An Efficient Economic Model User-Oriented for Cloud Computing

Ghalem Belalem
Dept. of Computer Science
Faculty of Sciences
University of Oran - Algeria
BP. 1524, EL M'Naouer,
Oran, Algeria (31000)

Samah Bouamama
Dept. of Computer Science
Faculty of Sciences
University of Oran - Algeria
BP. 1524, EL M'Naouer,
Oran, Algeria (31000)

Larbi Sekhri
Dept. of Computer Science
Faculty of Sciences
University of Oran - Algeria
BP. 1524, EL M'Naouer,
Oran, Algeria (31000)

ABSTRACT
The Cloud computing offers the availability and performance of services those users require to resolve their requests in reduced time and 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. The work presented in this paper allows in the first time to satisfy customers in terms of treatment quality of cloudlets cheaply in the cloud, while offering improved algorithms auction of simulator GridSim specific of CloudSim to simulate economic environment in the cloud, in order to provide virtualized resources. In the second time, we propose a hybrid business model that behaves well relative to the different models studied.

General Terms
Cloud computing, Grid computing, Distributed systems, Resource management.

Keywords
Cloud computing, CloudSim, Auction model, Cloudlet, Datacenter.

1. INTRODUCTION
Today, applications are becoming more and more resource-intensive, either in memory or computing power. These resources are limited on a machine, the need to distribute applications over a network of computers, local or remote, is obvious. Grid computing is mainly focused on the provision of a significant amount of shared resources for intensive computing (research or physical applications). The protocols used are tailored to the specific topology of the network (low latency, high throughput). Cloud Computing is based instead on the provision of many services and data to users [1], without their having to manage the complex infrastructure required. Because of its support web services, it uses Internet protocols and is therefore subject to the constraints of the latter. The cloud computing environment suggests a future where we do not calculate on local computers, but on equipment centralized computing and storage operated by third parties. This environment has great flexibility and ease of use, availability of data and services became one of the biggest problems to address and improve.

With cloud computing, companies can raise large capacity in an instant without having to invest in new infrastructure, new staff or new software. This model of development was born after the giants of the Internet have created their own infrastructure to meet their own needs; they are now able to ensure their development, but also to share resources at competitive prices. Cloud infrastructure modeling and 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 who have an interest in our work, the simulator GridSim which provides management support and economic resources CloudSim simulator [8] [10] to model the cloud. Our job is to propose a hybridization of simulators CloudSim and GridSim aspects, which are implementing economic models of GridSim and virtualization of computing resources of CloudSim and then consider four auction models and their improvement. Finally, propose an economic model that brings together all beneficial criteria on the client side of the auction models.

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 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.

2. 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*" [9]. Some examples of new infrastructure are Microsoft Azure Cloud Computing [5], Amazon EC2, Google and Aneka [6].

Computing power in cloud computing environments is provided by a collection of data centers, which are typically installed with hundreds of thousands of servers [3]. The layered architecture of a typical cloud-based data centers is shown in "Fig. 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 [7], 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.

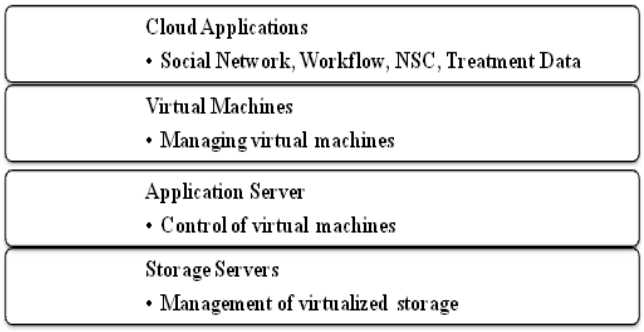

Figure 1. Typical Data Center.

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).

3. GRIDSIM

GridSim [3] 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 modeling 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.

4. CLOUDSIM

CloudSim [8] is a framework developed by the GRIDS laboratory of University of Melbourne which enables seamless modeling, 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 modeling 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 [10]. 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 modeling 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 modeling 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.

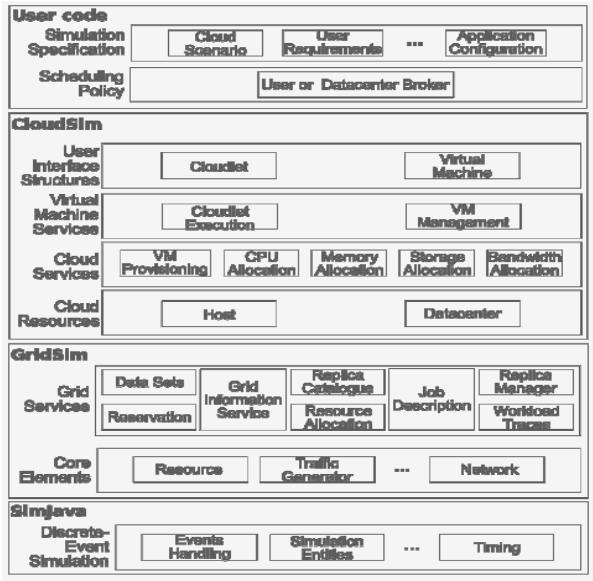

Figure 2. CloudSim Architecture.

5. 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.

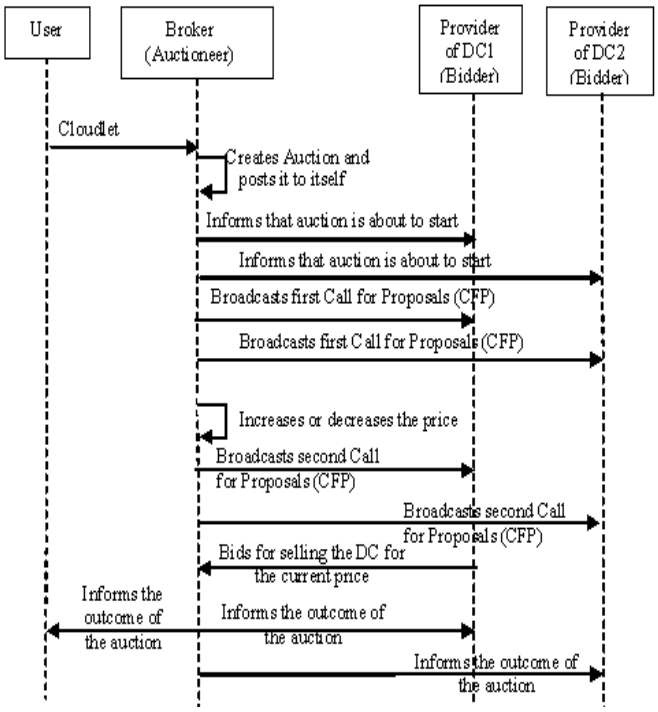

Figure 3. General view auction mechanism.

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.

5.1 First Price Sealed Bid Auction

Bidders are not aware of each other's offers [4]. 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 below:

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,Min_Price,cu
rrentRound);
    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;
```

5.2 English Auction (EA)

This auction [12] 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 below:

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;
```

5.3 Dutch auction (DA)

The DA [12] 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 below:

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;
```

5.4 Continuous double auction (CDA)

The Continuous double auction [13] 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 below:

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
     else  asks.add(ask);
   end if;
         end;
```

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

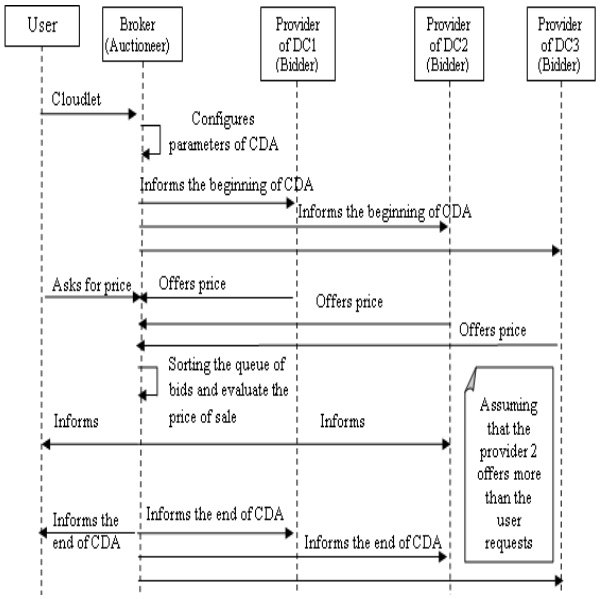

Figure 4. Continuous Double Auction Diagram.

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

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

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

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

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)).

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

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).

$$Budget/NB\_Cloudlet \quad (4)$$

For the min price of Dutch auction is 0, 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.

5.5 Reverse continuous double auction (RCDA)

Apart from changes in patterns of GridSim auction, we proposed a new model called Reverse Auction Continuous Double Auction (RCDA) combining: the auction CDA, DA and EA, for that we have built into the auction CDA the two auctions, DA and EA. In RCDA, the broker maintains the scheduling of client requests in descending order, as the Dutch auction model and offers from suppliers in increasing order, while respecting the principle of the English auction model. Upon receipt of an application compared to the broker's first offer of the list. If the demand price is less than or equal to the value of the offer, he informs that the Data Center and the user can share resources at the following prices:

$$(\text{AskPrice} + \text{OfferPrice}) / 2 \quad (5)$$

Otherwise, the broker adds the application in the list. Figure illustrates the main activities of the Broker in this case the proposed model RCDA.

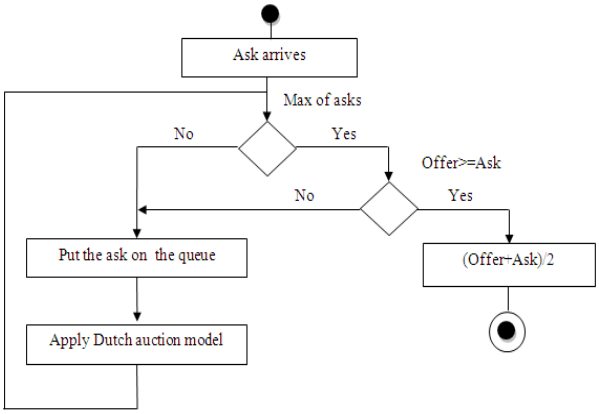

Figure 5. RCDA: receiving an ask by an offer.

In the case where the broker receives an offer, he compares it to the first request for the list. If the demand price is less than or equal to the value of the offer, it informs the user and the supplier can trade at the same price from the equation (5). Otherwise, the broker said the offer in the corresponding list; in this case the main activities of the Broker are shown in Figure 6.

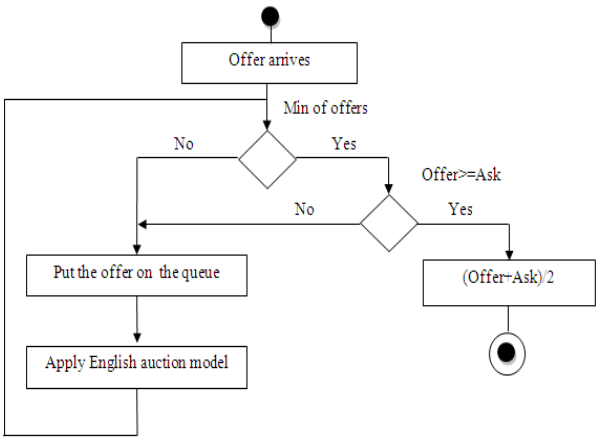

Figure 6. RCDA: receiving an offer by an ask.

6. SIMULATIONS

We conducted two 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 change on the number of cloudlets in its cost, therefore we have fixed the number of data center 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 $ 1000 and we varied the number of Cloudlets between 5 and 50 by 5 steps, the size of each one is100KB, number of rounds in the auction from Holland and English is 10 duration 120 ms,

Figure 7 shows the resulting graph, we note that the FPSB auction reduces the cost average processing Cloudlet more than other auction models despite the increase in the number of Cloudlet because the reserve price depends on how many Cloudlet (see the 2nd Formula) and that according to the function 1 / x where x is Cloudlet numbers, same for the auction CDA except that it decreases the average cost of Cloudlet from $ 100 up to $ 10, the proposed auction is almost confused with the auction CDA, except that it decreases continuously the average cost which is why the model improved CDA provides good QoS from raw CDA, same thing for auction DA and EA that they do not reduce the average cost of Cloudlet a great value compared to the All auction templates (begins at $ 180 and $ 200), this is due to its principle that is the price increase until reaching the reserve price.

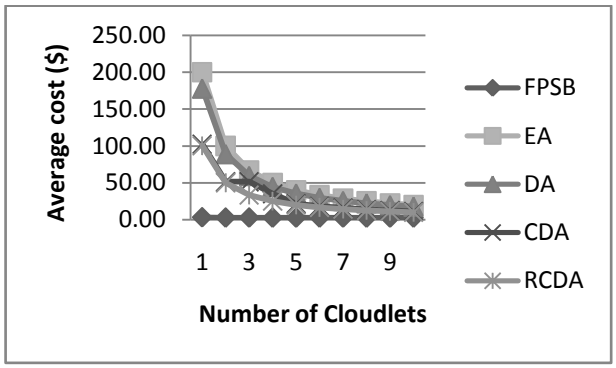

Figure 7. Impact of variation of the number of Cloudlet on their cost average.

The aim of the second set of simulation is to see the impact of varying number of users on the average cost of cloudlet, we set the number cloudlets to 20KB, and varied the number of users from 1 to 10 by one step, the Figure 8 shows that the average cost of auction DA and EA is stable for 4 users that they make high bid in the same time, while from 5 the average cost in the auction EA is up until it attains 49 $, for all bidders the price rise slowly to increase the chance of acquiring the resource, unlike the auction DA remains stable until the 7th user then increases rapidly to reach $ 70 because the bidders in this auction model tend to decrease the price set by the auctioneer by small value to trap the other competitors. When auction CDA, RCDA and FPSB (see Figures 8 (a) and 8 (b)), we note that the auction RCDA decreases the average cost more than the CDA auction and is almost stable despite the increase in the number of bidders as the auctioneer meets each according to the principle of auction CDA, unlike the FPSB auction that the average cost rises sharply from the 8th user.

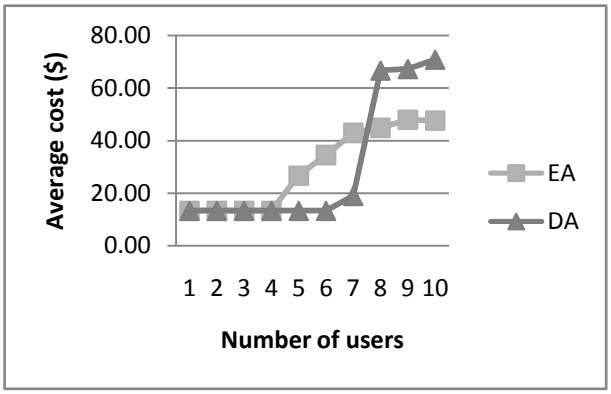

Figure 8 (a). Impact of variation of users number on the average cost of cloudlet.

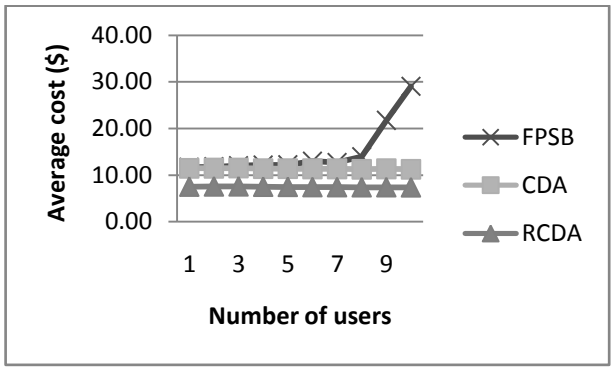

Figure 8 (b). Impact of variation of number of users on the average cost of cloudlet.

The third set of simulation has to stumble to see the impact of number de cloudlets on the average time of the auction, we observed the result of Figure 9, the entire auction models take a long time in terms of increasing the number of Cloudlet, but the auction proposed is much faster than other auction templates hence the advantage of CDA and DA auction, FPSB auction thirdly because the price generated by the user is compared with the reserve price and the supplier awards directly, the EA auction places a great deal of time versus the auction DA, as suppliers try to maximize the gain as possible without having a price limit.

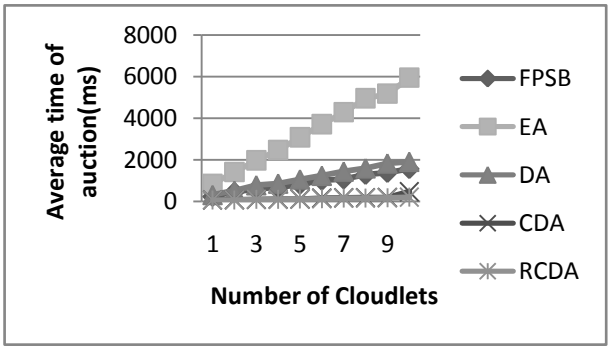

Figure 9. Impact of variation of cloudlet number on the average time of the auction.

The goal of the 4th set of simulation is to see the impact of varying the number of users on the average time of auction, the Figure 10 shows that increasing the number of users has a negative impact on the average time of the auction, but RCDA and CDA auctions are the fastest of all models of the auction saw their principles operation, also note that the auction EA and DA coincide to the extent [6, 150000] that means only six (06) both users are bidding the same time, but beyond six users of the EA auction places less time compared to the DA auction that is due to what's competitors to raise prices as quickly as possible to break the price of other competitors.

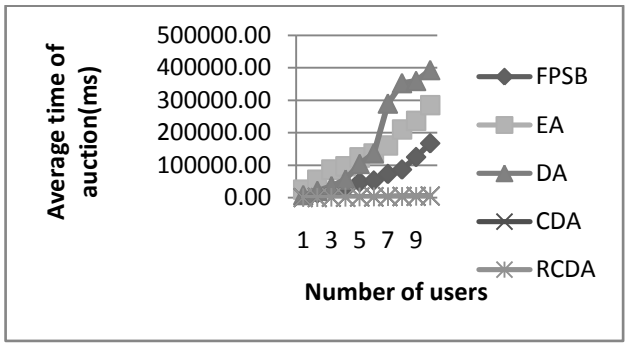

Figure 10. Impact of variation of users number on the average time of the auction.

7. CONCLUSION

The heading of a section should be in Times New Roman 12-point bold in all-capitals flush left with an additional 6-point of white space above the section head. Sections and subsequent sub- sections should be numbered and flush left. For a section head and a subsection head together (such as Section 3 and subsection 3.1), use no additional space above the subsection head. The goal of cloud computing is to provide the applications necessary to operate a business, all on the Internet. The model developed in recent years has many advantages. The main aim being saving of permanent infrastructure that same company will not have to manage, whether at the level of maintenance and infrastructure costs (electricity, air conditioning, troubleshooting, connectivity). The concept of machine no longer exists, the company stores its data and uses the cloud to work.

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.

We proposed several studies by using experiments, to analyze and study the impact of various parameters on the four types of auctions that we used, FPSB, EA, DA and CDA. Thus we have proposed a hybrid model combining these models to auction of further enhancing customers' needs in terms of QoS and performance. We have shown by this analysis that each auction can be influenced by variation of parameters. In generally, the model proposed RCDA remains the most appropriate as the rest standard models studied.

For a continuation of our work, several perspectives can be envisaged:

- Proposal for a new approach that can satisfy both the customer and the provider Resources inspiring Islamic economic models.
- Proposal of an economic model based on Club auction in overall interest of the club.
- Allow negotiation multi-criteria system and intelligent decision support Auction.
- Allow multi-criteria negotiation and intelligent system for decision support auction.

8. REFERENCES

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-degree compared", in Grid Computing Environments Workshop, pp.1–10, 2008.

[2] 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, 7(3), pp.375-393, 2009.

[3] 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), Volume 14, Issue 13-15, Wiley Press, Nov.-Dec., 2002.

[4] M. D. D. Assuncao, and R Buyya, "An Evaluation of Communication Demand of Auction Protocols" in Grid Environments, The University of Melbourne, Victoria, 3053, Australia. 2006.

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

[6] X. Chu, K. Nadiminti, C. Jin, 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, 2007.

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

[8] 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).

[9] R. Buyya, C. S. Yeo, and S. Venugopal, "Market oriented 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.

[10] 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.

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

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