FairCloud - Auction-driven Cloud Scheduling*

Artur Fonseca

Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal
artur.fonseca@tecnico.ulisboa.pt

Abstract. With Cloud Computing, access to computational resources has become increasingly facilitated, and applications can offer improved scalability and availability. However, those properties are supported by an energy consumption that is not proportional to the workload, and a pricing model that cannot be tailored to different types of users. One way of improving energy efficiency is by reducing the idle time of resources - resources are active but serve a limited useful business purpose. We present FairCloud, a Cloud-auction scheduler that facilitates the allocation by allowing the adaptation of Virtual Machines (VM) requests (through conversion to other VM types and resource degradation), depending on the user profile. Additionally, this system implements an internal reputation system, to detect providers with low Quality of Service (QoS). FairCloud was implemented using CloudSim and the extension CloudAuctions. It was tested with the Google Cluster Data. We observed that we achieve faster allocations and increased CPU utilisation, translating in more requests satisfied, improved energy efficiency and a higher provider profit. Our reputation mechanism proved to be effective by lowering the ranking of the providers with lower quality.

Keywords: Cloud computing; Energy; Pricing; Auctions; Scheduling; Quality of Service;

1 Introduction

Cloud Computing is defined as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

The following paragraphs describe two Cloud Computing issues that will be discussed in this article: energy consumption and cloud pricing models.

Energy Consumption Datacenters have a high energy consumption, but studies suggest that 30% of the servers are on idle, with less than three percent average

* following the paper accepted to *Cloud and Trusted Computing'17 (International Symposium on Secure Virtual Infrastructures), Springer LNCS, 2017*

daily utilisation [4]. They still consume energy but serve a limited useful business purpose. During periods of low service demand, servers cannot be terminated because they still run background tasks. On the other hand, applications cannot run on fully utilised servers, as even non-significant workload fluctuation will lead to performance degradation.

Cloud Pricing Cloud providers can apply different pricing models. In the **Pay-as-you-go** model, often referred as On-Demand, the user is charged hourly without any long-term commitments or upfront payments. In the **Subscription** model, the total price is reduced but an upfront payment and a long-time commitment (e.g. one year) are required. Another commonly implemented model is **Pay-for-resources**. The price is based on resource consumption (e.g. DiskReadOps, NetworkIn). Recently, providers developed a pricing model for Virtual Machine (VM) without a minimum Quality of Service (QoS) guaranteed, but at a lower fixed price.

1.1 Objectives

The proposed objectives are:

- Analyse previous research works in Cloud-auctions and related topics (e.g. auction theory, scheduling, energy efficiency).
- Formulate a taxonomy and classify the systems according to it;
- Propose, implement and evaluate an algorithm respecting the desired properties (i.e., improve energy efficiency, reduce allocation time, and use a dynamic pricing);

1.2 Document Structure

This work is organised as follows: Section 2 presents and analyses the related work. Section 3 describes FairCloud, including the main entities and their interaction, algorithms, and implementation. Section 4 explains the evaluation methodology and presents the results. Lastly, Section 5 presents some concluding remarks.

2 Related Work

In this section, we describe the **cloud-auctions taxonomy** and illustrate it with examples of research and commercial systems. We constructed the taxonomy by analysing the literature and the differences between systems. Any system, including the one we will develop, can be described based on this taxonomy. The main participants in a cloud-auction are the user and the provider.

2.1 Resource Type

The first aspect to consider when designing a cloud-auction is what will be traded. The majority of systems, [6,7,10,12,13] and Amazon Web Services (AWS) Elastic Compute Cloud (EC2), focus on VM. The system Ginseng [1] trades physical memory. Finally, OptiSpot [2] considers applications as a whole and not its resources individually.

2.2 Negotiation

The negotiation contains two dimensions, participants and bidding scheme.

Participants In **single** auctions, only one participant (either user or provider) submits bids. There are four types of single auctions that are used in non-automated contexts [5]: English, Dutch (or Open descending price), Sealed first-price and Vickrey (or Sealed second-price.). The commercial system AWS EC2 Spot Instances implements a single Sealed auction. In **double** auctions, both participants bid until an agreement is reached. This type of auctions is conducted by an independent participant — the auctioneer. The majority of the research system, namely [6, 10–12], use a double strategy.

Bidding Scheme Bidding scheme defines the process of submitting bids, and the possibility of improving a bid if it was not successful.

In the systems with **unique** bidding [2, 6, 10], each bid is only submitted once. A **multi-round** scheme is implemented in [1,11,12]. A Price and Time-slot Negotiation (PTN) system, proposed in [9], implements a **direct negotiation** scheme. The downside of the multi-round and direct negotiation is the participants need to be active after submitting the initial bids if they wish to improve the bid.

2.3 Time

Regarding the time dimension, auctions can be classified according to the reservation policy and the market type. The **reservation** can be on **spot** or **forward**. In spot auctions, the allocation starts immediately, unlike forward auctions. The algorithms [1,2,6,11,12] are spot auctions. In [3,9], the user can bid for a future spot.

The **market** is denominated **call** if it waits for a set of bids and processes them as a batch. Ginseng [1] is a call auction. AWS EC2 Spot Instances is also a call market but with shorter call periods (e.g. 5 minutes). In **continuous** auctions, also called online, the bid is processed as soon as it arrives. The examples are [2, 6, 7, 9–11].

2.4 Bid Representation Language

A bid representation language is necessary to capture the participant's features. The four main points are resources, quality constraints, time restrictions, and price.

The participant can choose the **resource** type and the amount. An auction is called combinatorial if the participants can bid in combinations of items (i.e., bundles). Recently, the auctioneers started to tailor their algorithms to user's **quality constraints** and the characteristics of the providers. Compatible Online Cloud Auction (COCA), proposed in [10], implicitly considers three types of users: `(I) job-oriented`, `(II) resource-aggressive`, `(III) time-invariant`. Depending on the type of user, the time restriction and price parameters in the bid language are different. The system [2] requires the user application workload and the provider's specification (number of resources and the request process rate).

The bid representation language in [13] requires only the provider energy consumption per VM type. The system proposed in [12] requires the provider QoS metrics (e.g. latency, uptime).

In most systems, particularly in spot auctions, **time restrictions** cannot be specified [1, 6, 7, 11, 12]. However, in [3, 9, 10, 13], it is possible to specify the available time (i.e., when it is possible to start the job), deadline and job duration.

Some systems allow the specification of a bid total **price** [3, 6, 7, 9, 11]. In [10, 13], it is possible to specify a price function based on elapsed time, usually with a constant price if the execution concludes before the deadline and a penalty for each time unit passed after the deadline.

Goals The auction algorithms try to find a compromise to satisfy the user and the provider goals. The most common factors are:

Equilibrium Price The equilibrium price calculation is important, and both participants are interested that their bid is reflected in the calculation. The most straightforward approach is doing the average between user and provider price [6, 11, 12]. In PTN, the participants have total control on the final price as they have to accept it. Ginseng [1] implements an exclusion compensation principle. In other words, it measures what would be the remaining users' allocated amount, if a particular user did not join the auction. Additionally, the user pays a fixed price for the base allocation. OptiSpot [2] constructed a pricing resource model (with a bid value as the input parameter) and populated it using a third-party pricing history. The final price is the bid that optimises the model dimensions.

Truthfulness Truthfulness is the property of algorithms that incentive participants to reveal their true valuation. Some systems could formally prove that they are truthful [7, 10, 12, 13]. Finally, [2] and AWS EC2 Spot are not truthful, but they consider non-truthful bidding should also be allowed as the participant's strategy.

Social Welfare Social welfare is not only related to having the QoS met but also the service quality and the auctioneer utilities and features. In Ginseng [1], there is only a guaranteed base (i.e., if the bid is not successful, a minimum memory is guaranteed). In [12], the providers are rated and order by QoS. Other systems support different categories of users. In PTN, [9] the participants can reject the price and time slot before the allocation starts. Some systems, like [11], implement strategies to converge the user and provider bids through belief functions (i.e., the chance of a bid getting accepted). AWS EC2 and Google Cluster provide a price history for the user to perceive the typical trading prices, but the VM terminates if the spot price increases above the bid. Finally, OptiSpot [2] decides what resources to rent and how to map them to the application modules.

3 FairCloud

In this section, we present FairCloud, a Cloud-Auction system. After analysing the state-of-the-art, we extracted the desired properties and formulated the requirements. Then, we will present the FairCloud bid representation language, data structures and detail the core auctioneer algorithms. Also, in this section, we classify FairCloud using our taxonomy. Finally, we present the main implementation details of FairCloud.

3.1 Desirable Properties

FairCloud has three types of entities: User, Auctioneer and Provider. Their main iterations are to send bids and the allocation results. It is crucial that a Cloud-Auctions system be continuous, combinatorial and truthful. A **continuous** auction is vital to reduce the user allocation time, as the bids are processed as soon as they arrive. Cloud applications are composed of multiple indispensable components, and it should be possible to specify them in one bid (i.e., **combinatorial**). Furthermore, some architectures require that all components be allocated. Therefore, a **roll-back** is necessary if all resources are not allocated. **Truthfulness** is also fundamental. As the auctioneer does not know the participant real valuation, truthfulness is the best method to prevent market manipulation. Additionally, FairCloud should **facilitate the resource allocations**. Therefore, it must consider QoS requirements, in particular: degradations (i.e., only a minimum of guaranteed resources), conversions (i.e., receive the preceding lower capacity machine), heuristics and reputation mechanism.

3.2 Data Structures

FairCloud is supported by four data structures. The "UserBidList" and "ProviderBidList" are managed by the auctioneer and contain all the user and provider bids, respectively. When a new bid is placed, the auctioneer adds it to the corresponding list. Later, when the auction is in progress, the auctioneer sorts, iterates, and updates the bids in the lists. The "AllocationManager", also managed

by the auctioneer, maps, for each user, a list of "AllocationEntry". An "AllocationEntry" contains the following information: provider ID, allocated VM, amount, profile, reputation score and unit price. It is reset before each auction execution and committed at the end when the allocations are sent to the providers. Lastly, the "ReputationMap" maps, for each provider, a queue of 20 values, representing the last 20 scores awarded to that provider (i.e., quality for the allocations).

3.3 Auctioneer Algorithms

FairCloud implements multiple algorithms. The Bid Sorting and Matching algorithm receives and matches the user and the provider bids, and invokes the Update Heuristics algorithm and the Auction Engine Algorithm. After the allocations are decided, the Auction Engine calls the Assignment Protocol. Finally, it implements the reputation operations: Update and Calculation.

3.4 Bid Sorting and Matching

The first step is to initialise the "AllocationManager". Initially, the "UserBidList" is ordered descending by bid density. Then, a join operation is made between "ProviderBidList" and "ReputationMap". The "ProviderBidList" is ordered ascending by $\frac{BidDensity}{Reputation}$. By using this ratio, if two providers have the same price, the one with more reputation is preferred. The **bid density** is a measure of how much a participant bids per unit of allocation.

For each "UserBid", the algorithm selects the current bids and updates the heuristics. Then, the algorithm iterates the "ProviderBidList", focusing on "ProviderBids" where the "UserBid" minimum reputation is assured, and the Auction Engine executes. FairCloud should be able to degrade bids to increase the number of allocations, but only when necessary. To avoid compromising the algorithm execution speed, we will use heuristics. They are a practical method not guaranteed to be optimal or perfect but consume far less computational resources. Lastly, the roll-back is executed. First, it checks if it is necessary and if it is, the allocations regarding that broker on the "AllocationManager" are cleared. In the other case, the allocations are committed and sent to the participants when the auction ends. The reputation is only updated and committed in this step.

3.5 Auction Engine

First, the Auction Engine iterates the requested VM list and checks if the user price is higher than the provider price and if the requested amount is positive. The profile is the maximum between the suggested by the heuristic and the user requested profile. The available amount is calculated by dividing the total capacity and each VM requested capacity, considering the degradation factor. Particularly, the conversion assign amount considers the profile value as 0.5 because each VM computational power is half of the following type.

The assigned amount is the minimum between the requested and available amount. Finally, if the user accepts conversions and the heuristic suggest a conversion, an "AssignVMsConversion" takes place. Otherwise, an "AssignVMs" takes place.

3.6 Assignments

The assignment protocol (i.e., Algorithms "AssignVMsConversion" and "AssignVMs") are composed of two steps: price calculation and reputation award. After the last step, the information regarding the allocation is added in the "AllocationManager".

Price The first step in the assignment is to determine the final price. The final unit price is calculated with Formula 1.

$$Average(UserPrice, ProviderPrice) \cdot degradationFactor \cdot compensation \quad (1)$$

The degradation factor is the real amount of resources allocated. The discount was introduced in [8], and it is a compensation for the users with a performance degradation. Consider the following example: the user profile is `demanding` and the provider (i.e., the assigned profile) offered a `conversion`. The degradation will be 0.5 and the compensation 0.6 (obtained from the matrix).

Reputation The second step is to award the reputation score. It depends on the quality offered: `demanding` - 1, `restricted` - 0.98, `relaxed` - 0.95, and `conversion` - 0.93. We chose these values considering the degradations factors (i.e., 100%, 80%, 60%, 50%).

3.7 Classification using our Taxonomy

To summarise the architecture, we will assess our system according to the taxonomy proposed in Section 2. First, FairCloud trades general-purpose VMs of five predefined types.

It is a double auction, and the bidding scheme is unique. Regarding the time dimension: the auctioneer makes spot reservations and operates in a continuous market.

The bid representation language supports the specification of the requested amount for each type. However, the provider can only specify its total capacity, regarding Million of Instructions per Second (MIPS) and Random Access Memory (RAM). FairCloud supports a variety of quality parameters: degradation profile (i.e., `demanding`, `restricted`, `relaxed`), conversions and minimum reputation desired. The only time restriction is the timestamp representing when the bid is visible to the auctioneer. The user and provider prices are per unit per hour.

The equilibrium price is obtained by multiplying the user and provider average price, the real amount of allocated resources and the compensation. We

can argue that our system is truthful, and it does not consider privacy preservation. Regarding the Service Level Agreement (SLA), FairCloud implements an internal reputation mechanism to order the providers and filter those under a minimum threshold. Furthermore, the reputation does not depend on metrics collected by the provider that could be easily forged. Finally, by implementing an auction, we can reduce the final price paid per VM increasing the utilisation and the energy efficiency. The algorithm always chooses the first available spot.

3.8 Implementation

The FairCloud implementation has three layers: **CloudSim**, **CloudAuctions** (extension) and **bid representation language**. CloudSim is a simulator that is widely used in works related to cloud energy efficiency, work-flows, scalability and pricing policies. This layer offers the Application Programming Interface (API) for VM management and for running the user cloudlets (i.e., user code). It also provides metrics (e.g. utilisation, energy consumption). We used the third-party extension CloudAuctions, proposed and used in [6]. This extension was updated to the latest CloudSim version, fine-tuned and new functionalities were implemented. Finally, a bid representation language was created, to support the user and provider bid API.

4 Evaluation

To evaluate FairCloud, we used a subset of the Google Cluster Data (2011_2). This data provides information regarding Machines, Jobs and Tasks, and Resource Usage. We prepared eight datasets with different goals and characteristics. They differ in the distribution of providers, user degradation profiles, conversions, minimum reputation required and the price.

The evaluation metrics are the following: **average price** per VM, **allocation time**, **execution time**, **CPU utilisation**, **number of allocations** per profile, and **providers rating order**.

We compare our system with one research work, CloudAuctions, and one commercial system, AWS EC2 On-demand.

4.1 Results

In this section, we explore the allocation time and the Central Processing Unit (CPU) Utilisation.

Allocation Time The first remarkable difference is that FairCloud is faster than CloudAuctions in all datasets and faster than AWS in Datasets 1 and 2. This time decrease is due to FairCloud allowing degradations in the requests. After Dataset 3, with heterogeneous prices, AWS is the fastest algorithm allocating requests, but with a cost, requests are rejected.

Dataset 4 introduced one of our features, conversions, and it decreased the allocation time by 25%. It is important to notice that the reputation mechanism does not affect the allocation time (Dataset 4b). The highest allocation time is observed in Dataset 5. This is caused by the users requesting a minimum reputation score — the user bids need to wait for a provider with quality to be available. On the other hand, the allocation time is the lowest in Dataset 6. With a single high capacity provider, the FairCloud heuristic is optimal. Finally, the heterogeneous providers increase the fragmentation, slightly reducing the heuristic effectiveness.

CPU Utilisation We produced a graphic that represents for each time instant, the overall CPU utilisation. Regarding Dataset 2, we observe that CloudAuctions and AWS EC2 On-demand have very close values. After the first hour, FairCloud has slightly less utilisation (53.9% vs 46.7%). This is because the requests are compacted and allocated sooner. On the other hand, in Dataset 3, AWS overall utilisation is the lowest: 29.8%.

4.2 Benefit to participants

After analysing the results, it is important to relate them and see what kind of participants benefit from using FairCloud. Starting with the users:

- Low/medium-end — by accepting to receive partial service, they get a discount and less execution time;
- High-end — they should specify in the bid that their profile is `demanding` to receive full service. They will have maximum priority (i.e., user bids are sorted descending by price) and will, shortly, receive their resources.

On the other hand, ultra-high users can opt for a traditional system, like AWS EC2 On-demand, for a swift allocation time and a predefined price.

Similarly, cloud providers have advantages in bidding in FairCloud. They receive more requests, increase the CPU utilisation and, consequently, their energy efficiency. The reputation mechanism assures that the providers with better quality receive the most requests. The price per VM will be lower, but the overall profit will increase. They still have control over the price. FairCloud pricing formula considers both the user and provider price.

5 Conclusion

We started this work by exploring the energy consumption in cloud computing and ways of optimising it (i.e., increased energy consumption but producing relatively more output). We identified the taxonomy core categories: resource type, negotiation, time, bid representation language, and the goals. FairCloud facilitates resource allocations and considers QoS requirements, in particular, degradations, conversions, heuristics to maximise allocations and their quality,

and a reputation mechanism. The core algorithms are the bid sorting and matching, update heuristics and auction engine.

We concluded that our system benefits low/medium-end users because, by accepting a partial service, they get a discount and less execution time. High-end users can specify that their profile is `demanding` to receive full service. Providers receive more requests, increasing their profit and energy efficiency. They also maintain partial control over the price.

Some suggestions for improvements are: FairCloud could allow geographic preferences and a pricing history. To enrich the pricing model, we could implement subscription and a variable request duration. We could improve the evaluation by adding a third benchmark — AWS EC2 Spot Instances, and measuring the energy consumption, using the CloudSim power consumption models.

References

1. Agmon Ben-Yehuda, O., Posener, E., Ben-Yehuda, M., Schuster, A., Mu'alem, A.: Ginseng: Market-Driven Memory Allocation. ACM SIGPLAN Notices 49(7), 41–52 (2014)
2. Dubois, D.J., Casale, G.: OptiSpot: minimizing application deployment cost using spot cloud resources. Cluster Computing 19(2), 893–909 (2016)
3. Fujiwara, I., Aida, K., Ono, I.: Applying double-sided combinational auctions to resource allocation in cloud computing. Proceedings - 2010 10th Annual International Symposium on Applications and the Internet, SAINT 2010 pp. 7–14 (2010)
4. Kaplan, J., Forrest, W., Kindler, N.: Revolutionizing data center energy efficiency. McKinsey & Company, Tech. Rep (2008)
5. Mcafee, R.P., McMillan, J.: Auctions and Bidding. Journal of Economic Literature 25(2), 699–738 (1987)
6. Samimi, P., Teimouri, Y., Mukhtar, M.: A combinatorial double auction resource allocation model in cloud computing. Information Sciences 357, 201–216 (2016)
7. Shi, W., Zhang, L., Wu, C., Li, Z., Lau, F.C.: An online auction framework for dynamic resource provisioning in cloud computing. ACM SIGMETRICS Performance Evaluation Review 42, 71–83 (2014)
8. Simao, J., Veiga, L.: Partial Utility-driven Scheduling for Flexible SLA and Pricing Arbitration in Clouds. IEEE Transactions on Cloud Computing (99), 1–1 (2014)
9. Son, S., Sim, K.M.: A Price- and-Time-Slot-Negotiation Mechanism for Cloud Service Reservations. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 42(3), 713–728 (2012)
10. Zhang, H., Jiang, H., Li, B., Liu, F., Vasilakos, A.V., Liu, J.: A framework for truthful online auctions in cloud computing with heterogeneous user demands. IEEE Transactions on Computers 65(3), 805–818 (2016)
11. Zhang, Y., Xu, K., Shi, X., Wang, H., Liu, J., Wang, Y.: Continuous double auction for cloud market: Pricing and bidding analysis. In: 2016 IEEE Wireless Communications and Networking Conference. pp. 1–6. IEEE (2016)
12. Zhao, Y., Huang, Z., Liu, W., Peng, J., Zhang, Q.: A combinatorial double auction based resource allocation mechanism with multiple rounds for geo-distributed data centers. 2016 IEEE International Conference on Communications, ICC 2016 (2016)
13. Zhou, R., Li, Z., Wu, C., Huang, Z.: An Efficient Cloud Market Mechanism for Computing Jobs With Soft Deadlines. IEEE/ACM Transactions on Networking 25(2), 793–805 (2017)