An Effective Economic Management of Resources in Cloud Computing

Ghalem Belalem
Dept. of Computer Science, Faculty of Sciences, University of Oran (Es Senia), Oran, Algeria
Email: ghalem1dz@yahoo.fr

Samah Bouamama
Dept. of Computer Science, Faculty of Sciences, University of Oran (Es Senia), Oran, Algeria
Email: samahu@hotmail.com

Larbi Sekhri
Dept. of Computer Science, Faculty of Sciences, University of Oran (Es Senia), Oran, Algeria
Email: Larbi.sekhri@univ-oran.dz

***Abstract*— In Cloud computing, the availability and performance of services are two important aspects to be lifted, because users require a certain level of quality service in terms of timeliness of their duties in a lower cost. Several studies have overcome this problem by the proposed algorithms borrowed from economic models of real world economy to ensure that quality of service, our job is to extend and enrich the simulator CloudSim by auction algorithms inherited from GridSim simulator, but its algorithms do not support the virtualization which is an important part of Cloud Computing, why we introduced several parameters and functions adapted to the environment of cloud computing as well as users to meet their requirements.**

***Index Terms*— Cloud computing, CloudSim, Auction model.**

I. Introduction

Cloud Computing environment is hinting at a future in which we won't compute on local computers, but on centralized facilities operated by third-party compute and storage utilities [9]. This environment, with a great flexibility and ease of use, the availability, of data and of services, becomes one of the biggest problems to treat and to improve [12].

With cloud computing, companies can scale up to massive capacities in an instant without having to invest in new infrastructure, train new personnel, or license new software. Cloud computing is of particular benefit to small and medium-sized businesses who wish to completely outsource their Data center infrastructure, or large companies who wish to get peak load capacity without incurring the higher cost of building larger Datacenters internally.

The CloudSim simulator was proposed to simulate and evaluate the various approaches suggested with the platforms of Cloud Computing [3] [6]. The execution of the simulator makes it possible to make announce a whole of the breakdowns in the treatment of the requests subjected by the users.

GridSim is an event management tool for simulation of heterogeneous Grid resources. It supports the modeling of network entities, users, machines, and the network, including network traffic.

It is able to model and simulate the behavior of Grid applications (execution, scheduling, allocation and monitoring) in a distributed environment consisting of multiple organizations of the grid, but is unable to withstand the demands infrastructure and application levels under the cloud computing paradigm.

In particular, there is very little or no support in the existing grid toolkits for simulation modeling of virtualization to the application to allow the management of resources and applications. Therefore, modeling cloud infrastructure simulation toolkits should provide support to economic entities such as brokers and exchanges cloud for the negotiation of services between customers and suppliers. Among the currently available simulators discussed in this document GridSim that offers support for managing economic resources [2] thing that CloudSim does not have. The work presented in this research is to satisfy the customer in terms of providing a high quality treatment in a minimum cost and in the Clouds Computing offering any improved of algorithms auction of GridSim specific of CloudSim to simulate an economic environment in the cloud computing, with the goal of providing virtualized resources to treat Cloudlet from users.

We offer a hybridization of the two main aspects of simulators GridSim and CloudSim, which are the implementation of economic models of bidding GridSim and virtualization of computing resources CloudSim.

The rest of the paper is structured as follows: in Section 2, we define cloud computing environments as an important scientific trend. Section 3 is dedicated to the simulator GridSim; define the simulator and its uses in the Grid. We present in Section 4, the simulator CloudSim, The following section identifies changes to CloudSim to ensure proper management of resources in

doi:10.4304/jcp.6.3.404-411

the simulator CloudSim. The different experiments are presented in Section 6. We end our paper with a summary and some extension work that we will consider doing so.

II. CLOUD COMPUTING

Cloud computing can be defined as "A type of parallel and distributed system consisting of a collection of interconnected computers and they are virtualized and dynamically generated and submitted as one or more computing resources based on service level agreement established through negotiation between the provider and consumers" [4]. Some examples of new infrastructure are Microsoft Azure Cloud Computing [8], Amazon EC2, Google and Aneka [15].

Computing power in cloud computing environments is provided by a collection of data centers, which are typically installed with hundreds of thousands of servers [5]. The layered architecture of a typical cloud-based data centers is shown in Figure 1. In the lower layers there are huge hardware resources (storage and application servers) which feed data centers. The servers are managed transparently by the level of virtualization [14], services and toolkits that enable the sharing of their capacity through virtual instances of servers. These virtual instances are isolated from each other, this helps to achieve fault tolerance and isolation of the security environment.

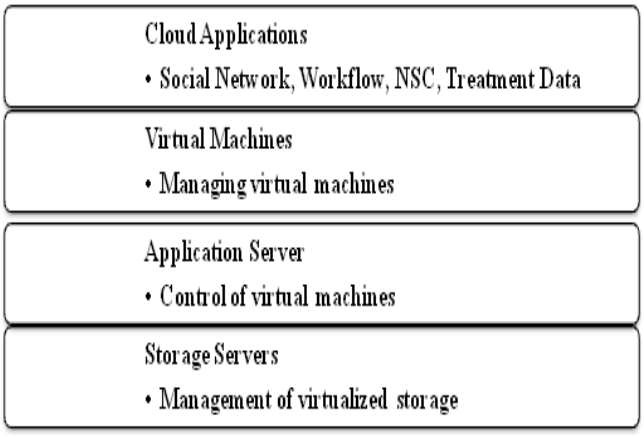

Figure 1. Typical DataCenter.

Cloud applications such as social networking, game portals, applications, economic and scientific workflows running in the highest layer of the architecture. The patterns of actual use of many real applications vary with time, mostly unpredictable ways. These applications have different requirements of Quality of Service (QoS) depending on the criticality of the time and modes of user interaction (online / offline).

III. GRIDSIM

GridSim [5] toolkit was developed by Buyya et al to address the problem of near impossibility of performance evaluation of real large scaled distributed environments (typically Grid systems but also P2P networks) in a repeatable and controlled manner. The GridSim toolkit is a Java based simulation toolkit that supports modelling and simulation of heterogeneous Grid resources and users spread across multiple organizations with their own policies. It supports multiple application models and provides primitives for creation of application tasks, mapping of tasks to resources and managing such tasks and resources.

IV. CLOUDSIM

CloudSim [3] is a framework developed by the GRIDS laboratory of University of Melbourne which enables seamless modelling, simulation and experimenting on designing Cloud computing infrastructures. CloudSim is a self-contained platform which can be used to model data centers, service brokers, scheduling and allocation policies of a large scaled Cloud platform. It provides a virtualization engine with extensive features for modelling the creation and life cycle management of virtual engines in a data center and provides flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services.

CloudSim framework is built on top of GridSim framework also developed by the GRIDS laboratory.

Figure 2 shows the conception of the CloudSim toolkit [6]. At the lowest layer, we find the SimJava [11] that implements the core functionalities required for higher-level simulation such as queuing and processing of events, creation of system components (services, host, data center, broker, virtual machines), communication between components, and management of the simulation clock. In the next layer follows the GridSim toolkit that support high level components for modelling multiple Grid components Such as networks, resources, and information services.

The CloudSim is implemented as layer by extending the core functionality of the GridSim layer. CloudSim provides support for modelling and simulation of virtualized Datacenters environments such as management interfaces for VMs, memory, storage, and bandwidth. CloudSim layer manages the creation and execution of core entities (VMs, hosts, Datacenters, application) during the simulation period. This layer handle the provisioning of hosts to VMs based on user requests, managing application execution, and dynamic monitoring. The final layer in the simulation stack is the User Code that exposes configuration functionality for hosts (number of machines, their specification and so on), applications (number of tasks and their requirements), VMs, number of users and their application types, and broker scheduling policies. A Cloud application developer can write an application configurations and Cloud scenarios at this layer to perform a cloud computing scenario simulations.

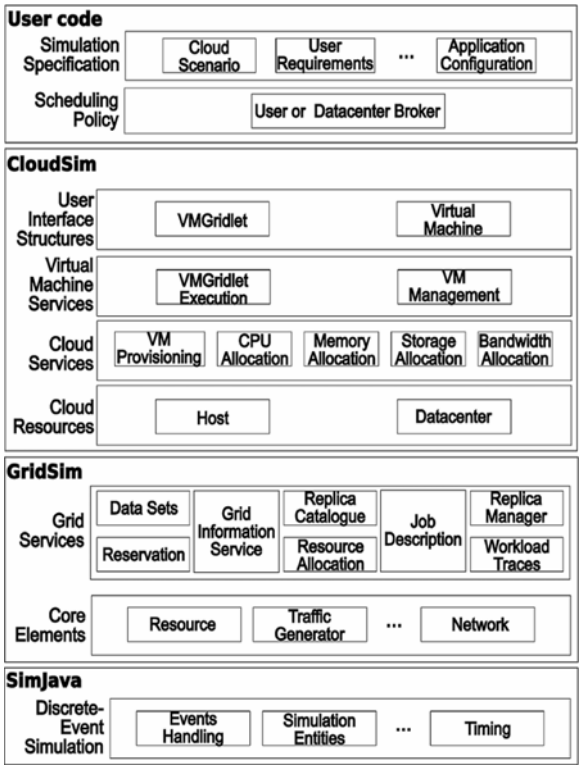

Figure 2. CloudSim architecture [9].

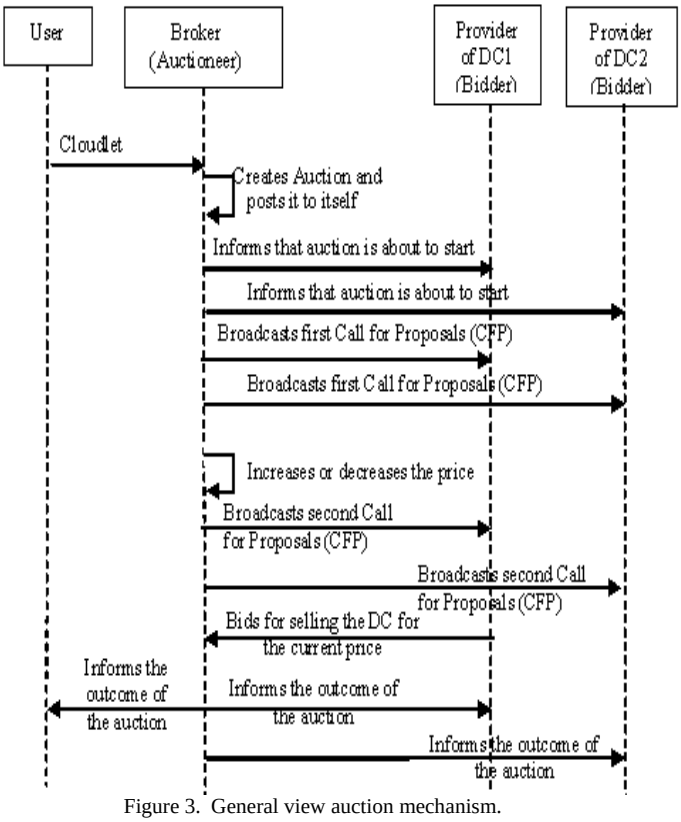

Figure 3. General view auction mechanism.

V. IMPROVED THE CLOUDSIM SIMULATOR

To improve the quality of treatment service of users cloudlets, we made a comparative study between the simulator CloudSim who inherits own bidding algorithms of GridSim simulator, as well as those we have implemented and integrated into the simulator GridSim specific of CloudSim, after decompiled CloudSim to make the changes mentioned above on the simulator GridSim and then recompiled with the toolkit GridSim amended. GridSim implements the four types of auction are: First Price Auction sealed bid (FPSB), English auction, Dutch auction and Continuous Double Auction (CDA), these types of betting revolves around one principle which is the increase or the starting price set by the supplier or the reduction of a very high price to reach the reserve price set by supplier to win the auction with a price that satisfies both the customer and supplier. The diagram below illustrates the general principle of auctions in the Clouds Computing.

Initially, the user submits Cloudlets to his broker. In the Cloud, a broker is responsible for submitting and monitoring Cloudlets on the user's behalf. The broker creates an auction and sets additional parameters of the auction such as Cloudlet length, the quantity of auction rounds, the reserve price and the policy to be used (e.g. English or Dutch auction policy). As the broker also plays the role of auctioneer, it posts the auction to itself; otherwise, the auction would be post to an external auctioneer. The auctioneer informs the bidders that an auction is about to start. Then, the auctioneer creates a call for proposals (CFP), sets its initial price, and broadcasts the CFP to all the bidders. Providers formulate bids for selling a service to the user to execute the Cloudlet. The first time that bidders evaluate the CFP, they decide not to bid because the price offered is below what they are willing to charge for the Data center. This makes the auctioneer to increase or decrease the price and send a new CFP with this new price. Meanwhile, the auctioneer keeps updating the information about the auction. In the second round, a bidder decides to bid. The auctioneer clears the auction according to the policies specified before. Once the auction clears, it informs the outcome to the user and the bidders.

A. First Price Sealed Bid Auction

Bidders are not aware of each other's offers [1]. In addition, it is a single round auction. When bidders receive a call for proposals, they can verify the minimum price and either decide to bid or not to bid for the good. The auctioneer waits a given time for the bids and then allocates the good to the bidder who has valued the good the most. Part of this class is presented in Algorithm 1.

B. English Auction (EA)

This economic model [7] is an ascending auction in which the auctioneer tries to find the price of a good by proposing an initial price below the supposed market value and slowly raising the price until no bidder is interested in paying the current price for the good. Then the best past bid is chosen. Part of this algorithm is presented in Algorithm 2.

Algorithm 1: FirstPriceSealedBidAuction

```
Var double Min_Price, Current_Price,
Reserve_Price, Final_Price;
/*ReservePrice is the maximum price at which a
seller is willing to sell a Data center*/
/*CurrentPrice is the price of asker*/
MessageCallForBids Msg ;
MessageBid best
int  AuctionID, currentRound;

/* This method is called when a round is
started*/
void onStart(int round)
          if (round == 1) then
          Min_Price:=Reserve_Price;
   Current_Price:=Reserve_Price;
   end if
/* Creates a call for proposal that is broadcast to
all bidders*/
     Msg = new
MessageCallForBids(AuctionID,AuctionProtocol,Mi
n_Price,currentRound);
     broadcastMessage(Msg);
end;

/* This method is invoked when a round
finishes*/
void onClose(int round)
   best = getFirstBid();
         if (best != null) then
         double price = best.getPrice();
           if (price >=Reserve_Price)
           then
             FinalPrice:=price;
        Winner_Id:=best.getBidder()
       else
      Final_Price:=Current_Price;
       end if
    else       Final_Price:=Current_Price;
    end if
end;
```

C. Dutch Auction (DA)

This economic principle [7] is a descending auction and differs from the English auction in the sense that the auctioneer starts by issuing a call for proposals with a price much higher than the expected market value. The auctioneer then gradually decreases the price until some bidder shows interest in taking the good for the price announced. Part of this algorithm is presented in Algorithm 3.

D. Continuous Double Auction (CDA)

The Continuous double auction [10] works with a system of bids and asks. The price is found by matching asks and bids. The auctioneer accepts asks and bids and tries to match them. The auctioneer informs the price to the bidder and the seller when a match is made. Part of this algorithm is presented in Algorithm 4.

Algorithm 2: English Auction

```
Var double Min_Price, Current_Price,
Reserve_Price, Final_Price;
/*ReservePrice is the maximum price at which
a seller is willing to sell a Data center*/
/*CurrentPrice is the price of asker*/
MessageCallForBids Msg ;
MessageBid  bestBid
int    NumberOfRounds ;
boolean  shouldIncrease

/* This method is called when a round is
started */
void onStart(int round)
    if(round == 1)then
         initialBidders =getBidders();
         Current_Price :=Min_Price;
   end if
       broadcastMessage(msg);
end;
/* This method is invoked when a round
finishes */
void onClose(int round)
boolean stop = false;
     if (!shouldIncrease|| round
==NumberOfRounds)
        stop = true
   else  shouldIncrease = false;
        double increase=(Max_Price–
Min_Price)/(NumberOfRounds - 1);
        Current_Price= Current_Price +
increase;
    end if
  if (stop) then
        if (bestBid != null) then
         double price = bestBid.getPrice();
                  if (price >= Reserve_Price)
       then
                  Final_Price:=price;
                  Winner:=bestBid.getBidder()
              else Final_Price:=Current_Price;
              end if
         else Final_Price:=Current_Price;
         end if
   end if
end;
```

The principle of continuous double auction is shown in the diagram in Fig. 4.

Among the improvements in our work is that we have introduced a specific parameter to the user who is the budget in order to manage its budget by the number of Cloudlet and their importance in terms of size.

The user will bid with the broker representing the Data Center on an auction mentioned above, when the user tries to treat all its Cloudlet in minimum time and with less cost. CloudSim GridSim and assume that the size of the Cloudlet before treatment is fixed this means

Algorithm 3: Dutch Auction

```
Var double Min_Price, Current_Price,
Reserve_Price, Final_Price, Current_Price;
/*ReservePrice is the minimum price at which a
seller is willing to sell a Data center*/
/*CurrentPrice is the price of asker*/
MessageCallForBids Msg ;
int    NumberOfRounds ;
boolean  shouldIncrease

/* This method is called when a round is started
*/
public void onStart(int round)
          if (round == 1) then
        Current_Price:=MaxPrice;
    end if
        broadcastMessage(msg);
end;

/* This method is invoked when a round
finishes */
public void onClose(int round) {
          if (round >=NumberOfRounds) then
           if (bestBid == null) then
           setFinalPrice(getCurrentPrice())
                  else double decrease=
                  Max_Price/(NumberOfRounds
                  - 1);
           Current_Price:=Current_Price -
        decrease;      end if;
   end if;
end;
```

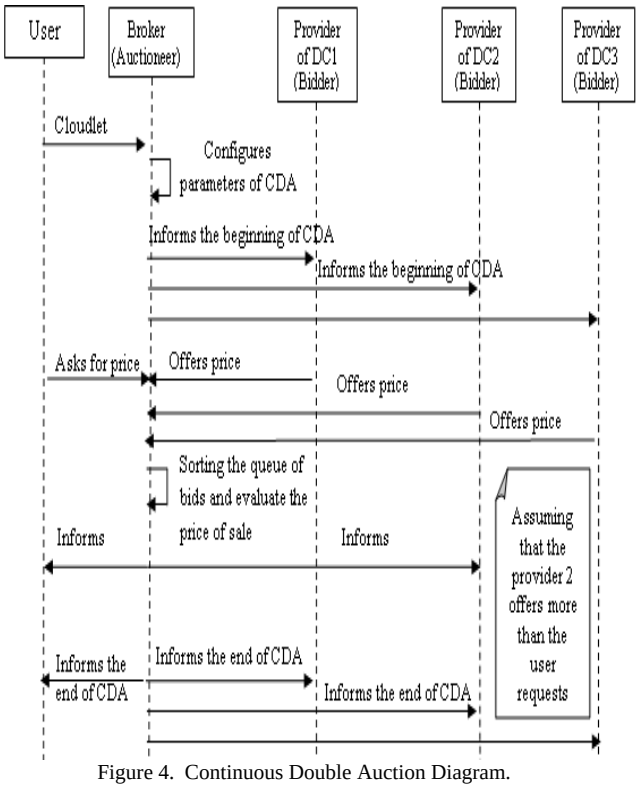

Figure 4. Continuous Double Auction Diagram.

Algorithm 4: Continuous Double Auction

```
Var LinkedList asks,bids; /*the list of offers
and requests*/
Comparator compAsks,compBids;
/* Called when a bid is received.*/

public void onReceiveBid(MessageBid bid)
Collections.sort(asks,compAsks);
          if (asks.size() > 0) then
        MessageAsk ask =
(MessageBid)asks.getFirst();
       double priceAsk = ask.getPrice();
       double priceBid = bid.getPrice();
             if(priceBid >= priceAsk){
             double finalPrice = (priceAsk +
         priceBid) / 2;
             match(ask, bid, finalPrice);
             asks.remove(ask)
             else    bids.add(bid);
             end if
      else   bids.add(bid) ;
      end if;
end;

/* Called when a ask is sent by a provider. */
public void onReceiveAsk(MessageAsk ask)
Collections.sort(bids,compBids);
           if(bids.size() > 0) then
        MessageBid bid =
(MessageBid)bids.getFirst();
       double priceAsk = ask.getPrice();
       double priceBid = bid.getPrice();
             if(priceBid >= priceAsk)then
                double finalPrice = (priceAsk +
priceBid) / 2;
             match(ask,bid,finalPrice);
             bids.remove(bid)
            else  asks.add(ask);
            end if
      elseasks.add(ask);
    end if;
        end;
```

that all have the same budget Cloudlet view that it depends on their size using the following formula:

$$UnitaryPriceKB = Budget/TotalSizeOfAllCloudlets \quad (1)$$

From the Formula (1), we can deduce the estimated cost to cloudlet given by the formula below:

$$CostOfCloudlet = UnitaryPriceKB*SizeOfCloudlet \quad (2)$$

In specific auction algorithms of GridSim, the price offered by the Data Center is random and that generated by the user depends on this price, does not guarantee the true estimate of data center cost and the treatment of a cloudlet is why we have proposed a function that generates the offer price. This function estimates the cost of executing the Cloudlet and the cost of their transfers

before and after the execution assuming that its size before and after treatment is the same (see Formula 3).

GenBid = (CostPerCPU*TimeCPU)+2*(SizeOfCloudlet* CostPerBandWidth) (3)

Each algorithm has 3 major auction prices for auction process, these prices are: max price, min price and reserve price.

The reserve price is the same in all algorithms of auction and it is equal to the Formula 2, the maximum price in the Dutch and English auction is equal to Formula 4.

Maximum price = Budget/NB_Cloudlet (4)

For the minimum price of Dutch auction is zero, while that of the English auction is equal to a price that exceeds the offer price of formula (3) to respect the procedure of the English auction described above.

VI. SIMULATIONS

We conducted four sets of simulation to prove that the auction algorithms provide good quality service in terms of cost and duration of minimum bid. The aim of the first series is to see the impact of size variation of the cloudlet in its cost, therefore we have fixed the number of datacenter to 5, the number of host to 20 each has 2 processors which costs $3 (Price of treatment) with 2MB of RAM which costs $0.5 (price list), 128 MB / sec which costs $0.01 (cost of the bandwidth), 500GB of storage which costs $0.1 (price of storage), the number of virtual machine is 2, the number of user is 1 with a budget of $300 and 20 Cloudlet varied their size between 100 and 1000 KB in increments of 100, the number of rounds for the Dutch and English auction is 10 duration 120 ms, Figure 5 shows the resulting graph.

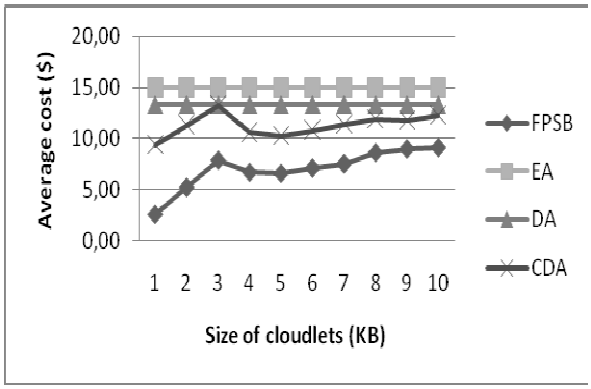

Figure 5. Average cost of Cloudlets according to their size

We can note from the Figure 5 that FPSB lower the cost of Cloudlet more than other auction models despite the increased size of Cloudlet because it is the user who set the starting price this is dependent on its budget, unlike the CDA, where the price proposed is carried by the user and the datacenter divided by two, we also note that despite the size variation of Cloudlet, the average cost DA and EA is stable because in this example the reserve price coincides with the ask price, with a net increase of EA versus DA.

The aim of the second set of simulation is to see the impact of varying number of data centers on the average cost of cloudlet, we set the size to 300KB Cloudlet 50 each, and varied the number of data center from 1 to 10 by no one, the figure below shows (see Figure 6) that the average cost of four model bid.

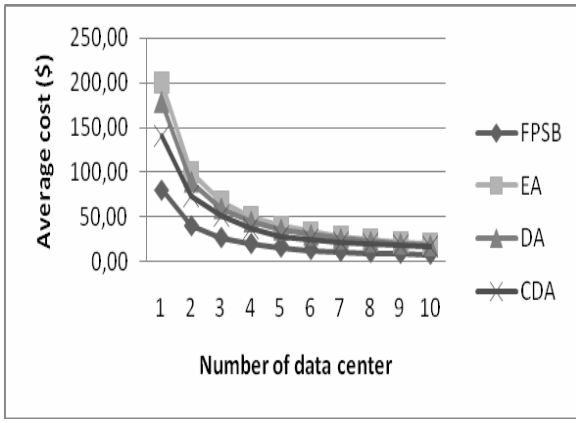

Figure 6. Average cost of Cloudlets according the number of datacenter

The third set of simulation has to stumble to see the impact of number de cloudlets on the average time of the auction, we found the following result (see Figure 7), all bidding models are slow in terms of increasing the number of Cloudlets, but the auction CDA is much faster than other auction for it is not a continuous enhancements such as DA or EA, FPSB in second place because the price generated by the user is compared with the reserve price and the supplier awarded directly, but EA much time compared to DA because it attempts to maximize the greatest possible gain of the supplier without it has a price limit.

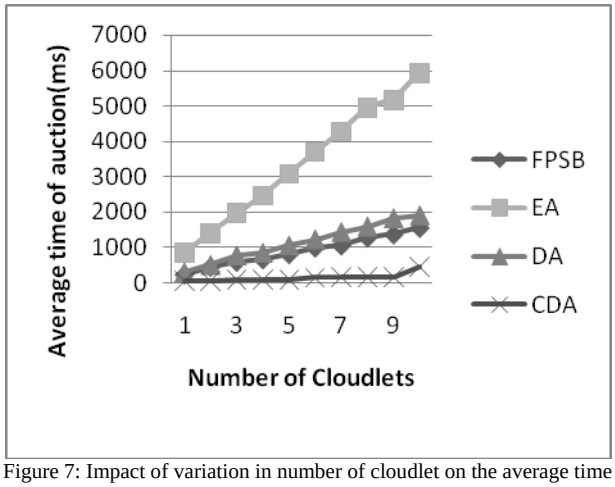

Figure 7: Impact of variation in number of cloudlet on the average time of the auction

Fourth set of simulation has to stumble to see the impact of varying the number of users on the average time of auction, Figure 8 shows that increasing the number of users to a negative impact on the average time

of the auction, but CDA is the fastest of all the models auction saw its operating principle, we also note the EA and DA will coincide in the point (6 ; 150000) that is to say, 6 users both bids are the same time, but beyond 6 EA takes less time compared to the DA this is due to that the competitor to raise prices as quickly as possible to break the price of other competing offers.

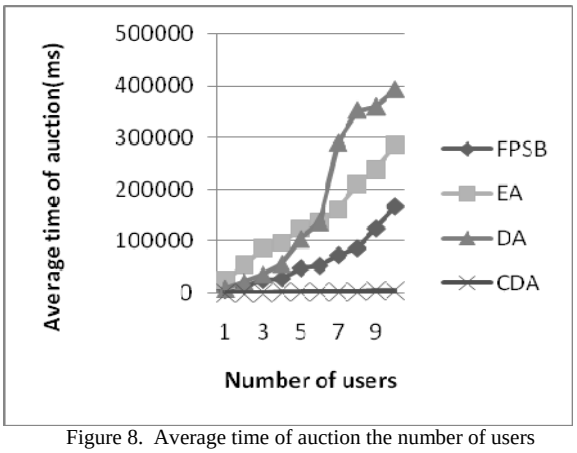

Figure 8. Average time of auction the number of users

VII. CONCLUSION

The recent efforts to design and develop Cloud technologies focus on defining novel methods, policies and mechanisms for efficiently managing Cloud infrastructures.

The main objective of our work is to satisfy customers Clouds, while providing economic functions that reduce the cost of processing cloudlets and improved auction algorithms implemented in GridSim to reduce the time auction and to assure a rapid and effective acquisition of computing resources. The experimental results obtained are very satisfactory and meet our expectations. For a continuation of our work, several perspectives can be envisaged:

- Proposing a method or approach that will satisfy both the customer and supplier resource.
- Extend this work by a process of discovery and allocation of appropriate resources in the cloud computing [13].
- Find a solution for customers who want a high quality of treatment but who do not have enough budgets, by the possibility of credit.
- Propose an implementation of auction club using a multi-agent system.

ACKNOWLEDGMENT

This work is supported by Computer Science Laboratory of Oran, Algeria (LIO).

REFERENCES

[1] M. D. D. Assuncao, and R Buyya, "An Evaluation of Communication Demand of Auction Protocols in Grid Environments", *In Proceedings of the 3rd International Workshop on Grids Economics & Business (GECON 2006)*, pp. 24-33, Singapore, May 2006.

[2] G. Belalem, B. Yagoubi and S. Bouamama, "An Approach Based on Market Economy for Consistency Management in Data Grids with OptorSim Simulator", *International Journal of Information Technology and Web Engineering (IJITWE)*, vol. 3, no. 3, pp.1-16, 2008.

[3] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities", Keynote Paper. In *Proceedings of the 7th High Performance Computing and Simulation* (HPCS 2009) Conference, Leipzig, Germany 2009.

[4] R. Buyya, C. S. Yeo, and S. Venugopal, "Marketoriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities". *In Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications*, 2008.

[5] R. Buyya, and M. Murshed, "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing", *The Journal of Concurrency and Computation: Practice and Experience (CCPE)*, vol. 14, no. 13-15, pp. 1175-1220, Wiley Press, Nov.-Dec., 2002.

[6] R. N. Calheiros, R. Ranjan, C. A. F. De Rose, and R. Buyya, "CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services", *Technical Report*, GRIDS-TR-2009-1, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, 2009.

[7] R. Jr. Cassady, "Auctions and Auctioneering". *University of California Press*, Berkley and Los Angeles, California, 1967.

[8] D. Chappell, "Introducing the Azure services platform". *White paper*, Oct. 2008.

[9] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-degree compared", *in Grid Computing Environments Workshop*, GCE'08, Austin, TX, pp.1–10, 12-16 Nov 2008.

[10] D. Friedman, and J. Rust, "The double auction market: institutions, theories, and evidence". Addison-Wesley, 1993.

[11] F. Howell, and R. Mcnab, "SimJava: A discrete event simulation library for java". In *Proceedings of the first International Conference on Web-Based Modeling and Simulation*, 1998.

[12] T. Rings, G. Caryer, J. R. Gallop, J. Grabowski, T. Kovacikova, S. Schulz, and I. Stokes-Rees, "Grid and Cloud Computing: Opportunities for Integration with the Next Generation Network", *Journal of Grid Computing*, vol. 7, no. 3, pp.375-393, 2009.

[13] A. Sharma and S. Bawa, "Comparative Analysis of Resource Discovery Approaches in Grid Computing". *Journal of Computer (JCP)*, vol. 3, no. 5, pp. 60-64, 2008.

[14] J. E. Smith, and R.Nair, "Virtual Machines: Versatile platforms for systems and processes". *Morgan Kauffmann*, 2005.

[15] C. Xingchen; K. Nadiminti, J;Chao; S. Venugopal, and R. Buyya, "Aneka: Next-generation enterprise grid platform for e-science and e-business applications". In *Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing*, pp. 151-159, Bangalore, 10-13 Dec. 2007.

Ghalem Belalem graduated from University of Oran, Algeria, where he received PhD degree in computer science in 2007. His current research interests are distributed systems, grid and cloud computing, placement of replicas and consistency management in large scale systems and mobile environment.

Samah Bouamama a master's candidate in the Department of Computer Science, Faculty of Sciences, University Of Oran, Algeria. Her research interests are Grid and Cloud Computing, replication strategies and consistency management.

Larbi Sekhri received his PhD in Computer Science from Oran University in 2006 (Algeria). His current research area include formal modeling in wireless ad-hoc and sensor networks, systems modeling using Petri nets, automata theory, diagnosability and monitoring of automated production systems and reasoning in artificial intelligence.