peers of his region, the current clearing prices of the other goods. Using the same approach as in the WALRAS system, asynchronous communication is used between peers and super-peers. At any point of time, a peer uses the last received prices and a super-peer uses the last received preference information. The detailed algorithm of the price adjustment is described in section 4.

To guarantee that pricing algorithm converges to a unique equilibrium price vector, we need to fulfill the hypothesis of gross substitutes. Complementary commodities, like CPUs and disk storage used in [14], are then not suitable. In [11], the retained approach is to build markets with substitutable commodities by clustering computing nodes into a finite set of resource categories such as high or low speed nodes. Instead of trading computing nodes, we propose to build substitutable commodities at higher level of abstraction. These commodities are services as defined in service oriented architecture. The determination of all the services is an important issue in a real global computing system. However, in this paper, we are mainly concerned with the elaboration and the pricing model. Nevertheless, to illustrate the approach of substitutable services and to evaluate the pricing algorithm for such computational commodities, we have developed a limited case study where goods are services remotely accessible and useful for matrix computation. A commodity is defined by the type of matrix operation and the matrix size. The substitutability between these goods is described in [7].

A global demand $d_{global}^{g}(p_{g}^{i}(t))$ calculated by a super-peer $SP_{g}^{i} \in SP_{g}$ for a price $p_{g}^{i}(t)$ at an instant $t$, is the sum of the local accumulated demand $d_{local}^{g}(p_{g}^{i}(t))$ of his associated peers and the not local accumulated demands $\tilde{d}_{local}^{g}(p_{g}^{i}(t))$ for every super-peer $SP_{g}^{j} \in SP_{g} \setminus \{SP_{g}^{i}\}$. The expressions (1) and (2) represent the global excess demand function which is used to search the equilibrium price.

\[
\begin{align*}
&d_{global}^{g}(p_{g}^{i}(t)) = \sum_{j \neq i} d_{local}^{g}(p_{g}^{i}(t)) + \sum_{j \neq i} \tilde{d}_{local}^{g}(p_{g}^{i}(t)) \quad (1) \\
&d_{local}^{g}(p_{g}^{i}(t)) = \sum_{j \neq i} d_{local}^{g}(p_{g}^{i}(t)) + \sum_{j \neq i} \tilde{d}_{local}^{g}(p_{g}^{i}(t)) \quad (2)
\end{align*}
\]

where $d_{i,g}^{j}$ is the demand announced by a peer $p_{g}^{j}$ to the super-peer $SP_{g}^{i}$. Note that $d_{i,g}^{j}$ depends on the price $p_{g}^{i}(t)$ of the good $g$ and also on the partial equilibrium prices $p_{g}^{j}(t-s_{j}^{g}(t))$ of the other goods, noted by $\sim g$. Each of them is updated since a delay period $s_{j}^{g}(t)$ before $t$.

If the price $p_{g}^{i}$ computed by a super-peer $SP_{g}^{i} \in SP_{g}$ is a clearing price, it will be announced to all peers $p_{g}^{j} \in PE_{g}^{i}$ and all super-peers in $SP_{g}$. Consequently, they stop their price search process

IV. THE PRICING ALGORITHM

Our algorithm uses the work of Cheng and Wellman [6] where the competitive equilibrium is computed via a distributed tâtonnement process (the WALRAS algorithm). While in this process, only one auctioneer is used per type of good $g$, in our algorithm a set of super-peers $SP_{g}$ is deployed. To ensure the convergence in the same way as the proof given in [6], the behavior of each set of super-peers $SP_{g}$ must be equivalent to a single super-peer per type of good $g$. Note that when the size of $SP_{g}$ is equal to one for all types of goods, our algorithm corresponds to the WALRAS algorithm.

A. Super-Peer Algorithm

A super-peer is directly involved in the pricing of only a single good. The pricing algorithm applies the principle of the tâtonnement process of Walras. Whereas, according to this process and for each adjusted price, the auctioneer requests from consumers (respectively, producers) to provide their demand (respectively, supply). In our solution, a super-peer submits different potential prices and then receives the corresponding proposed quantities, i.e. preference information. It aggregates this information, received from all peers of its region, and builds an approximated local excess demand function.

A super-peer is directly involved in the pricing of all the other goods, noted by $\sim g$. Consequently, they stop their price search process.

---

1 The term demand will be used indifferently to refer to the supply or the demand.
and adopt the price \( p^l_g \). A partial equilibrium for the good \( g \) is then reached. This price \( p^l_g \) is also announced to the super-peers, of the other goods, in \( SP^l \). A general equilibrium is reached when prices of all goods do not change anymore. So before reaching a general equilibrium, several partial equilibrium prices are computed for each good. Indeed, while a super-peer adjusts the price of a good, the prices of the other goods may change.

The demand of a peer changes during time since it depends on prices of all goods and these prices also change from a partial equilibrium to another one. Like in the WALRAS system and without compromising the convergence of the pricing algorithm [6], a peer does not need to have the most recent price of every good; instead, it uses the last received value. Similarly, a super-peer uses the last preference information received from each peer. Variable communication delays are tolerated. Peers and super-peers exchange data asynchronously.

To speed up the search of a partial equilibrium price of a good \( g \), the search space is divided into intervals. These intervals are explored in parallel by separate super-peers. The maximum price is limited according to the endowment of the consumers. Every super-peer \( sp^l \) limits its search to an interval \( I_g \). A price is calculated according to the previous price and to the global excess demand which is multiplied by a step-size. A super-peer \( sp^l \) calculates its price \( p^l_g \) as following:

\[
p^l_g = p^l_g + \lambda_g^l d_{global}^l.
\]  

(3)

As in [9], the determination of the step-size \( \lambda^l_g \) is like a binary search. \( \lambda^l_g \) is divided by 2 if the global excess demand changes sign and its absolute value does not decrease significantly (it means that the equilibrium price was exceeded). Otherwise and if \( \lambda^l_g \) is not greater than 1/2 then it is multiplied by 2.

The complete algorithm of a super-peer is given in [7].

B. Peer Algorithm

The peers keep a price vector of dimension \( K \) where each element of this vector is the last received equilibrium price of a specific good. When all prices of this vector are clearing prices then the exchanges can take place.

Besides and for a good \( g \), each peer receives periodically a potential prices vector from a super-peer \( l \in SP^l_g \) of his region. Consequently, a peer \( pe_i \) calculates for each price \( p^l_g(t) \) (in the potential prices vector) a new demand \( d^l_{i,g} \) defined by:

\[
d^l_{i,g}(p^l_g(t), p^l_g(t-s^l_{i,g}(t))) = x^l_{i,g}(p^l_g(t), p^l_g(t-s^l_{i,g}(t))) - e^l_{i,g}
\]  

(4)

if \( pe_i \) is a consumer else,

\[
d^l_{i,g}(p^l_g(t), p^l_g(t-s^l_{i,g}(t))) = -y^l_{i,g}(p^l_g(t), p^l_g(t-s^l_{i,g}(t)))
\]  

(5)

if \( pe_i \) is a supplier. \( x^l_{i,g} \) (respectively \( y^l_{i,g} \)) is the optimal demand for the good \( g \) requested (respectively produced) by the peer \( pe_i \) given the potential price \( p^l_g(t) \). The value of \( x^l_{i,g} \) (respectively \( y^l_{i,g} \)) is determined by a maximization program resolution as described in the section 5. The demand depends also on the price of the other goods \(-g\). For the peer \( pe_i \), \( e^l_{i,g} \) is the initial endowment of the good \( g \). The complete algorithm of a peer is given in [7].

V. SIMULATIONS AND RESULTS

In order to evaluate our pricing algorithm, we have carried out two modelling and simulation case studies. In the first one, consumer preferences are represented by commonly used utility functions that fulfill the hypotheses of convexity and gross substitutability needed to ensure convergence. In the second case study, the convexity of preferences is not preserved. Nevertheless, the results of the simulations show that this last condition may be relaxed and the pricing algorithm still converges. In the two cases, a limited computational market is modeled where commodities correspond to a representative set of computing services needed for matrix computation. We consider only square matrices, their sizes are: \( N \times N \) and \( 2N \times 2N \). The possible matrix operations are addition and multiplication. We assume that all commodities work at the same speed \( V \), but our model can easily be extended to more than one speed resulting in more commodities. As simulations are very time consuming, we consider only four commodities. Each of them is defined by a type of operation and a matrix size. As we have showed in [7], these services are substitutable commodities and the degrees of substitutability can be used to express the utility functions.

A. The First Case Study: Convex Preferences

The preferences of a consumer \( pe_i \) are represented by a CES (Constant Elasticity of Substitution) utility function. This function is commonly used in several works implementing market mechanisms [6] [8] [16]. The form of this function is given by the following expression:

\[
utility(t) = \left( \sum_{g} \alpha_g(x^l_{i,g}(t))^\rho \right)^\frac{1}{\rho}.
\]  

(6)

Where \( \alpha_g \) is the substitutability’s degree of the commodity \( g \) and \( \rho \) is a generic parameter. To ensure the convexity of preferences, the range of \( \rho \) must be restricted to the interval \([-\infty, 1]\). As in WALRAS algorithm [6], we set \( \rho \) to 0.5.

A peer \( pe_i \) has an initial endowment, in particular, the currency is considered as being a good and every consumer has an initial monetary budget which will be periodically refreshed. The initial endowment of the peer \( pe_i \) for the good \( g \) is denoted \( e^l_{i,g} \). This quantity can be sold or exchanged on the market (so a consumer may also be a supplier). In this exchange economy, the
value of the initial endowment of a peer is its wealth. The wealth of a peer $p_i$ at the instant $t$ is defined by:

$$ p(t)^* e_i = \sum_{g=1}^{K} p_g(t)^* e_{i,g}. $$

(7)

Each consumer aims to determine an optimum demand which maximizes, under a budgetary constraint, the quantity of computing to be performed. So a consumer $p_i$ must resolve the following maximization problem:

$$ \max utility(t_i) \text{ under constraint} $$

$$ \sum_{g=1}^{K} p_g(t)^* x_{i,g}(t) \leq \sum_{g=1}^{K} p_g(t)^* e_{i,g} $$

(8)

where $x_{i,g}$ $(1 \leq g \leq K)$ is the demand announced by the consumer $p_i$ for the commodity $g$.

We have implemented a discrete event simulator that simulates the computational market model and the pricing algorithm proposed in the previous sections. Fifty computational economies and fifty trials have been generated each with four commodities, sixteen consumers and sixteen suppliers. The suppliers (respectively, consumers) were given randomly-generated memory size that limits the feasible production (respectively, currency endowments, so each consumer have only currency as good). These economies are used to average each point in the following comparison charts. Vertical lines at the top of a bar indicate the minimum and the maximum values for the fifty trials.

To simulate the asynchronous behavior of peers, we assume that the number of new demands, some bids for a subset of commodities, follows a random draw. When a peer does not submit a new demand for a commodity, the super-peer considers its demand from the last iteration. Each super-peer, upon receiving the demands, computes the clearing price. Then it notifies its peers (the bidders) the new price. Peers do not possess the same state of price information on which they compute their next demands. Our developed simulator follows iteratively this process on each super-peer until the equilibrium prices are reached. For each iteration, a random number of super-peers compute new clearing prices. One iteration is called a cycle.

The simulations results show that our distributed algorithm converges to a unique equilibrium for each generated economy. Moreover, we have varied the number of super-peers per commodity and have measured the number of cycles and the number of explored prices to achieve the general equilibrium. Since the super-peers work in parallel, at each cycle the number of considered prices is taken as the maximal value among all super-peers. Figure 2 gives the average number of cycles and the average number of computed prices to reach the general equilibrium, for different numbers of super-peers associated with the same commodity.

Recall that the WALRAS algorithm corresponds to the case where only one super-peer is used per commodity. It should be noted that increasing the number of super-peers does not result in a worse number of cycles. Moreover, a significant improvement of the average number and the maximal number of cycles is observed. Hence, the asynchrony of the bidding process – implying that new partial equilibrium prices are computed according to not fully updated information – is not emphasized by using more than one super-peer per commodity. Besides, both average and maximal number of computed prices decreases significantly when the number of super-peers is increased. Indeed, a parallel search is performed and the search space assigned to each super-peer is reduced by increasing the number of super-peers per commodity.

**B. The Second Case Study: Linear Preferences**

Let us consider the simple case where preferences of consumers are represented, at the time $t$, by a linear utility function as described by:

$$ utility(t) = \left( \sum_{g=1}^{K} \alpha_g x_{i,g}(t) \right) $$

(9)

Like in the first case, each consumer aims to determine an optimum demand under the budgetary constraint as described by the expression (8). As the preferences are not convex, the demand functions are discontinuous. Consequently, if the equilibrium price does not exist, our pricing algorithm is able only to determine the two nearest prices leading to a change of the sign of the excess demand. If these two prices have close enough values, we may consider these prices as equilibrium prices. A general equilibrium is reached when prices of goods do not change anymore or oscillate between two close values.

\[ \text{Figure 2. Convex preferences case: (a) Average number of cycles.} \]

\[ \text{(b) number of estimated prices.} \]
For all simulations, we calculate the difference between oscillating prices and we found that these gaps are negligible compared to the corresponding prices. Like in the first case study, the simulation results show that our distributed algorithm converges well, as it is shown in figure 3. These results suggest that for linear preferences the pricing algorithm converges well to equilibrium and behaves as in the case of convex preferences.

VI. CONCLUSION AND PERSPECTIVES

In this paper, a distributed pricing model is proposed to achieve a competitive equilibrium. Prices are adjusted according to a tâtonnement like process. The search space is divided into intervals, explored by separate coordinated auctioneers, to speed up the convergence of the pricing algorithm. Although the convergence guarantee depends on several conditions, in particular the preference convexity assumption, we have found empirically that our algorithm converges even if the preferences are linear. As a future work, we intend to apply our market model and pricing algorithm to cloud computing systems where pricing will be unified and services will be fungible. We look also to include the trust granted to each announced offer or demand in the pricing process. A reputation mechanism will be used to evaluate the trust level of each bidding peer.

REFERENCES