International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 14, No. 1 (Dec-23)

http://dx.doi.org/10.12785/ijcds/1401120

Resource Allocation in Cloud Computing with Economical Strategic Setting

Anjan Bandyopadhyay[1], Sujata Swain[1], Dipti Dash[1] and Arup Roy[2]

[1]*School of Computer Engineering, Kalinga Institute of Industrial Technology, Bhubaneswar, India*
[2]*Budge Budge Institute of Technology, Kolkata, India*

Received 29 June 2023, Revised 18 Sep. 2023, Accepted 31 Oct. 2023, Published 28 Dec. 2023

Abstract: Resource allocation and pricing techniques in cloud computing is a challenging task for every researcher. Everyone try to find out a good solutions of resource allocation and pricing in a dynamic way. Every user give their demands for taking the cloud resources for a particular time period. If the request is not fulfilled within the time period then the user ask for extra time for completion his task. The cloud computing automatically generate a reverse auction strategy for implementing the task and the task will be performed in the other cloud which have been participated in the auction. Here we have proposed Resource Allocation with Economical Strategy (RAES) as an efficient and cost-effective framework for resource allocation with pricing . We have compared our algorithm with ECON and Greedy algorithm. For simulation purpose, we keep number of users and capacity of the system constant and vary average number of CPU required by the users. We observe that number of allocations made by our algorithm is significantly good perform than the basic ECON scheduling algorithm and Greedy algorithm.

Keywords: Resource Allocation, Auction Theory, Pricing Technique, Capacity conflict

1. INTRODUCTION

A new computer paradigm called cloud computing enables users to share on-demand computing resources like CPU and RAM. Services like Dropbox and Google Docs, as well as apps like Flikr, leverage cloud computing to store documents. Because cloud computing is an infrastructure-less service, we must share a significant amount of virtual infrastructure with the users rather than installing physical equipment at their end. When sharing these resources, users confront two main challenges: the first is how to distribute system infrastructures like CPU, RAM, and other hardware across several users; the second is how to reimburse service providers for the pricey infrastructures they offer. Allocating infrastructure to users that require services has involved a lot of work. Server sharing and task scheduling across a time horizon are the two main concerns that infrastructure allocation primarily addresses. [1].

Virtual machines and container-based technology are used to share servers among users who are concerned about security [1], [2], [3]. Scheduling tasks that arrive over a time horizon is primarily driven by two goals. The first is service level objectives, or SLOs, which are addressed in [1], [4], [5], [6] (for example, meeting the deadline of the tasks under consideration). Utilizing clusters is the second goal, which is well-discussed in the literature [7], [8], [9]. Both public and private clouds have implemented the theories concerning system level problems that have developed in the literature [10], [11]. This is a strong indicator that allocation perspective has developed to some extent [1]. It is vital to design effective pricing systems (that is, how much money should be required based on the service being delivered) together with suitable allocation methods when infrastructure-less service demand from individuals (may be a person, an organisation, etc.) grows [1]. There haven't been many recent efforts in this regard, where the main emphasis is on economically viable solutions [1], [12], [13], [14], [15].
In the current cloud computing environment, the following pricing models are in use:

- Prepaid fixed prices with a certain volume of services guaranteed [1], [16], [17].
- Unit costs based on the demand for the resources being used [1], [9], [10], [11].
- Based on the spot instance price, an interesting pricing scheme is there, where an individual can bid according to his maximum willingness to pay [1], [17].

In contrast to the current pricing schemes utilised in the current deployment of cloud computing, the major challenge

E-mail address: anjan.bandyopadhyayfcs@kiit.ac.in, sujata.swainfcs@kiit.ac.in, dipti.dashfcs@kiit.ac.in

is to establish an efficient price scheme paired with resource allocation. A paradigm [1] has been put out in this regard, where a value-based efficiency is produced while developing an effective economic system. This efficiency maximize social welfare.

The research questions are:

- How can we allocate cloud resources within a stipulated time period?
- How can we allocate the resources so that it will give better efficiency and also avoid deadlock?
- How can we calculate the dynamic pricing technique?

In this setting we have addressed two other constraints namely capacity conflict and a base price issue that are slightly different than [1]. Now we can summarize our contributions as follows:

- Providing users with flexibility if they are unable to accomplish their duties within the designated time range.
- Taking care of the capacity conflict in this framework to ensure better efficiency of the system.
- Improving the number of transaction by carefully considering a base price when demand may be less than the supply.

This gives an opportunity to devise a well-designed algorithm which accepts or rejects a given bid based on the valuation given by the user, calculating minimum price that involves base price which allocates user within the time-window when demand is less than the supply, takes care of capacity conflict and giving a second chance to the request providing the flexibility if the user is unable to complete its job within the prescribed time. There is a guarantee from the service provider that if a certain request is accepted then it is for sure that the charged price will always be less than the valuation given by the user. This makes the mechanism incentive compatible. Since, in real-world applications of cloud, a system or a machine cannot have infinite capacity, therefore after sufficient allocation of resources, there may arise the problem of capacity conflict(discussed broadly in section 6.4 of this paper), our paper also deals with it under our current setting. Since, it also gives the user an opportunity to violate the prescribed deadlines, thus more flexible. In this paper, we have discussed an efficient solution to deal with the violation of deadlines made by the user based on the second-price auction and preferably they will be allocated to a lower priority cloud (which will ensure that they may at least complete their tasks by using the cloud structures.

2. Related Work

The research community has shown a great deal of interest in the creation of market mechanisms for cloud computing, especially auction mechanisms for selling cloud resources. In the recent years, many auction mechanisms have been created. Resource allocation in cloud computing with the aid of the marketing plan is a difficult task for any researcher. Particularly, the academic community is becoming increasingly interested in auction systems for exchanging cloud resources [18], [19], [20].

Early virtual machine auctions are straightforward because they are one-round auctions that assume that the cloud only offers a single type of virtual machine or that virtual machine configurations are identical up to linear scaling. Additionally, they consider the case of static virtual machine provisioning, in which the quantity and kind of virtual machines to be offered are decided upon in advance of the auction's beginning.

In the last decade, dynamic virtual machine provisioning has been investigated, in which the cloud provider decides which virtual machines to construct and how many based on demand learnt from user bid during the auction. A convex decomposition technique is used in [21] to develop a randomised auction for dynamic resource provisioning in the cloud that is accurate and ensures a low approximation ratio for social welfare. In [18], a follow-up investigation on dynamic resource provisioning is done, and online auctions are designed in which decision-making is linked in time due to set user budgets.

Cloud auctions conducted online take place after their one-round equivalents. It was built in [21] to be the first to research online cloud auction. Online auctions are designed in the [18] work, however the temporal correlation in decision-making due to projects spanning numerous time slots is not taken into account. The research in [21] examines online cloud auctions in which a user places a bid during a predetermined window of time for job execution; as a result, the scheduling dimension is absent from their solution space.

In [22], a multi-objective scheduling algorithm (OWPSO) is proposed for task scheduling in cloud computing. A strategy-proof mechanism for the resource allocation to a buyer is proposed in [23]. In Combinatorial auctions (CA) which has been widely studied problem by the researchers [24], [25], [26], [27], [28] allow service providers to sell bundle of items rather than individual item and the users (buyers) to bid on any combination of items or services. Whenever multiple buyers and multiple sellers present in the economic market a double auction mechanism is appropriate for cloud resource allocation. However, when multiple buyers and multiple sellers are present in the market a double auction mechanism is visible [18], [29], [30], [31]. The double auction mechanism is extended into the online double auction environment in [32], [33]. In [34], [35], [21] an efficient online auction mechanism was proposed where cloud user gives their bids for future cloud resources to execute its job. While studying mechanism design in cloud

computing there are literature that have worked on the batch jobs with deadlines. [21] [36], [37]. In these papers they analyze and find out the competitive ratio for non committed scheduling which does not require to finish executing a job that has started execution in the earlier phase. In [38], [37] design a truthful allocation and pricing mechanism for job scheduling with soft dead lines.

3. System Model and Problem Formulation

In this section, we formulate the problem discussed in this paper. In this model there is a service provider represented as S having the cloud infrastructure. The service provider have, say k different types of resources that he/she wants to sell. The set of k different resources is given as $r = \{r_1, r_2, ..., r_k\}$. On the other hand, we have n users represented as $U = \{U_1, U_2, ..., U_n\}$. Each user $U_i \in U$ requests a set of resources represented as $R_i \subseteq r$. The set of requests of all the n users is given as $R = \{R_1, R_2, ..., R_n\}$. Each user will have a some request based on time-window, time requirement, resource type, and their valuation. The time-window will be preferable interval of time of the user which may be greater than the time required. The system will try to allocate the resources within the time-window. Each request from the user will be associated with a bid which is a collective information for the system using which it will decide whether to reject the request or accept it. The bidding mechanism is discussed in later sections of this paper. The system will extract the bids whose start time is at time is t. Let B_t be the set of bids at time t, then B_{ti} will be the bid of i^{th} user at time t.Let n' be the requests at time t then B_t will be:

$$B_t = \{B_{t1}, B_{t2}, B_{t3}, ..., B_{tn'}\} \quad (1)$$

Now, we need to schedule these requests within their preference time with required amount of resources. A request may be rejected if it does not satisfy the criteria of the service provider and if accepted a price (P_{min}) will be charged. There is a guarantee from the service provider that P_{min} will always be lesser than the valuation of the accepted request's user. This was the first aspect of our model in which only allocation is being done.
In the second aspect of our model, there is one more challenge after successful allocation of the resources to a user say U_i. If the user U_i is unable to complete his/her task with the resources supplies in the allocated time. Then, he/she will ask for some extra time to complete his task. A possible solution for this aspect is also given in this paper.

4. Solution for the problem

The service provider will collect bundle of request R from the users U. Each request R_i have the following characteristics:-

- Quantity and type of resources.
- Time required by the user to complete his job.
- Time slot in which the user wants the resources to be allocated.
- Their valuation.

There will be n number of bids at time t and a bid will be selected by bid selection mechanism discussed in Proposed Mechanism section of this paper and will be processed. Now, we have found solutions of the two aspects of our model as discussed below,

- In the first aspect, our model dynamically sets a minimum price P_{min} that can be charged to the user u_i. A set of Prices P_t will be calculated in every possible time t_i within the requested time window T. For each price $P_i \in P$, we have considered the resources that are already promised to some user and predicting demand d at each time based on the previous history of allocations. If the P_{min} is greater than the given valuation V by the user then our model automatically rejects that request. Otherwise, it allocates the required resources at time t_i for which p_{min} was calculated. If there is a capacity conflict at the cheapest time t_i of the allocation, then our model deletes that slot from the time window T and then again calculates the minimum price P_{min} from the remaining time window. If there is a capacity c_k conflict then it rejects the request and tells the user to give some other time window.

- In the second aspect of our model, the resources are successfully allocated to the user u_i. If the user u_i is unable to complete his/her task with the resources supplied in the allocated time. Then, he/she will ask for some extra time to complete its task. Seeing the busy schedule of the main cloud, our model will automatically generate a reverse auction between the low priority clouds and allocates to them without telling the user. The user may think that he/she is getting resources from the main cloud. If the winning cloud charges c_w amount for the resources then total expenditure of the user will be $P_{min} + c_w$ for the resources.

5. Proposed Mechanism

A. Bidding Mechanism

1) Bid of the i^{th} user at time t and its consequences:

A bid of the user can be defined as follows:

$$B_{ti} = \{n, t_y, start, end, T, V\} \quad (2)$$

n = number of resources
t_y = type of resources
$start$ = start time
end = end time
T = required time
V = valuation given by the user
A bid from the i^{th} user u_i will be given to the system. Then the system will decide whether to accept it or not. Let us consider a basic allocation without considering the capacity conflict (which will be discussed later in this paper). If V is greater than or equal to P_{min} then accept it. Otherwise V

is less than P_{min} then reject it. where P_{min} is the minimum cost calculated by the system based on preliminaries factors discussed in the previous section.

B. Bid Selection Mechanism

At the time of allocation, all the bids having common time-window will be extracted from B_t, then all the bids will be sorted in decreasing order of their $V*n$ value. Now, all the bids will be processed in the system until one bid gets accepted and gets an allocation from the system. All the remaining bids at time t will now be treated as the bids of time $t+1$ including the bids coming at time $t+1$. A request once rejected, the user have to request again with an improved bid.

1) An illustrative example

Suppose there are 5 requests comes at time t,

$$B_{t1} = \{5, CPU, 10am, 5pm, 3hrs, Rs10000\} \quad (3)$$

$$B_{t2} = \{7, CPU, 10am, 3pm, 2hrs, Rs5000\} \quad (4)$$

$$B_{t3} = \{8, CPU, 10am, 5pm, 3hrs, Rs7000\} \quad (5)$$

$$B_{t4} = \{4, CPU, 10am, 5pm, 3hrs, Rs4000\} \quad (6)$$

$$B_{t5} = \{25, CPU, 10am, 5pm, 3hrs, Rs3000\} \quad (7)$$

Now, let the system generated price P_{min} be $Rs35000$. Now, there are two sets of bids having same time window,

$$S_1 = \{B_{t1}, B_{t3}, B_{t4}, B_{t5}\} \quad (8)$$

$$S_2 = \{B_{t2}\} \quad (9)$$

Allocations for S_1 and S_2 will be done independently. For S_1, sorting the bids based on their $V*n$ value we get:

$$S_1 = \{B_{t5}, B_{t3}, B_{t1}, B_{t4}\} \quad (10)$$

Now, B_{t5} will be processed first. Since the valuation of B_{t5} is less than P_{min}, the request gets rejected and the user will have to bid once again. Then B_{t3} will be processed, assuming there is no capacity conflict this request gets accepted since $V >= P_{min}$. Now, remaining bids will be added in the set B_{t+1} and will be processed similarly. In the same way bids in S_2 will be processed in parallel with S_1.

C. Unit price and P_{min} calculation mechanism

Suppose a request comes 'I need n CPU's for T hours somewhere between a time-window $\{start, end\}$. Our algorithm will think of it as a request for each time and each resource and will set a price for the same using an auction among the prices. It means that there will be auction between the prices for $n*(end-start)$(resource*secs) times. The winning price will be the allocated price for that time for that resource. Let w_i be the winning price for time t and i^{th} resource. This is called the unit price. Calculation of this unit price will be as follows:
A demand prediction mechanism will be there which will be dependent on current time and prices. A snapshot at some time t is represented as follows in the form of table:-

Demand:	d_1	d_2	d_3	——	d_n
Prices:	p_1	p_2	p_3	——	p_n

Now a condition should be formulised in such a way that it takes care of future externilities imposed on the system after accepting this job. This condition is the backbone of unit price calculation.The condition is:

$$A : demand_t(p) + AP[t] + i > Capacity \quad (11)$$

where, $demand_t(p)$: Demand of resource 'p' at particular time 't'
$AP[t]$: At particular time frame "t", required resource is given by user.
i: cost of the resource 'p'
$Capacity$: Maximum resource capacity

For a time t, $AP[t]$ will be fixed and i will be increasing for each resource allocation. Now, the maximum price among the set of prices that would qualify the condition A will be called the winning price for i^{th} price resource at time t that is w_i If there is no winning price , then w_i will be the base price.
Then the total price obtained for time t for n resources will be:

$$c_{t'} = \sum_{i=1}^{n} w_i \quad (12)$$

Similarly, there will be a price calculation at each time of the requested window. Now, to calculate the total price of allocation for T hours will be:

$$P_t = \sum_{t=start}^{end-T} \sum_{t'=t}^{T} c_{t'} \quad (13)$$

The above equation will give a price for allocation at each possible time. The minimum price among all t's will be the P_{min}.

1) Need of base price

The auction done here may result into no winning price. In such a case base price which is fixed will be the value of w_i at that iteration. The auction done here is based of earlier promised resources and demand of that resource at that time. Therefore, this auction will ensure that highest demand price should be charged. But in case if there was no demand of that resource at that time, and earlier promised resources are sufficiently less than the capacity, then no price will win in that case. Therefore, the system will charge a minimum base price with fixed profit margin.

D. Decision Method

In this section, we will discuss how the system decides whether to accept a given request or to reject it. Suppose a request B_t arrives at time t, and say B_{tm} is the selected

bid for allocation. Now, the system generated price P_{min} will now be compared by corresponding valuation V of the user. If $V >= P_{min}$ then the given bid has passed the first stage of allocation process. Now, the system will check for capacity conflict. If there is no capacity conflict in the system then user's request will be accepted and the user will be told the allocated time. But if, there is a capacity conflict then, the conflicted time-slot will be deleted from the user's time window and the system will apply the same process in the remaining time-window, and accept if there is no capacity conflict, otherwise reject the request.

E. Flexibility for the user

Suppose, the user has been succesfully allocated to the user. Now, the user is unable to complete its task with the resources supplied in the allocated time. Then, he might ask for some extra hours for completion of its task. Seeing the busy schedule of the main cloud, system will automatically generate a second price reverse auction between low-priority clouds and allocates to them without telling the user. The user may think he is getting resources from the main cloud. This mechanism will increase flexibility for the user that if he is unable to complete his job in the allocated time, he will not have to request again by measuring the amount of resources required and a new time-window. He will be supplied the same set of resources for extra time but from low-priority clouds not with the main cloud. Let's say if winning price for the auction done is c_w then it will be the extra charge which the user have to pay for the extension of time. This solution will be better than penalty system in which there was a fixed charge based on degree of violation of deadlines.

F. Proposed scheduling and pricing algorithm:

In this section we have proposed RAES algorithm for pricing and scheduling of cloud resources for the framework discussed in the earlier sections. The algorithm is terms as Resource Allocation with Economic Strategy (RAES). The RAES algorithm has four major components:

- Main_Routine
- Bid_Selection_Mechanism
- Basic_Reservation_Mechanism
- Execute_Function

The $SelectBid()$ returns a qualified bid from a set of available bids at some time t (*i.e* B_t). Now, the selected bid goes for the reservation process through $Reserve()$ function in the Basic Reservation Mechanism part of the algorithm. If $Reserve()$ returns $False$, then it signifies that the given bid is rejected mid-where by the system. Therefore, this bid will be added in the rejected list. If $Reserve()$ function returns $True$, then it signifies that the valuation given by the user is greater than P_{min} and there is no capacity conflict at the time of scheduling. Therefore, this bid gets allocated at the time t^* and charged a price P_{min}. Now, after this allocation, the remaining requests in the list B_t will be added in $B_{t+t'}$ where t' is the time consumed in the reservation of the just reserved request.

Algorithm 1 Main_Routine

```
for each t where R! = NULL do
    Extract B_t from R
    while SelectBid(B_t)! = NULL do
        B_tm ← SelectBid(B_t)
        if Reserve(B_tm) == True then
            execute(B_tm, t*, P_min)
            B_{t+t'}.add(B_t − B_tm − rejected)
            break
        else
            rejected.add(B_tm)
        end if
    end while
end for
```

Algorithm 2 Bid_Selection_Mechanism

```
SelectBid(B_t)
S.sort()
//sorting based on their V*n value
B_tm ← S.first()
S.pop()
return B_tm
```

G. Illustrative Example of algorithm 3

Suppose a request comes "I require 4 CPU's for 3 hrs somewhere between $10am$ to $3pm$. Now, for time 10 am and for 1^{st} resource, a snapshot of the demand function at some time t is represented as follows in the form of table:

Demand:	8	5	6	3	4
Prices:	1	2	3	4	5

Also, $AP[10am] = 4$

$$demand_{10}(1) + AP[10am] + i = 8 + 4 + 1 = 12 > Capacity \quad (14)$$

$$demand_{10}(2) + AP[10am] + i = 5 + 4 + 1 = 10 = Capacity \quad (15)$$

$$demand_{10}(3) + AP[10am] + i = 6 + 4 + 1 = 11 > Capacity \quad (16)$$

$$demand_{10}(4) + AP[10am] + i = 3 + 4 + 1 = 8 < Capacity \quad (17)$$

$$demand_{10}(5) + AP[10am] + i = 4 + 4 + 1 = 9 < Capacity \quad (18)$$

The prices which satisfy the condition are 1 and 3.
the highest price = 3 or w_i is 3.

Also, to show the existance of base price, we see an example of $11am$. Now, For $11am$ and for 2_{nd} resource:
Let $AP[11] = 0$
Let snapshot of the demand function be:

Demand:	8	5	6	3	4
Prices:	1	2	3	4	5

$$demand_{11}(1) + AP[11am] + i = 8 + 0 + 2 = 10 = Capacity \quad (19)$$

$$demand_{11}(2)+AP[11am]+i = 5+0+2 = 7 < Capacity \quad (20)$$

$$demand_{11}(3)+AP[11am]+i = 6+0+2 = 8 < Capacity \quad (21)$$

$$demand_{11}(4)+AP[11am]+i = 3+0+2 = 5 < Capacity \quad (22)$$

$$demand_{11}(5)+AP[11am]+i = 4+0+2 = 6 < Capacity \quad (23)$$

No prices will satisfy the condition therefore w_i will be the base price.

Algorithm 3 Basic_Reservation_Mechanism

```
Reserve(B_tm)
for each t in [start, end] do
    demand_t() ← the demand estimate function at t
    for each i in [1, n] do
        the highest price w_i such that demand_t(p) + AP[t] + i > Capacity
        if no highest price then
            w_i ← base price
        end if
    end for
    for each i in [1, n] do
        c_t ← c_t + w_i
    end for
end for
for each t in [start, end − T] do
    for i in [t, T] do
        P_t ← P_t + c_i
    end for
end for
P_min ← min(P_t)
t* ← argmin(P_t)
if V >= P_min and len(time_window) > T then
    if CC[t*] = 0 then
        return True with t* and P_min
    else
        B_tnew={n, t_y, start_new, end_new, T, V}
        Reserve(B_tnew)
    end if
else
    return False
end if
```

H. Execute function

Execute() function is responsible for identifying the user with the bid B_{tm} and then scheduling at time t^* for the required time T at the minimum price P_{min}. After scheduling, if the user is unable to complete his/her task within the stipulated time, then while the user requests for further allocation of the same resources for updated required time T', there will be an auction among low priority clouds and extra price c_w will be charged to the user for further allocation to the winning cloud k.

Algorithm 4 Execute_Function

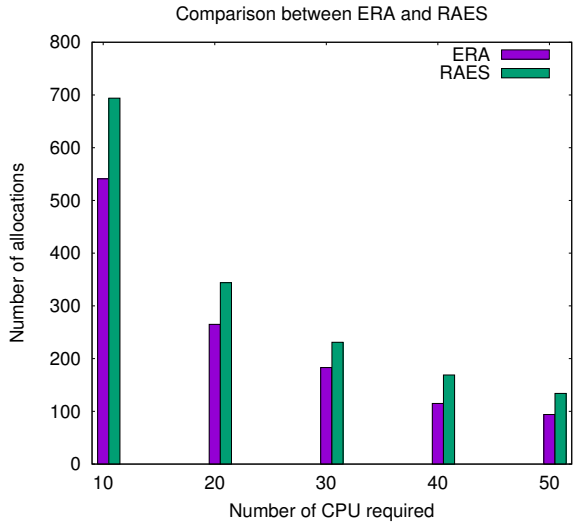
```
Execute(B_tm, t*, P_min)
u ← user with bid B_tm
u.Schedule(t*, t* + T)
u.price() ← P_min
if u.task()! = completed and t >= (t* + T) then
    while u.further_request() == True do
        k ← lp_cloud.auction()
        c_w ← k.price()
        B_tm.required_time() ← T'
        Execute(B_tm, t* + T, c_w)
    end while
end if
```

6. Simulation and Experimental Results

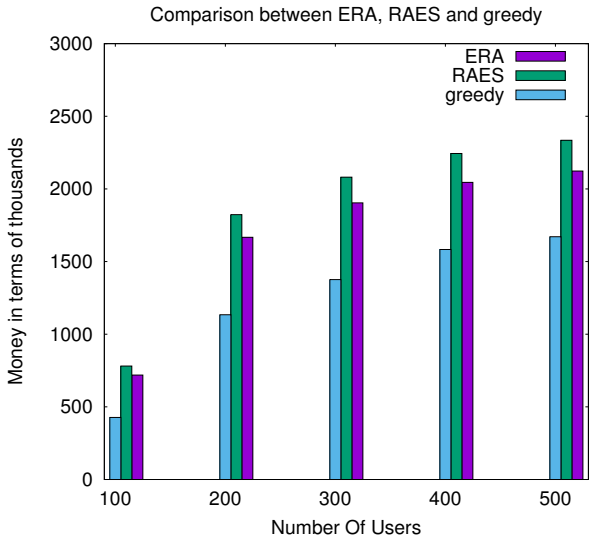
We have conducted the experiments through the help of python language in VS code. We have simulated the proposed algorithms and the parameters are demand of the resource capacity, supply of the cloud resource, pricing techniques of the cloud resources and scheduling of the cloud resources (in a particular time frame).

We have compared the econ scheduling and our algorithm. We have taken the data from http://gwa.ewi.tudelft.nl/datasets/gwa−t−12−bitbrains data set. If we keep number of users and capacity of the system constant and vary average number of CPU required by the users on x-axis, we observe that number of allocations made by our algorithm is significantly high than the basic econ scheduling. In the graph shown in the figure, for simulation we have taken requests of 4000 users. There were 800 users who required 10 CPU's, 800 users who required 20 CPU's and so on. It can be observed that our algorithm performs much better than the basic econ scheduling algorithm. The decrement of number of allocations with the increase of number of CPU's is due to limited capacity of the system. Our algorithm performs much better than the econ scheduling because it takes care of the capacity conflict and gives a second chance to the request for allocation. On the other hand econ scheduling algorithm gives only one chance to the request. Now, a question may arise that what if there is a rejection in the second chance given by the system to the request and still there is a possibility of allocation in the remaining time window. We have restricted the number of chances for a request to two. This is because giving more chances and scheduling because the system will then be biased towards the users who have given a large time window and also consume the time of the system which may have worked for allocation of other requests. Although, in our algorithm if a user has given a large time window he will have chances that he will be charged less than the user who has given a small time window. But, that is not called as biasing because both the requests have been given equal number of chances but as the first user has provided a large time window, he has parallelly provided a large number

of slots from which the system will allocate the request. Thus, there is only possibility that it may have to pay less and not an assurance from the system. But, if we use recursion rather than fixed number of chances given to the request then it will increase the number of chances for a user who has given more time window hence biasing.On the other hand, it will also increase the time complexity of the system. Thus, it leads to a conclusion that there must be a restriction on number of chances given to the request.



Next, we have compared the total money obtained in our algorithm, greedy and econ scheduling. The greedy algorithm that allocates the request which fits in the system first (First-Fit). It charges a fixed discounted price for each resource. It is observed that our algorithm performs much better than both of the greedy and econ scheduling. This is because of the two factors: (1) Number of allocations and (2) Base Price. As the number of allocations made by our algorithm is high it consequently increases the total money gained in the system. The base price is the discounted price that was given in greedy algorithm. If greedy has given $x\%$ fixed discount to each resource then our algorithm will give at most $x\%$ discount. The discount will be based on dynamic calculation and will not be fixed and have a range from 0 to $x\%$ discount. This is the reason why base price becomes a factor of increased price in our system. As econ scheduling has also not used the concept of base price therefore our algorithm makes more money than it. The money obtained shown on y-axis of the plot is obtained due the accepted requests from the n requests from the users varied on x-axis. For instance, the money obtained is due to accepted requests from the 100 requests from the users assuming one request from each user.
From the two plots, it can be concluded that our algorithm performs much better in terms of allocation and money gained due to allocation of those requests.



Discussion: In this research, the proposed mechanism is verified through theoretically. We have proposed four algorithms for the system. We have simulated the algorithms and compared with greedy algorithm and ECON algorithm. When the system works on the real world environments, the result may not be same as the simulated environment because the real world environment has several constraints like, system scalability, heterogeneity of infrastructure, workload variations, and customer behavior. We have discussed one case study through the examples. So the proposed mechanism will provide the satisfactory results when it will deploy on the real world environment.

7. Conclusion and future works

In this paper, we have proposed Resource Allocation with Economical Strategy (RAES) to allocate the cloud resources. In this mechanism, we have considered dynamic pricing technique. This mechanism is an efficient and cost-effective framework for resource allocation. Let us say, a user has requested resources for certain amount of long time window. Every user give their demands for taking the cloud resources for a particular time period. If the request is not fulfilled within the time period then the user ask for extra time for completion his task. The cloud computing automatically generate a reverse auction strategy for implementing the task and the task will be performed in the other cloud which have been participated in the auction. We have compared our algorithm with ECON and Greedy algorithm. The system should set a limit for the user for providing time window. Whether this limit be static or dynamic. Should it be told to the user or not. In our future work we are trying to implement combinatorial auction mechanism for multiple resource allocation with in particular time frame.

8. Data Availability

In this paper, we have taken data from http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains data-set

which are available in the Internet. In this data set we have taken the details of CPU and access time of the CPU.

References

[1] M. Babaioff, Y. Mansour, N. Nisan, G. Noti, C. Curino, N. Ganapathy, I. Menache, O. Reingold, M. Tennenholtz, and E. Timnat, "Era: A framework for economic resource allocation for the cloud," in *Proceedings of the 26th International Conference on World Wide Web Companion*, 2017, pp. 635–642.

[2] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud container technologies: a state-of-the-art review," *IEEE Transactions on Cloud Computing*, vol. 7, no. 3, pp. 677–692, 2017.

[3] J. Park, D. Kim, and K. Yeom, "An approach for reconstructing applications to develop container-based microservices," *Mobile Information Systems*, vol. 2020, pp. 1–23, 2020.

[4] J. Ding, R. Cao, I. Saravanan, N. Morris, and C. Stewart, "Characterizing service level objectives for cloud services: Realities and myths," in *2019 IEEE International Conference on Autonomic Computing (ICAC)*. IEEE, 2019, pp. 200–206.

[5] M. Kaminski, E. Truyen, E. H. Beni, B. Lagaisse, and W. Joosen, "A framework for black-box slo tuning of multi-tenant applications in kubernetes," in *Proceedings of the 5th International Workshop on Container Technologies and Container Clouds*, 2019, pp. 7–12.

[6] H. Hamzeh, S. Meacham, K. Khan, K. Phalp, and A. Stefanidis, "Ffmra: a fully fair multi-resource allocation algorithm in cloud environments," in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2019, pp. 279–286.

[7] O.-M. Ungureanu, C. Vlădeanu, and R. Kooij, "Kubernetes cluster optimization using hybrid shared-state scheduling framework," in *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems*, 2019, pp. 1–12.

[8] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Towards understanding heterogeneous clouds at scale: Google trace analysis," *Intel Science and Technology Center for Cloud Computing, Tech. Rep*, vol. 84, pp. 1–21, 2012.

[9] S. Machiraju, S. Gaurav, S. Machiraju, and S. Gaurav, "Introducing the cloud computing platform," *Hardening Azure Applications: Techniques and Principles for Building Large-Scale, Mission-Critical Applications*, pp. 1–41, 2019.

[10] R. Yang, C. Hu, X. Sun, P. Garraghan, T. Wo, Z. Wen, H. Peng, J. Xu, and C. Li, "Performance-aware speculative resource oversubscription for large-scale clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 7, pp. 1499–1517, 2020.

[11] S. Chawla, J. B. Miller, and Y. Teng, "Pricing for online resource allocation: Intervals and paths," in *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2019, pp. 1962–1981.

[12] I. Lee, "Pricing schemes and profit-maximizing pricing for cloud services," *Journal of Revenue and Pricing Management*, vol. 18, pp. 112–122, 2019.

[13] S.-H. Chun, "Cloud services and pricing strategies for sustainable business models: analytical and numerical approaches," *Sustainability*, vol. 12, no. 1, p. 49, 2019.

[14] R. Bhan, A. Singh, R. Pamula, and P. Faruki, "Auction based scheme for resource allotment in cloud computing," *Digital Business: Business Algorithms, Cloud Computing and Data Engineering*, pp. 119–141, 2019.

[15] T. Ni, Z. Chen, L. Chen, S. Zhang, Y. Xu, and H. Zhong, "Differentially private combinatorial cloud auction," *IEEE Transactions on Cloud Computing*, 2021.

[16] G. George, R. Wolski, C. Krintz, and J. Brevik, "Analyzing aws spot instance pricing," in *2019 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2019, pp. 222–228.

[17] G. Portella, E. Nakano, G. N. Rodrigues, and A. C. Melo, "Utility-based strategy for balanced cost and availability at the cloud spot market," in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*. IEEE, 2019, pp. 214–218.

[18] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. Lau, "An online auction framework for dynamic resource provisioning in cloud computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 1, pp. 71–83, 2014.

[19] X. Zhang, Z. Huang, C. Wu, Z. Li, and F. C. Lau, "Online auctions in iaas clouds: Welfare and profit maximization with server costs," *IEEE/ACM Transactions On Networking*, vol. 25, no. 2, pp. 1034–1047, 2016.

[20] Q. Wang, K. Ren, and X. Meng, "When cloud meets ebay: Towards effective pricing for cloud computing," in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 936–944.

[21] H. Zhang, H. Jiang, B. Li, F. Liu, A. V. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," *IEEE Transactions on Computers*, vol. 65, no. 3, pp. 805–818, 2015.

[22] A. Chhabra, K.-C. Huang, N. Bacanin, and T. A. Rashid, "Optimizing bag-of-tasks scheduling on cloud data centers using hybrid swarm-intelligence meta-heuristic," *The Journal of Supercomputing*, pp. 1–63, 2022.

[23] Y. M. Teo and M. Mihailescu, "A strategy-proof pricing scheme for multiple resource type allocations," in *2009 International Conference on Parallel Processing*. IEEE, 2009, pp. 172–179.

[24] L. Mashayekhy and D. Grosu, "Strategy-proof mechanisms for resource management in clouds," in *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 2014, pp. 554–557.

[25] P. R. Milgrom, *Putting auction theory to work*. Cambridge University Press, 2004.

[26] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge University Press, 2007.

[27] G. Baranwal and D. P. Vidyarthi, "A fair multi-attribute combinatorial double auction model for resource allocation in cloud computing," *Journal of systems and software*, vol. 108, pp. 60–76, 2015.

[28] I. Fujiwara, K. Aida, and I. Ono, "Applying double-sided combina-

tional auctions to resource allocation in cloud computing," in *2010 10th IEEE/IPSJ International Symposium on Applications and the Internet*. IEEE, 2010, pp. 7–14.

[29] D. Lehmann, L. I. Oćallaghan, and Y. Shoham, "Truth revelation in approximately efficient combinatorial auctions," *Journal of the ACM (JACM)*, vol. 49, no. 5, pp. 577–602, 2002.

[30] A. Archer, C. Papadimitriou, K. Talwar, and É. Tardos, "An approximate truthful mechanism for combinatorial auctions with single parameter agents," *Internet Mathematics*, vol. 1, no. 2, pp. 129–150, 2004.

[31] A. Mu'Alem and N. Nisan, "Truthful approximation mechanisms for restricted combinatorial auctions," *Games and Economic Behavior*, vol. 64, no. 2, pp. 612–631, 2008.

[32] S. Ibrahim, B. He, and H. Jin, "Towards pay-as-you-consume cloud computing," in *2011 IEEE International Conference on Services Computing*. IEEE, 2011, pp. 370–377.

[33] Y. Bartal, R. Gonen, and N. Nisan, "Incentive compatible multi unit combinatorial auctions," in *Proceedings of the 9th conference on Theoretical aspects of rationality and knowledge*, 2003, pp. 72–87.

[34] S. Zaman and D. Grosu, "Combinatorial auction-based dynamic vm provisioning and allocation in clouds," in *2011 IEEE Third International Conference on Cloud Computing Technology and Science*. IEEE, 2011, pp. 107–114.

[35] H. Zhang, H. Jiang, B. Li, F. Liu, A. V. Vasilakos, and J. Liu, "A framework for truthful online auctions in cloud computing with heterogeneous user demands," *IEEE Transactions on Computers*, vol. 65, no. 3, pp. 805–818, 2015.

[36] B. Lucier, I. Menache, J. Naor, and J. Yaniv, "Efficient online scheduling for deadline-sensitive jobs," in *Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures*, 2013, pp. 305–314.

[37] R. Zhou, Z. Li, C. Wu, and Z. Huang, "An efficient cloud market mechanism for computing jobs with soft deadlines," *IEEE/ACM Transactions on networking*, vol. 25, no. 2, pp. 793–805, 2016.

[38] Y. Azar, I. Kalp-Shaltiel, B. Lucier, I. Menache, J. Naor, and J. Yaniv, "Truthful online scheduling with commitments," in *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, 2015, pp. 715–732.

Dr. Anjan Bandyopadhyay is an Assistant Professor of Kalinga Institute of Industrial Technology, Bhubaneswar, Odisha, India. He has completed his Ph.D. from NIT, Durgapur, West Bengal, India in Vishveshwarya Ph.D. Fellowship under MHRD. He has completed his M. Tech in Information Security from the Department of Information Technology, NIT, Durgapur, West Bengal, India. He is broadly interested in Algorithmic Game Theory (Mechanism Design). He has published many Conference and Journal papers in esteemed Conferences and Journals. His current research interests include Cloud Computing, Fog Computing, Metaverse, IoT, Healthcare and Crowd Sourcing. He bags a number of best paper awards in many conferences like 3PGCIC.

Dr. Sujata Swain received her B.Tech. degree in Computer Science and Engineering from BPUT University India. She also received her M.Tech. and Ph.D. degree in Computer Science and Engineering from Indian Institute of Technology Roorkee, India. She is currently an Assistant Professor in the School of Computer Science and Engineering, Kalinga Institute of Industrial Technology, Deemed to be University, Bhubaneswar. She has published many Conference and Journal papers in esteemed Conferences and Journals. Her research interests include, Service Oriented Computing, Cloud Computing, Game Theory, Metaverse, Medical Imaging, and Health-Care System.

Dr. Dipti Dash has done her MCA from BPUT University , India . She has done her M.Tech in Comp.Sc. from Utkal university, India and done her PhD in Comp.Sc. From KIIT university, India. She has done many research on Mobile Computing. Her new research interests are Cloud Computing and Machine learning. She has 8years of teaching experiences.Currently She is an Asst. Professor in KIIT University.

Arup Roy received his PhD in Computer Science and Engineering from Birla Institute of Technology, India. He is currently working as an Associate Professor at Budge Budge Institute of Technology, Kolkata, India. Previously, he worked as an assistant Professor at Amity University, Manipal University. His research interests include Recommendation Systems, Artificial Intelligence, Machine Learning etc.