RESOURCES PRICE ADJUSTMENT IN CLOUD COMPUTING FOR LOAD BALANCING AND FAIRNESS BETWEEN USERS AND SUPPLIERS

Lilia Chourou Gherir[1], Ahmed Elleuch[2] and Mohamed Jemni[1]

[1]LaTICE laboratory, Higher School of Sciences and Techniques of Tunis, University of Tunis, Tunis, Tunisia
[2] CRISTAL Laboratory, National School of Computer Sciences, University of Manouba, Manouba, Tunisia

ABSTRACT

Resource price adjustment is a challenging issue when building a cloud computing system. For this purpose, we present an approach that adopts a market-driven price adjustment model as an economic incentive model and ensures the balance between supply and demand. We show how this model may be adapted and applied to an infrastructure as a service (IaaS) in cloud computing. We place ourselves in the context of the virtual machines market. To achieve competitive equilibrium, prices are adjusted according to a tâtonnement-like process. The prices are computed by a third party, the auctioneer. However, this process requires several assumptions to ensure the existence and stability of competitive equilibrium, not all of which are verified by cloud computing systems. The atomicity and convexity of preferences are not guaranteed by cloud computing systems. These questions will be the subject of our empirical study. We find empirically that the change in the sign of excess demand, indicating the existence of equilibrium, occurs between two similar prices. The experimental results show that the proposed dynamic price adjustment approach is more profitable than fixed-price pricing, allowing dynamic suppliers to become important market agents.

KEYWORDS

Cloud Computing, Pricing, Virtual Machines Market, Competitive Equilibrium, Auctioneer.

1. INTRODUCTION

Cloud computing is an evolution of information technologies that involves outsourcing hardware and software resources by making them available on demand as services over the internet. Typically, these services' prices are set by their providers. The use of price adjustment mechanisms can therefore play a decisive role in the wider adoption of these systems. An appropriate economic model should therefore provide more incentives for customers, as well as for suppliers, to adhere to cloud systems. Market-oriented price adjustment approaches in cloud computing are currently the subject of several related studies. In this context, we propose a fair price adjustment model that considers aggregated supply and demand as key parameters. We adopt the general equilibrium theory introduced by the economist Léon Walras [1]. The interactions of economic agents, consumers and suppliers are coordinated by a third party, the auctioneer. The auctioneer is responsible for adjusting the price of all the markets to obtain an equilibrium between supply and demand.

We are adapting this solution to cloud systems [2] to support scaling up. Indeed, the solution that we propose is distributed and asynchronous. A market in a geographical region is governed by an auctioneer. All auctioneers communicate with each other to achieve general equilibrium.

However, certain assumptions must be verified to ensure the existence and uniqueness of the general equilibrium. The five conditions of pure and perfect competition must be verified to ensure the existence of general equilibrium. One of these conditions is the assumption of market atomicity: the market must include a large number of suppliers and consumers. Consequently, no single supplier or consumer can exert influence on the market alone. This assumption is not verified by cloud computing systems due to the limited number of suppliers. Under these conditions, and in the case where the equilibrium price is not reached, we determine a price that approaches the equilibrium price.

We place ourselves within the framework of the virtual machine market. Our interest is particularly the adjustment of the price of an elementary unit of computation. We model the behavior of subscribing suppliers and consumers by rational algorithms. The price offered by the auctioneer must cover the operational cost of servers, in particular, electricity costs. We assume that users work in the context of high-throughput computing with a single objective: the use of a maximum number of resources within their budget.

The remainder of the paper is organized as follows: in section 2, market-oriented price adjustment approaches proposed in cloud computing are reviewed. In section 3, we cite the concepts of the Walras tâtonnement process (French for "trial and error") as well as the necessary assumptions for the existence and uniqueness of the equilibrium price. In section 4, we present our dynamic price adjustment approach while describing the algorithms of economic agents, suppliers and users, as well as the auctioneer. Finally, several experiments are performed in section 5 to evaluate the effectiveness of our approach. Section 6 summarizes the paper.

2. RELATED WORK

Market-driven price adjustment approaches are attracting significant attention from many academics and industries working on cloud computing systems. These methods offer a number of advantages over fixed-price methods. They allow the price and allocation of resources in the cloud to be determined dynamically. In this section, we review the work that has already been done, and we identify the research gaps.

Spot instances of Amazon's EC2 service were introduced by Amazon in 2009 [3]. The first mechanism was to adjust prices dynamically in a Cloud. To acquire a spot instance, the customer announces a maximum price to be paid per hour. The request is accepted if the customer's price exceeds the price given by Amazon; otherwise, the request is put on hold. Periodically, Amazon announces a new price and launches all pending requests whose price is higher than the new price proposed. Amazon claims that prices vary periodically according to supply and demand. However, much research has been performed on this subject. We cite [4], **Invalid source specified.** and**Invalid source specified.**. All of these studies show that Amazon adjusts the price of spot instances according to a predefined strategy, independent of user demand. Orna Agmon Ben-Yehuda et al. demonstrated in [4] that prices vary according to a well-defined algorithm. On the basis of real price traces, they constructed an adjustment algorithm. Prices are governed by an AR (1) autoregressive process within a predefined interval. The simulation results show that the prices generated by the constructed algorithm match well with the prices present in the traces. In addition, inspired by spot instances, F. Alzhouri et al. [5] propose an approach for adjusting the prices of unused resources in cloud computing systems to maximize the supplier's revenue. The

price does not depend on customer demand. It follows an exponential function that depends solely on the supplier's available capacity at a given time.

Several works, such as [6], [7] and [8], are based on an auction mechanism.

S. Li et al. [6] applied the concept of auctions in which several users compete to obtain resources offered by a supplier. The user's objective is to maximize a utility function to achieve a high QoS by obtaining as many resources as possible while spending the least. The QoS corresponds to the response time obtained by the user. Each user announces a set of information, in particular a budget expressing the maximum monetary expenditure that the user is prepared to pay per unit of time and a desired response time interval without specifying a particular quantity of resources. The desired response time interval is elastic. Periodically, the provider adjusts its price according to the information provided by users. It should be noted that at a given moment, the price adjustment is based on requests received only at that time and not on a global request. The supplier sets the price according to an optimization programme with the aim of maximizing revenue while ensuring a minimum profit. This profit enables the supplier to recover operational costs and guarantee a minimum gain. Operational costs are assumed to come mainly from the cost of electricity consumption of the IaaS infrastructure. The minimum gain is calculated proportionally to this cost. The proposed provider's optimization program also ensures that each user has an optimal amount of resources that maximizes their utility. The total amount allocated must not exceed the capacity of the IaaS infrastructure. The authors showed that the optimization problem describing the supplier's behavior is NP-hard. They then propose a price search algorithm that approximates the provider's maximum revenue. Only users with a budget that can support the price set by the provider obtain the resources. The experimental results show that for low budget levels, a large number of users are sacrificed to reach the maximum revenue. This solution therefore favors the supplier. Furthermore, the existence of other competing suppliers is not taken into account, as this compromises the existence of such markets or at least limits their size.

L. Lu et al. [7] applied the second price auction principle. William Vickrey showed that this type of auction encourages bidders to be truthful in their price announcements [9]. The customer announcing the highest price wins the auction, but the price paid is the second highest price announced by all customers. The major disadvantage of auctions, even at the second price, is that the price may be much higher than the equilibrium price. Fairness is not ensured.

Shi et al. [8] propose an intelligent price adjustment mechanism allowing a Cloud provider in the presence of competing Cloud providers to maximize its long-term profit. The evolution of the market is described by an increasing function reflecting an increase in the number of users over time. The authors limit themselves to two competing providers, one applying the proposed strategy and the other a different strategy, notably a price reduction strategy. They assumed that suppliers would adopt an auction mechanism to offer their services. Based on the prices published by the suppliers, the user calculates an expected utility for each supplier. The expected utility of a given supplier at a given time is calculated as a function of a marginal value estimated by the user for a unit of resource requested at that time, the price advertised by that supplier and a preference value given by the user to that supplier. The user chooses the supplier that gives the maximum expected utility. After choosing the supplier, the user determines its price according to a feedback system that allows him or her to automatically adjust his or her price with the aim of winning the auction. It then announces its price and the required quantity. The supplier adjusts his price according to the prices announced by the users as well as the price of the competing supplier to maximize his profit. The price calculation also takes into account the cost incurred by the supplier during the production of a Cloud service. A marginal cost function is used. It represents the cost per unit of service at a given point in time. The problem is modeled as a

Markov game solved by a multiagent reinforcement learning algorithm (multiagent deep deterministic policy gradient - MADDPG). This algorithm is based on the actor-critic method, which allows the supplier to gradually learn the optimal pricing strategy that maximizes its profit. The trade-off between exploration and exploitation that characterizes the actor-critic method manifests itself. Exploitation involves using current information, while exploration involves gathering additional information to make the best decision. Exploiting too much can lead to nonoptimal policies, while exploring too much can unnecessarily slow learning. In addition, only users who have proposed prices higher than the price calculated by the supplier obtain the resources, which indicates that the proposed solution is rather in favor of the supplier. The approach adopted in this work considers that each agent, supplier or customer, proposes its own price. This approach runs counter to our approach, which consists of determining a common fair price that satisfies all the agents.

Other works are based on game theory, for example, [10], [11], [12], [13], [14], [15], [16] and [17].

In [10], J. Xu et al. proposed an approach for dynamic price adjustment of resources in a geodistributed computing system (fog computing). The price adjustment process is viewed as a Stackelberg game where the supplier plays the role of market leader in deciding the price, and each user plays the role of a follower and reacts accordingly. The influence of other suppliers in the market is not taken into account. Suppliers offer 2 types of resources: dedicated resources, i.e., fog servers, and nondedicated resources, such as routers. The price of dedicated resources differs from that of nondedicated resources. The authors are interested in the communication between a provider and a user. Initially, the provider announces a price for each of the two types of resources. It should be noted that the authors do not indicate how they set these initial prices. For the user, he announces quantities: according to the prices announced by the supplier, the user determines an optimal utilization rate for each type of resource that minimizes its expenses. The user's total workload is divided into units and distributed according to the utilized rates. Then, at each iteration, the supplier determines for each user and each type of resource the optimal price that maximizes its profit. The price announced to a user depends on the requested quantity, the total processing time and the electricity cost. The authors show through an empirical study only that the provider and a small number of users reach a Nash equilibrium. This work does not study the case of several competing suppliers and assumes that the supplier and the users are governed by well-known algorithms: maximizing the gain on the supplier's side and minimizing expenses on the user's side. Certainly, this is a rational behavior of the two actors, but other behaviors exist depending on the issues and the nature of the applications deployed.

X. Zhang et al. [11] aim to maximize social welfare, which is taken to be equal to the sum of a supplier's profit and the aggregate utility of customers. The authors limit themselves to a single supplier. Here again, the influence of other suppliers on the market is not taken into account. The supplier manages multiple data centers offering customers different types of resources (CPU, RAM, disk and bandwidth). Each customer requests a tailor-made VM by specifying the quantity needed for each type of resource. This concept is being adopted by several emerging Cloud providers, such as [18] and [19]. The customer demand is elastic. The same client issues different requests, called options. Each of them contains a start and end time for execution of the requested VM, the quantity needed for each type of resource, the data center and the proposed budget. Requests are assumed to occupy an amount of resources for a future duration of use. The supplier accepts at most one request from all requests issued by the same customer. The problem of maximizing social welfare is modeled by an optimization programme. The authors use Fenchel's duality to determine the demand and the server in the specified data center, which maximizes the utility of each user as well as the price of a unit of resource on each server that maximizes the provider's profit. Note that servers in a data center are assumed to be heterogeneous with different

operational costs and resource capacities. A user's utility depends on the prices previously set by the supplier on each server. The user's objective is to spend as little as possible. The price of a unit of resource on a server at a given time is calculated as a function of the predicted total demand on that server at that moment. If the predicted total demand is less than the capacity of the server, then the price of the resource is equal to its marginal cost; otherwise, the price increases exponentially as a function of the current demand to eliminate future demand at low prices. The marginal cost is the cost of a unit of resource at a given time. The authors assumed that the operational cost of a server is mainly due to the cost of electricity, which increases as the number of resources occupied on the server increases. They propose a general convex cost function representing different server operating models (cubic, linear or null examples) in the real world of cloud IaaS.

J. Zhao et al. [12] are interested in maximizing the profit of a Cloud provider through price adjustment and minimization of operating costs. The problem is modeled by a constrained optimization program based on customer requests. Each request contains a type of VM (which refers to the quantity of resources) as well as a reservation duration. Requests containing the same type of VMs and the same reservation duration are assumed to be of the same type. The resolution of the optimization program allows the provider to determine, at a given time and for each of its data centers, the appropriate price for each type of request and the optimal number of servers dedicated to each type of request to maximize its profit in the long term. The authors assumed that a server at a given time only hosts requests of the same type. They use a VM migration algorithm [20] within a data center (intradatacenter), allowing VMs to be moved from one server to another to reduce the number of active servers. Solving the optimization program also allows the provider to determine the optimal number of requests of each type to schedule and abandon. Abandoned requests are requests whose delay exceeds the latency that the provider must guarantee as an SLA. The provider processes so-called abandoned requests while incurring a certain penalty. To solve this problem, the authors use Lyapunov optimization, which is a technique for translating a long-term optimization problem into a series of optimization problems at each time interval. The price depends mainly on user demand and revenue that the supplier seeks to guarantee over the long term while ensuring an SLA to the customer. The influence of other suppliers on the market is not taken into consideration.

M. Aazam et al. [13] worked in the context of federated cloud systems coordinated by an auctioneer. The authors are interested mainly in the integration of customer reputation in determining the price of a service. Indeed, the price of a service for a given user is calculated depending on the price given by the supplier for this service and an average overall abandonment probability representing the services usage behavior followed by the user. An increase in the probability of abandonment increases the price of the service with the aim of minimizing the underutilization of resources and therefore ensuring the profit targeted by the provider. It should be noted that a supplier decides to register a service at the auctioneer only when the service is requested by a minimum number of customers presenting a total abandonment probability greater than a certain threshold set by the supplier. This work also focuses on the QoS delivered by the provider during the service. The supplier is subject to a reimbursement amount if the promised QoS, described through an SLA, is not respected. The authors apply their model to the 2 categories of instances, on-demand and reserved. A reputation mechanism is also necessary for our price adjustment mechanism to determine the degree of reliability of agents' announcements, customers and suppliers, which are used as input parameters in our price calculation algorithm. The integration of the reputation mechanism will be the subject of future work.

D. Rane et al. [14] propose a dynamic algorithm that gives each client a price interval for the required quantity of resources and a given period. Suppliers and customers communicate through a broker who is responsible for calculating price intervals. The authors suggest a demand function

that approximates the quantity demanded by a customer from a supplier at a given time. The bounds of a price interval announced to a customer are calculated from its demand function and predicted prices. The change in price at a given time is predicted using the proposed demand function. It should be noted that the capacity offered by the supplier is not involved in the price calculation. The aim of this work is to achieve an average price that is lower than the price set by the supplier for the same resource. The empirical study shows that a customer's expenses using the proposed algorithm are lower than the customer's expenses using Amazon EC2's spot service. The authors also find that the lower bound of the interval announced to the customer is higher than the prices given by Amazon over the study period considered, which allows the supplier to obtain a higher profit. However, setting a high price can lead to the loss of several customers, hence the need for an equilibrium price that takes into account both customer demand and supplier supply.

In , H. Peng et al. [15]worked in the context of preemptive cloud services, such as Spot VMs, which use the excess capacity available at a provider after satisfying high-priority services. They are particularly interested in unused CPU capacity. However, this capacity fluctuates stochastically over time. Cloud operators often disclose a maximum preemption rate to alleviate perceived uncertainty about interruptions. The dynamic price adjustment problem is formulated as a constrained nonstationary Markovian decision process. The authors decompose this process into a finite number of constrained stationary Markovian decision processes (CMDP). Then, each CMDP is transformed into a dual Lagrangian problem. The authors propose a new approach, Q-Learning (QL), which aims to determine the optimal Lagrange multiplier that maximizes the supplier's revenue generated by excess capacity under the constraint of the maximum preemption rate.

In , Y. Li et al. [16]used contract theory to construct the economic interaction between service providers and users. The authors consider several types of applications or types of users. For each type of user, the provider designs a contract that determines the set of resources and the associated price. Given a predicted demand, the supplier determines the optimal contract that maximizes its profit while maintaining user participation in the market. To attract users to participate in the market, the contract satisfies 2 constraints: an individual rationality constraint that guarantees the user positive utility and an incentive compatibility constraint that guarantees the user maximum utility for the contract chosen and the quantity demanded. The optimal contract problem is formulated as a profit maximization problem under these 2 constraints.

In , J. Chen et al. [17]proposed a price adjustment approach in which the seller is a provider of cloud/fog computing resources and the buyers are miners in a proof of work (PoW) blockchain system. The authors restrict themselves to a single supplier. They do not consider competition between providers but rather competition between miners who seek to make a profit from block mining. The objective of a miner is to maximize his utility, which is equal to the expected revenue minus the cost of the computing resources requested from the cloud/fog provider. Depending on the demand from miners, the provider adjusts its price to maximize its profit. Miners are faced with 2 strategies: to buy or not to buy cloud or fog computing resources. The authors showed that there are several stable states for miners. A stable state is defined as a state where none of the miners would like to change their strategy. The authors also show that for each stable state, there is an optimal price that maximizes the supplier's profit.

In summary, several studies have been carried out in the context of dynamic price adjustment and allocation of cloud resources. Different hypotheses and concepts have been used. However, to our knowledge, few works use the equilibrium price to equalize aggregate demand and aggregate supply to set the price of cloud resources and to ensure fairness between users and suppliers. Our

work is based on the tâtonnement process of Walras, who was the first to formulate the idea of general equilibrium [1].

3. CONCEPTS AND HYPOTHESES

Our approach is based on the tâtonnement process of Walras. The interactions of economic agents, consumers and suppliers are coordinated by a third party, the auctioneer. The auctioneer is responsible for adjusting the price of all the markets to obtain a general equilibrium. The basic model considers a single auctioneer. The auctioneer can set a potential price for consumers and suppliers. Each consumer responds with a quantity of demand. In addition, each supplier responds with a quantity of supply. The auctioneer then determines the excess demand, which is equal to the aggregate demand minus the aggregate supply. The equilibrium price is the price that cancels the excess demand.

To ensure the existence of an equilibrium price, the five conditions of pure and perfect competition must be satisfied on the market:

- Market atomicity: The market must include a large number of suppliers and consumers. Consequently, no single supplier or consumer can exert influence on the market. This assumption is not verified by targeted cloud computing systems. We therefore plan to empirically study the prices causing a change in the sign of excess demand for a small number of suppliers and consumers.

- Product homogeneity: All products with the same characteristics are considered to be identical. We propose a dynamic price adjustment algorithm for VMs. The characteristics of VMs include the CPU, RAM, disk and bandwidth. We consider that VMs with different characteristics are different products with different market prices.

- No barriers to market entry or exit: Suppliers and consumers are not constrained by technical, financial, regulatory or legal barriers to market entry or exit. In the case of cloud computing systems and unlike traditional markets, these constraints should disappear thanks to the internet, which is deployed on a global scale and breaks down such barriers.

- Transparency of information: Each agent must have access to global and complete information to make the best decisions that maximize their well-being. This must be guaranteed by an information service integrated into cloud computing systems.

- Mobility of factors of production (labor and capital): workers and capital must be able to move freely without obstacles from one activity to another. Once again, this is catalyzed by access to cloud computing systems via the internet.

K. Arrow and G. Debreu showed that these assumptions are not sufficient [21]. They add other conditions concerning consumer preferences and their endowment:

- Each consumer has a nonzero initial endowment.

- The demand function is convex. According to the principles of economics, it is capable of representing the law of demand: when the price of a good increases, the quantity demanded for this good decreases. The hypothesis of convexity of preferences is used to ensure that the demand function is well defined and continuous. In the context of cloud computing systems, [5] and [22] use a convex function representing user demand.

Under these conditions, K. Arrow and G. Debreu also showed that any general equilibrium in pure and perfect competition is a Pareto optimum. Furthermore, K. Arrow et al. [23] prove that the general equilibrium is stable when all products are gross substitutes. Two products are gross substitutes if an increase in the price of one product causes an increase in the demand for the other product.

In summary, the existence and stability of the general equilibrium require several assumptions, not all of which are verified by cloud computing systems. The atomicity and convexity of preferences are not guaranteed by cloud computing systems. These questions will be the subject of our empirical study.

4. SYSTEM MODELING

We place ourselves in the context of the virtual machines market. Our system supports two types of agents: users and suppliers of VMs. The actions taken by the users and suppliers are coordinated by an auctioneer. In this work, we consider a single auctioneer for simplification. We propose a distributed approach to the Walras model in [2]. To speed up the price adjustment process, we assume that the auctioneer emits a vector of potential prices. Each agent responds with a vector of quantities. The auctioneer then constructs an aggregate excess demand function used to determine the equilibrium price.

Each provider has a set of geographically distributed data centers of size D indexed by d, where $1 \leq d \leq D$. We assume that the data centers have identical servers. At time t, the provider has a number of servers in a data center d denoted by $S_d(t)$. Like the majority of the works described in section 2, we consider that the operating cost of servers mainly comes from the cost of power consumption. We ignore fixed charges such as material cost and others. The price of electricity differs according to the geographical location of the data center. A provider offers users a set of resources (CPU, RAM, disk and bandwidth) in VMs. IaaS providers generally offer preconfigured VM instances of fixed types. For example, Amazon EC2 currently provides 10 instance types within six instance families optimized for different application types [24]. We assume that each server has a number of elementary computation units or elementary VMs. The auctioneer then announces a vector of potential prices for an elementary computation unit per unit of time. The price is static for one unit of time. Suppliers are assumed to be rational and competitive agents. Based on the prices announced by the auctioneer, the supplier determines the optimal number of active servers for each of its data centers that maximize its total profit. The supplier calculates its expected profit for a dynamic potential price Ppd announced by the auctioneer in a data center d as follows:

$$profit^d(t) = (Ppd - c_d) \times N \times S_d \quad (1)$$

c_d is the operational cost of an elementary VM per unit of time, and N is the number of elementary VMs per server. We assume that the cost c_d is determined on the basis of the fixed operational cost of a server in a data center d when its power consumption is at its maximum: we assume that the supplier manages to occupy almost all the resources of each operating server by using mechanisms for migrating VMs within the same data center. The supplier offers all the servers if the expected profit covers the cost of running them and guarantees a minimum profit. Like S. Li et al. [6], we assume that the required gain per elementary VM per unit of time is calculated proportionally to its power cost per unit of time. At time t, the minimum price required per elementary VM per unit of time in a data center d is calculated as follows:

$$p_{min}^{d}(t) = (1 + r_d(t)) \times c_d \quad (2)$$

where $r_d(t)$ is the rate of gain required at time *t* per elementary VM per unit of time in the data center d. Consequently, the minimum profit required by the supplier to offer all its servers in data center d is determined as follows:

$$profit_{min}^{d}(t) = r_d(t) \times c_d \times N \times S_d \quad (3)$$

If the price announced by the auctioneer does not allow the supplier to recover both the operating cost and the required gain associated with all the servers, then the supplier activates a number of servers in proportion to the expected profit. In this case, if the announced price covers the operating cost, the number of active servers in a data center d at time *t* is calculated as follows:

$$s_d\,(t) = \left\lfloor \frac{profit^{d}(t)}{profit_{min}^{d}(t)} \times S_d \right\rfloor \quad (4)$$

Otherwise, the number of active servers is zero. The total quantity of elementary VMs offered by the supplier at time t noted *Q(t)* for each potential price announced by the auctioneer is given by Algorithm 1. We denote by *r(t)* and *S(t)* the vectors of the gain rates and the vectors of server numbers, respectively, in the various data centers at time t. c is the vector of operating costs for the various data centers assumed to be fixed throughout the simulation period.

Algorithm 1: Provider Total Profit Maximization

Input: r*(t)*, *c*, *Ppd*, *N*, *S(t)*,*D*

Output: *Q(t)*

```
1) Q(t)=0
2) for Each datacenter d < D do
3)     Determine the expected profit,
       profit^d(t)using equation (1).
4)     Determine the required profit,
       profit_min^d(t)using equation (3).
5)     if profit^d(t) ≥ profit_min^d(t) then
6)         Q(t)+= N × S_d
7)     else
8)         if Ppd ≥ c_d then
9)           Determine the number of active
             servers, s_d (t)using equation (4).
10)          Q(t)+= N × s_d (t)
11)        else
12)          Q(t)+= 0
13)        end if
14)    end if
15) end for
```

Users are defined by their preference and budget. These agents are also assumed to be rational agents. We aim for high-throughput computing applications that require large amounts of resources over long periods of time, weeks or months. In this context, the users' objective is to obtain the maximum number of resources within their budget. We take into account the influence of various suppliers on the market. Algorithm 2 describes at time t the behavior of a user with budget *B(t)* faced with a potential price announced by auctioneer Ppd and a price announced by a supplier adopting a static solution *Ps(t)*. The quantities of elementary VMs requested by the user, *Qd(t)* and *Qs(t)*, are given for the dynamic supplier and the static supplier, respectively. At a time t, the dynamic supplier and the static supplier have a maximum computing capacity, denoted by *Qmaxd(t)* and *Qmaxs(t)*, respectively. The user first chooses the supplier with the lowest price; if his demand is not fully covered by this supplier, the user then goes to the second supplier with the remainder of his budget, denoted *Br*.

```
Algorithm 2: User Demand Maximization

Input: B(t), Ps(t),Ppd, Qmaxs(t), Qmaxd(t)

Output: Qs(t), Qd(t)

1) if Ps(t) < Ppd then
2)    Qs(t) = floor(B(t)/Ps(t))
3)    if Qs(t) > Qmaxs(t) then
4)      Br = B(t) - Ps(t) * Qmaxs(t)
5)      Qd(t) = floor(Br/Ppd)
6)    else
7)      Qd(t) = 0
8) else
9)    if Ps(t) == Ppd then
10)     Qs(t) = floor (B(t)/2/Ps(t))
11)     Qd(t) = Qs(t)
12)   else
13)         Qd(t) = floor (B(t)/Ppd)
```

The actions taken by users and suppliers are coordinated by the auctioneer, whose task is to balance the market. The auctioneer periodically runs an algorithm applying the principle of Walras' trial-and-error process. The auctioneer submits a vector of potential prices at each time interval Δt and receives a vector of quantities from the suppliers and users agents representing their preferences. We consider asynchronous communication: agents are not obliged to send their preference information at the same time. When an agent does not submit any preference information, the vector of quantities retained is that of the last auction he or she made. Wellman and Cheng [25] formally show under the assumptions of strict convexity of agents' preferences and gross substitutability between products that convergence is guaranteed even when communication between the auctioneer and the agents is asynchronous. The auctioneer aggregates the quantity vectors of all agents and constructs an approximate excess demand function used to search for the equilibrium price. To reduce the search time, we assume that the dynamic price should not exceed 20 times the static price. We ignore the problem that the dynamic price may exceed the cost of the VM if the user buys the necessary hardware himself. We assume that the search space at time t is the interval $[0, 20 \times P_s(t)]$. The minimum and maximum bounds of this interval correspond to the minimum and maximum potential prices announced by the auctioneer. We divide the search interval into N potential prices that are uniformly distributed in the search interval. Excess demand for a potential price is equal to

aggregate user demand minus aggregate supplier supply for that price. The equilibrium price is the price for which excess demand is close to zero. The auctioneer sets a random initial price within the search interval. He then adjusts the price on the basis of the excess demand function until an equilibrium price is reached. The price is adjusted according to the last set price p, an estimated excess demand $d(p)$ and a step λ initially set to a random positive value:

$$p = p + \lambda \times d(p)$$

As in [26], the determination of step λ resembles a binary search. λ is divided by 2 if the excess demand changes sign and its absolute value increases. This means that the equilibrium price has been exceeded; otherwise, if λ is strictly less than 1/2, then it is multiplied by 2. However, not all the assumptions necessary for the existence and stability of the equilibrium price are verified by the system, particularly the atomicity and the convexity of preferences assumptions. The binary search may not find an equilibrium price or may even recalculate identical prices in a loop. We limit ourselves to a precision of 3 digits after the decimal point. The equilibrium price search function records the prices and excess demands calculated by the binary search. The stopping condition of this function is an equilibrium price found, i.e., $d(p)$ below a certain threshold ε or an oscillation between two identical prices causing a change in the sign of excess demand, i.e., approaching equilibrium without being able to reach it or a maximum number of iterations reached, noted by *maxit*. In the case of oscillation, we take the price that represents a nonzero supply and demand. If the 2 oscillation prices both have nonzero supply and demand, then we retain the price giving positive excess demand to increase competition between dynamic and static suppliers. Algorithm 3 describes the behavior of the auctioneer in the search for the dynamic price.

```
Algorithm 3: search of equilibrium price
Input: Δt, Ps(t), N, maxit, ε
Output: p
   1)  Set init price p to a randomly value in [0, 20*Ps(t)]
   2)  Set λ to a randomly positive number
   3)  Start a thread that receives from agents
       their vector of quantities
   4)  For each interval of time Δt
   5)      update the vector of potential prices PPD(Ps(t), N)
   6)      Submit PPD to all agents
   7)      update the excess demand function d(PPD)
   8)      Determine d(p)
   9)      nb_it = 0
   10)     Repeat
   11)        done = 0
   12)        Repeat
   13)           p=p+λ*d(p)
   14)           Determine d(p)
   15)           nb_it = nb_it+1
   16)           Storage p into P and storage d(p) into D
   17)           If(change sign(D) and increase(D)) then
   18)              λ = λ/2
   19)           else
   20)              done = 1
   21)              If λ<1/2 then
   22)                 λ = λ*2
   23)        Until done = 1
   24)     Until (d(p) ≤ε or oscillation(p, P, D) or nb_it==maxit)
   25)     If nb_it<maxit
   26)        Announce p to all agents
   27)     Else
   28)        Send to agents an error message
```

5. EMPIRICAL STUDY

In this section, we seek to empirically evaluate our dynamic price adjustment algorithm. As we described in section 2, the assumptions of atomicity and convexity of preferences necessary for the existence of the equilibrium price are not verified by our system. We therefore begin by studying the existence of the equilibrium price or the price to be retained (see section 3). We also study the influence of the user budget and the capacity of the dynamic supplier on our price adjustment approach. We compare the two static and dynamic approaches through the study of prices, the quantities sold by suppliers and the profits obtained. For simplicity, we consider one static supplier, one dynamic supplier and one user of VMs. The quantity offered by a supplier may represent the aggregated quantity offered by a set of suppliers. Similarly, the quantity requested by a user from a particular supplier can be the aggregated quantity requested by a set of users from that supplier.

5.1. Study of Convergence

We set the computing capacity for the dynamic provider equal to that of the static provider. It is fixed at 200 elementary VMs per server, and the number of servers is equal to 750, which is of the same order of magnitude as in previous works [11], [7] and [12]. The cost of an elementary VM per unit of time is set to 1. The gain required by suppliers to be able to offer their full capacity is set at 40% of the cost. The static price is equal to (1.4), and the threshold of excess demand ε is set to 1. A price is considered to be an equilibrium price if the associated excess demand is less than 1. The budget is assumed to be equal to 10^5. We analyze the prices calculated by the equilibrium price search function and the estimated supply and demand. Table 1 shows the results obtained over the last 10 iterations.

Table 1. Results of the equilibrium price search function (budget = 10^5)

Iteration	Prices	Demand	Supply	Excess demand
30	1.220	81967.000	82400.000	-433.000
31	1.220	81967.000	82400.000	-433.000
32	1.220	81967.000	82400.000	-433.000
33	1.220	81967.000	82400.000	-433.000
34	1.220	81967.000	82400.000	-433.000
35	1.220	81967.000	82400.000	-433.000
36	1.220	81967.000	82400.000	-433.000
37	1.220	81967.000	82400.000	-433.000
38	1.219	82034.000	82000.000	34.000
39	1.220	81967.000	82400.000	-433.000
40	1.219	82034.000	82000.000	34.000

The price oscillates between (1.22) and (1.219). Excess demand for price (1.219) is positive, while excess demand for price (1.22) is negative. The equilibrium price is not found because the quantity offered by the dynamic supplier ranges from 82000 for the price 1.219 to 82400 for the price 1.22, which corresponds to a minimum price change according to our precision. The dynamic supplier then activates 2 additional servers (400 VMs). Similarly, the quantity requested by the user changes discretely by 67 VMs when the price oscillates between these two values (1.22 and 1.219). The user's demand function is not continuous. Consequently, the assumption of convexity of preferences necessary for the existence of the equilibrium price is not verified. This

is the same on the dynamic supplier side. However, the uniqueness of the equilibrium price, if it exists, is guaranteed because all products, VMs offered by the dynamic supplier and VMs offered by the static supplier, are gross substitutes. The equilibrium price is therefore between 1.219 and 1.22. We retain the price of 1.219 given that the associated excess demand is positive. The dynamic supplier sells all its quantity offered. The next section looks at the behavior of the price adjustment algorithm as a function of the budget.

5.2. Influence of budget on dynamic price

Apart from the budget, we keep the same parameters as those described in the previous section. We vary the user's budget to study its influence on the dynamic price. The price value of 1.4 represents the price asked by the static supplier for an elementary VM. This price covers the gain required by the dynamic supplier, from which he is ready to sell all his VMs. Therefore, a supplier can sell all its computing resources for a budget equal to 210 000 (200x750x1.4). We then vary the budget to generate all possible situations: demand is less than or equal to the capacity of one supplier, demand is greater than the capacity of one supplier without exceeding the capacity of both suppliers, and demand is greater than the capacity of both suppliers. The aim is to compare the dynamic price with the static price, the quantity of VMs sold by each supplier and the profit obtained by each of them. Table 2 lists the retained dynamic prices for the different budgets. It also indicates supply, demand and excess demand on the dynamic provider side for the retained dynamic prices. Table 3 shows the supply, demand and excess demand on the static provider side for the retained static price.

Table 2. Prices, supply and demand on the dynamic provider side for different user budgets.

Budget	Dynamic price	Supply	Demand	Excess demand
100000	1.219	82000	82034	34
200000	1.385	144200	144404	204
210000	1.399	149600	150107	507
300000	1.399	149600	214438	64838
400000	1.399	149600	285918	136318
420000	1.4	150000	150000	0
500000	1.933	150000	150025	25
600000	2.599	150000	150057	57

Table 3. Prices, supply and demand on the static provider side for different user budgets.

Budget	Static price	Supply	Demand	Excess demand
100000	1.4	150000	29	-149971
200000	1.4	150000	201	-149799
210000	1.4	150000	506	-149494
300000	1.4	150000	64791	-85209
400000	1.4	150000	136220	-13780
420000	1.4	150000	150000	0.000
500000	1.4	150000	357142	207142
600000	1.4	150000	428571	278571

For budgets strictly less than 210 000, the dynamic price obtained is lower than the static price. The user chooses the dynamic supplier. He consumes all the quantity offered by the dynamic supplier. Note that for such prices, the dynamic supplier offers part of its total capacity, as long as

the 40% gain and therefore price 1.4 are not reached. A slight excess demand from the dynamic supplier is handled by the static supplier (1 to 2 active servers).

For budgets in the interval [210 000, 420 000[, the dynamic price is constant. This value is slightly lower than the static price (static price of 0.001 = 1.399). The user always chooses the dynamic supplier. Demand continues to exceed the quantity offered by the dynamic supplier. The static supplier takes on the excess demand from the dynamic supplier, which increases by increasing the budget.

For the 420 000 budget, the dynamic price equals the static price (1.4). This is an equilibrium price. The total user demand equals the capacity of the two suppliers.

For budgets strictly greater than 420 000, the dynamic price is higher than the static price. In this case, the user chooses the static supplier. He consumes all the capacity of the static supplier. The dynamic supplier takes on the excess demand of the static supplier, which increases as the budget increases. The dynamic price increases to equalize supply and demand on the dynamic supplier's side. Although the dynamic price exceeds the static price, the dynamic supplier sells all its resources. Demand for the dynamic supplier slightly exceeds its capacity because we retain the oscillation price, which gives positive excess demand.

Figure 1 shows the profits of suppliers for the different budget levels. We can see that the dynamic supplier's profit is higher than the static supplier's profit, regardless of the customer's budget level, except for the budget of 420 000, where the dynamic supplier's profit equals that of the static supplier. For a budget of 100 000, the static supplier's profit is negative (-159.4), whereas the dynamic supplier's profit is equal to (17958). For a budget of 600 000, the profit of the dynamic supplier is approximately 4 times that of the static provider.

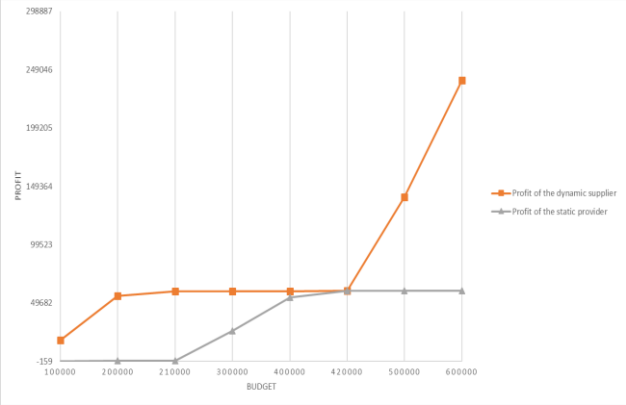

Fig. 1. Dynamic supplier profits and static supplier profits for different user budgets

5.3. Influence of Dynamic Supplier Capacity

The total capacity of the static provider is set at 750 servers. The capacity of the dynamic provider varies from 100 to 1500 servers (double the static provider capacity). Both providers offer the same number of VMs per server. The gain rate is always set at 40% of the operating cost for both suppliers. We take 3 budget examples to generate all the possible situations described in paragraph 4.2: a budget of 1 equal to 100 000, a budget of 2 equal to 300 000 and a budget of 3 equal to 600 000. The aim is to compare, for different capacities of the dynamic supplier, the dynamic price with the static price, the quantities sold by each supplier and the profits obtained. The tables below compare the prices, quantities sold and profits of the suppliers for different

capacities of the dynamic supplier and for different user budget levels. The quantities sold by a supplier are expressed as a percentage of active servers relative to the total number of servers owned by that supplier. Profits are given per activated server.

Table 4. Prices, quantities sold and profits of suppliers as a function of dynamic supplier capacity for a level 1 budget (100 000).

Dynamic supplier capacity (nb. servers)	100	500	750	1000	1500
Dynamic price	1.4	1.306	1.219	1.17	1.119
Static price	1.4	1.4	1.4	1.4	1.4
Percentage of active servers at the dynamic supplier	100	76.4	54.66	42.4	29.73
Percentage of active servers at the static supplier	34.4	0.13	0.13	0.4	0.13
Profit per dynamic provider server	80	46.756	23.944	14.416	7.076
Profit per static provider server	27.198	0.026	-0.212	0.243	-0.022
Total profit of dynamic supplier	8000	23378.398	17958	14416	10614.797
Total profit of static supplier	20399.197	19.8	-159.4	182.6	-16.6

For budget level 1, the dynamic price is less than or equal to the static price, regardless of the capacity of the dynamic supplier. It decreases when the capacity of the dynamic supplier increases. As the capacity of the dynamic supplier decreases, the percentage of active servers at the dynamic supplier increases. For a capacity equal to that of the static provider (750 servers), the dynamic provider activates 54.66% of its capacity (410 servers), whereas the static provider activates 0.13% of its capacity (1 server). The number of active servers at the static provider is high when the capacity of the dynamic provider is very low. By increasing the capacity of the dynamic provider, the percentage of active servers at the static provider is close to 0. The profit per activated server of the dynamic provider is higher than that of the static provider, regardless of the capacity of the dynamic provider. It increases when the capacity of the dynamic provider decreases.

Table 5. Prices, quantities sold and profits of suppliers as a function of dynamic supplier capacity for a level 2 budget (300 000).

Dynamic supplier capacity (nb. servers)	100	500	750	1000	1500
Dynamic price	4.5	1.4	1.399	1.399	1.306
Static price	1.4	1.4	1.4	1.4	1.4
Percentage of active servers at the dynamic supplier	100	100	99.73	99.7	76.46
Percentage of active servers at the static supplier	100	76.26	43.2	10.13	0.26
Profit per dynamic provider server	700	80	79.587	79.560	46.797
Profit per static provider server	80	60.796	34.543	7.783	0.004
Total profit of dynamic supplier	70000	40000	59690.406	79560.625	70196.406
Total profit of static supplier	60000	45597.598	25907.398	5837.799	3.2

Table 6 . Prices, quantities sold and profits of suppliers as a function of dynamic supplier capacity for a level 3 budget (600 000).

Dynamic supplier capacity (nb. servers)	100	500	750	1000	1500
Dynamic price	19.5	3.9	2.600	1.95	1.399
Static price	1.4	1.4	1.4	1.4	1.4
Percentage of active servers at the dynamic supplier	100	100	100	100	99.73
Percentage of active servers at the static supplier	100	100	100	100	86.4
Profit per dynamic provider server	3700	580	319.999	190	79.587
Profit per static provider server	80	80	80	80	69.090
Total profit of dynamic supplier	370000	290000.031	239999.969	190000.016	119380.813
Total profit of static supplier	60000	60000	60000	60000	51817.594

For budget levels 2 and 3, the dynamic price is much higher than the static price, particularly for low dynamic supplier capacity. It decreases as the capacity of the dynamic provider increases. The profit per activated server of the dynamic supplier is always higher than that of the static supplier, regardless of the capacity of the dynamic supplier. It increases when the capacity of the dynamic provider decreases and the user's budget increases. The results show that even for a lower capacity than the static supplier, the dynamic supplier obtains a much greater total profit than the static supplier when the user's budget is high (level 3 budget). This will allow the dynamic supplier to increase its capital and therefore its resources and thus become a heavy actor in the VM market.

6. CONCLUSION

The pricing of VMs in cloud computing systems plays a key role not only in the revenue of providers but also in the load balancing of these systems. Our approach aims to balance supply and demand by pricing VMs. We apply Walras's trial-and-error process, which requires several assumptions to ensure convergence toward equilibrium. These assumptions are not all verified in the real world of cloud computing systems. We propose a realistic model of the behavior of economic agents, both users and suppliers. We find empirically that the change in the sign of excess demand, indicating the existence of equilibrium, occurs between two similar prices. The experimental results show that our dynamic price adjustment approach is more profitable than fixed price pricing is, allowing dynamic suppliers to become important agents in such a market. Our price adjustment algorithm is based on the supply and demand announced by economic agents, users and suppliers. We then plan to integrate a reputation mechanism into our price adjustment system so that the price calculation is not distorted by incorrect announcements.

REFERENCES

[1] Walras L., "Elements of Pure Economics," English translation by William Jaffe, 1954.

[2] A. E. a. M. J. Chourou L., "Global Pricing in Large Scale Computational Markets," In Proceedings of the 7th International Conference on Grid and Pervasive Computing (GPC 2012), pp. 264-278, 2012.

[3] Amazon, "Elastic Compute Cloud (Amazon EC2)," p. Available: https://aws.amazon.com/ec2/, Accessed: May 28, 2021.

[4] M. B.-Y. A. S. a. D. T. .. I. P. o. t. I. 3. I. C. o. C. C. T. a. S. (. Ben-Yehuda O. Agmon, "Deconstructing Amazon EC2 spot instance pricing," In Proceedings of the IEEE 3rd International Conference on Cloud Computing Technology and Science (CloudCom), pp. 1 - 20, 2011.

[5] A. A. a. Y. L. Alzhouri F., "Maximizing Cloud revenue using dynamic pricing of multiple class virtual machines," IEEE Transactions on Cloud Computing, pp. pp. 682-695, vol. 9, 2021.

[6] J. H. a. B. C. Li S., "Resource Pricing and Demand Allocation for Revenue Maximization in IaaS Clouds: A Market-Oriented Approach," IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, pp. 3460 - 3475, 2021.

[7] J. Y. Y. Z. a. M. L. Lu L., "A double auction mechanism to bridge users' task requirements and providers' resources in two-sided Cloud markets," IEEE TRANSACTION ON PARALLEL AND DISTRIBUTED SYSTEMS, pp. pp. 720-733, vol. 29, 2018.

[8] L. H. a. R. S. Shi B., "A deep reinforcement learning-based approach for pricing in the competing auction-based cloud market," Service Oriented Computing and Applications, pp. pp 83-95, Volume 16, Issue 2., 2022.

[9] W. Vickrey, "Counterspeculation, Auctions and Competitive Sealed Tenders," Journal of Finance, pp. 8 - 39, 1961.

[10] K. O. M. D. a. A. C. P. Xu J., "Efficiency-Aware Dynamic Service Pricing Strategy for Geo-Distributed Fog Computing," IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING, Volume: 7, Issue: 4, pp. 814 - 824, 2022.

[11] Z. H. C. W. Z. L. a. F. C. M. L. Zhang X., "Online Auctions in IaaS Clouds: Welfare and Profit Maximization with Server Costs," IEEE/ACM Transactions on Networking, pp. 1034-1047, 2017.

[12] H. L. C. W. Z. L. Z. Z. a. F. L. Zhao J., "Dynamic pricing and profit maximization for the Cloud with geo-distributed data center," In proceedings of the 33rd IEEE Conference on Computer Communications, pp. 118-126, 2014.

[13] E. N. H. M. S.-H. C. H. L. a. I. L. Aazam M., "Cloud Customer's Historical Record Based Resource Pricing," IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 27, NO. 7, pp. 1929 - 1940, 2016.

[14] V. C. a. I. I. Rane D., "A Novel Approach to Dynamic Pricing for Cloud Computing Through Price Band Prediction," ICWE 2021 Workshops, pp. pp. 50-61, 2022.

[15] Y. C. Peng H., "A Novel Dynamic Pricing Approach for Preemptible Cloud Services," IEEE Access, vol. 11, pp. pp. 97807-97825, 2023.

[16] R. L. D. F. a. X. L. Li Y., "Study on Pricing Strategy for Cloud Services based on Contract Theory," 2nd International Conference on Computing, Communication, Perception and Quantum Technology (CCPQT), pp. Pages: 72-77, 2023.

[17] Y. C. a. Z. X. Chen J., "A Novel Two-Stage Game Model for Pricing Cloud/ Fog Computing Resource in Blockchain Systems,," Asia-Pacific Journal of Operational Research, Vol. 40, No. 01, pp. Vol. 40, No. 01, 2240001, 2023.

[18] CloudSigma, p. Available: https://www.Cloudsigma.com, Accessed May 28, 2021.

[19] IONOS, pp. Available: https://www.ionos.com/enterprise-Cloud, Accessed May 28, 2021.

[20] U. D. a. K. G. Hines M. R., "Post-copy live migration of virtual machines,," ACM SIGOPS Operating Systems Review, vol. 43, no. 3, p. p. 14, 2009.

[21] Arrow K.J. and G. Debreu, "Existence of an equilibrium for a competitive economy," in Econometrica vol. 22, pp. 265 - 290, 1954.

[22] J. B. H. O. O. A. W. a. A. M. Taghavi M., "On the Effects of User Ratings on the Profitability of Cloud Services," IEEE International Conference on Web Services (ICWS), pp. 1-8, 2017.

[23] H. B. L. H. Arrow K., "On the stability of competitive equilibrium 2," Econometrica, pp. vol. 27, 82-109, 1959.

[24] Amazon, "EC2 Instance Types," https://aws.amazon.com/fr/ec2/instance-types.

[25] J. C. a. M. Wellman, “The WALRAS algorithm: a convergent distributed implementation of general equilibrium outcomes,” Kluwer Academic publishers, 1998.
[26] F. Y. Sandholm T., “Constructing speculative demand functions in equilibrium markets,” WUCS, pp. 26 - 99, 1999.

AUTHOR

Lilia Chourou Gherir

Training

2019-2024 Faculty of Economic Sciences and Management of Sfax Computer science doctoral student

Subject: Load distribution in large-scale distributed systems, following a market-oriented approach.