Journal of Computer Science and Cybernetics, V.31, N.3 (2015), 231–243
DOI: 10.15625/1813-9663/31/3/5296

SCHEDULING ALGORITHM FOR USER REQUIREMENTS ON CLOUD COMPUTING BASE ON DEADLINE AND BUDGET CONSTRAINTS

NGUYEN HOANG HA[†] AND NGUYEN THANH BINH[‡]

Hue University of Sciences, Vietnam, [†]nhha76@gmail.com, [‡]ntbinh.tt@gmail.com

Abstract. The goal of the SaaS provider is the most profitable; the user's goal is to meet requirements as quickly as possible but still within budget and deadline. In this paper, a heuristic ACO (Ant Colony Optimization) is used to propose an algorithm to admission control, then building a scheduling algorithm based on the overlapping time between requests. The goal of both algorithms is to minimize the total execution time of the system, satisfying QoS constraints for all requirements and provide the greatest returned profit for SaaS providers. These two algorithms are set up and run a complete test on CloudSim, the experimental results are compared with sequential and EDF (Earliest Deadline First) algorithms.

Keywords. Admission control, scheduling algorithms, constraint QoS, resource allocation

1. INTRODUCTION

Cloud computing is a distributed computing model for large scale; it provides services to users by employing resources (hardware, software, storage resources, etc.) via internet. Users may employ the various resources through their requirements and pay as they use. When users send requests together with the constraints as to arrival time, deadline, budget, workload, etc. to SaaS vendors, SaaS providers use PaaS to admission control, then conduct scheduling requirements as Figure 1. PaaS provider searches for suitable resources on IaaS to logical mapping to user requirements.

Generally, the admission control and scheduling request with parameters such as arrival time, deadline, budget, workload, and penalty rate, etc. is an NP-complete problem [1]. Therefore, to give an optimal solution one must often do exhaustive search while complexity is exponential, so this method can't be applied. To overcome this disadvantage people often use heuristic methods to offer a near optimal solution as ACO method [2, 3], techniques optimized fuzzy bees [4], greedy method EDF [5, 6], ...

In cloud computing environment, users rent through internet services and pay a fee for use. Therefore, the scheduling algorithm based on constraint QoS (Quality of Service) is often used. In this case, the user's parameters such as time, users' service fees, providers' service fees, reliability, etc., are given priority when scheduling. J. Deng and colleagues [7] made scheduling model for the requirements on the cloud computing environment with the goal of bringing the highest profit for the service provider but looking in detail at the two participating elements of budget and deadline requirements. The study [8, 9] focuses on the scheduling requirements for power savings on data center. The recent study by N. Ramkumar [10] of schedule in real-time requirements used for priority queues mapped into resource requirements but focused to solve scheduling tasks quickly satisfy most

of the requirements deadline regardless of cost and its budget. S. Irugurala and K. S. Chatrapati [11] make scheduling algorithm with the objective to bring the highest return for SaaS providers but considering between the two types of costs: the cost of initializing virtual machine (VM) and the fee of virtual machine which are used to select resources. In this paper, the virtual machines on the data center are used to map the requirements aiming at making real-time implementation of the system minimal but still meeting deadlines and budgets requirements. An ACACO algorithm is proposed with the goal of making real-time implementation of the system to the least in order to satisfy user and combining with this algorithms for proposing MACO algorithm to bring big profits to SaaS providers.

The article includes: building system model [section 2], building algorithm, introducing two ACACO and MACO algorithms then simulating, evaluating between the algorithms [section 3] and conclusions [section 4].

2. SYSTEM MODEL

Systems in cloud computing environment consist of components: User, SaaS providers, PaaS and IaaS. Users send requests to use the attached software to their QoS requirements to the SaaS provider. PaaS providers use component admission control here to analyze the QoS parameters and to decide acceptance or rejection of the request based on the user's abilities, the availability and cost of virtual machines. If the request is accepted, the scheduling component is responsible for locating the resources for the user's requirements such as Figure 1.

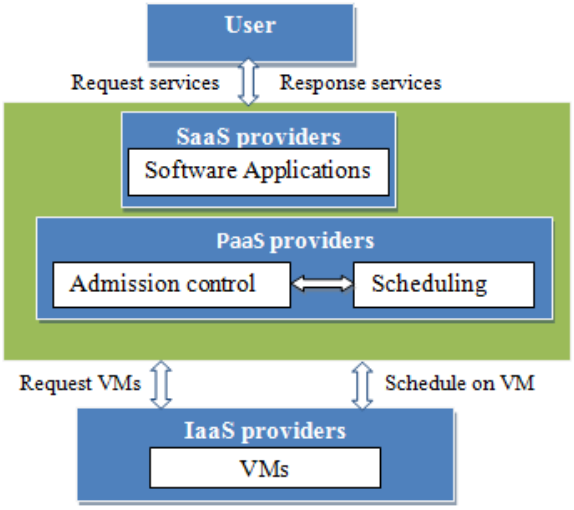

Figure 1: General model of components in cloud computing

2.1. User model

Users send N service requests $\{t_1, t_2, ..., t_N\}$ to SaaS vendors, each request $t_i(a_i, d_i, b_i, \alpha_i, w_i, in_i, out_i)$ includes the following constraints:

- a_i: Arrival time of request.
- Deadline d_i: Longest time users need to wait for the results.
- Budget b_i: The maximum cost users will pay for the services.
- Penalty rate α_i: A ratio of compensation to the user if the SaaS vendor does not provide timely.
- Workload w_i: How many MI (million instruction) are required to meet the requirements.

- Size of input and output file: in_i and out_i

2.2. SaaS providers model

SaaS providers lease resources from the IaaS provider and its leasing software as services for users. The goal of SaaS provider is how to minimize the cost of using resources from the IaaS providers to bring the highest profit to them.

2.3. IaaS provider model

In cloud computing environment with Y IaaS provider $\{x_1, x_2, ..., x_Y\}$, each IaaS provider provides M virtual machines $\{vm_1, vm_2, ..., vm_m\}$ for SaaS providers and is responsible for coordinating the VMs which runs on the their physical resources, each virtual machine $vm_{jx}(t_{jx}, p_{jx}, s_{jx}, Dtp_{jx}, Dts_{jx})$ of the vendor x attributes includes:

- Initialization time t_{jx}: How long it takes to deploy one virtual machine.
- Price p_{jx}: Pricing depends on per hour that SaaS vendors must pay for IaaS providers using VMs
- s_{jx}: Processor speed of virtual machines (MIPS)
- Dtp_{jx}: The price SaaS vendors must pay to transport data from resource provider to user's computer.
- Dts_{jx}: Data transporting speed depends on network performance.

2.4. PaaS provider model

All IaaS's resouce providers are not related to one another, can be executed in parallel and are represented by R. We set schedule for N requests independently not to follow any particular order of priority (non-preemptive) on Y providers. The requirements are denoted *npmtn*. The aim is to find the minimum total completed time for requirements but still satisfying deadline and budget of the requirements, it means that T_{min} must be found. So the model is $R \mid npmtn \mid T_{min}$

Let T_{ijx} is the time to process the request i on the virtual machine j of resource providers x. Then time T_{ijx} is determined as follows:

$$T_{ijx} = CT_{ijx} + DT_{ijx} + TI_{ijx} + \beta_{ijx} \tag{1}$$

Therein:

- CT_{ijx}: Time to process the requests depends on the workload w_i of the request i and the speed s_{jx} of virtual machines j on provider x:

$$CT_{ijx} = \frac{w_i}{s_{jx}} \tag{2}$$

- DT_{ijx}: Time to transfer data including time to send data to and retrieve data from resource providers depend on the size of the input file size in_i and output file size out_i of the request i, data transfer speed Dts_{jx} of virtual machines$j$ on provider x:

$$DT_{ijx} = \frac{in_i + out_i}{Dts_{jx}} \tag{3}$$

- TI_{ijx}: Virtual machine initialization time is given.

- β_{ij}: exceeded time deadline of the request i on virtual machine j of provider x.

Call C_{ijx} the cost of executing the request i on the virtual machine j of the resource provider x. Mean while C_{ijx} costs include costs:

- The cost of executing request CP_{ijx} depends on the price of p_{jx}, s_{jx} speed of virtual machine j of resource provider x and workload w_i:

$$CP_{ijx} = p_{jx} * \frac{w_i}{s_{jx}} \tag{4}$$

- The cost of data transmission CTD_{ijx} includes the cost of sending data to and retrieve data from resource providers depend on the size of the input file in_i and output file out_i of the request i, data transfer speeds Dts_{jx} and prices to transfer data Dtp_{jx} from the virtual machine j of the resource provider x to user computers:

$$CTD_{ijx} = Dtp_{jx} * \frac{in_i + out_i}{Dts_{jx}} \tag{5}$$

- Costs initialized CI_{ijx} virtual machine depends on the initialization time t_{ijx} and price p_{ijx} of the virtual machine j and the resource providers x:

$$CI_{ijx} = t_{ijx} * p_{ijx} \tag{6}$$

- Costs of the SaaS provider must be returned to the users if not meeting the deadline (CR_{ijx}), depending on the penalty rate (α_i) and exceeded time deadline β_{ijx}:

$$CR_{ijx} = \alpha_i * \beta_{ijx} \tag{7}$$

The goal of the paper is to construct algorithms to find the virtual machine in the data center to minimize the time of completion, such as:

$$\underset{i = 1..N}{\text{Min}} \left(\sum_{x=1}^{Y} \sum_{j=1}^{M} (CT_{ijx} + DT_{ijx} + TI_{ijx} + \beta_{ijx}) \right) \tag{8}$$

- For the profit of SaaS provider, the cost of request i must satisfy the requirements of its budget that is:

$$CP_{ijx} + CTD_{ijx} + CI_{ijx} + CR_{ijx} < b_i, i = 1 \ldots M, j = 1 \ldots M, x = 1 \ldots Y \tag{9}$$

- To satisfy the constraints of user, the execution time of request i must meet the deadline itself.

$$CT_{ijx} + DT_{ijx} + TI_{ijx} + \beta_{ijx} \leq d_i, i = 1 \ldots M, j = 1 \ldots M, x = 1 \ldots Y \tag{10}$$

Thus, to achieve the proposed goals (8), it must satisfy two constraints (9) and (10).

3. CONSTRUCTION OF ALGORITHM

The ACO algorithm is used to make a scheduling algorithm with the objective of making the total completion time to the minimum but still meet the budget and deadline of the requirements. To apply the ACO algorithm, one must determine the minimum information function F, heuristic information ηi, pheromone update and probability P [2,12].

3.1. Minima function F and heuristic information η_i

Minima function F and heuristic information η_i are used to find the best IaaS provider, depending on the time taken (T_{jx}) on the virtual machine j of resource provider x as follows:

$$F = Max(T_{jx}),\ j = 1 \dots M,\ x = 1 \dots Y \tag{11}$$

$$\eta_i = \frac{1}{T_{jx}},\ i = 1 \dots N, j = 1 \dots M,\ x = 1 \dots Y \tag{12}$$

Use η_i to find virtual machine j of the resource provider x in having highest priority because the smaller the time T_{jx} the higher the information η_i of the request i. The minima function F is used to calculate probability for the request i; selecting the virtual machine of provider x.

3.2. Pheromone update

Every ant starts from the resource provider IaaS and requests resources randomly. All ants are maintained in a list, whenever they choose the request on the next resource provider, they will be saved to the list. At each iteration of the ants, find the minima function and pheromone update as follows:

$$\tau_{ijx} = \rho\tau_{ijx} + \Delta\tau_{ijx} \tag{13}$$

Therein:

- $\Delta\tau_{ijx} = \frac{1-\rho}{F_k}$: with F_k is a minima function of the ant k, the smaller F_k the higher pheromone it gets.
- τ_{ijx}: pheromone rate of request i on the virtual machine j of resource provider x.
- $\Delta\tau_{ijx}$: added to the pheromone.
- ρ is the evaporation rate determined in the range $(0, 1)$.

3.3. Request probability

The scheduling algorithms are required to maintain two sets. A set of processing requirements and other approaching are unhandled. The algorithm is automatically started when all the requests have been executed, which would moved into the scheduling component. According to [13] first request is done and it selects providers randomly. The next request will be processing and it selects the next provider with the probability:

$$P_{ijx} = \frac{\tau_{ijx}\eta_{ijx}}{\sum_{x=1}^{Y}\sum_{j=1}^{M}\tau_{ijx}\eta_{ijx}} \tag{14}$$

Therein:

- η_{ijx}: heuristic information, τ_{ijx} pheromone rate left when moving.

- P_{ijx} is the probability to request i on the virtual machine j of resource provider x depending on a combination of heuristic and pheromone rate left when moving, basing on the formula (14) P_{ijx} which has been identified:

$$P_{ijx} = \frac{\left(\rho\tau_{ijx} + \frac{1-\rho}{F_k}\right) \cdot \frac{1}{CT_{ijx}+DT_{ijx}+TI_{ijx}+\beta_{ijx}}}{\sum_{x=1}^{Y}\sum_{j=1}^{M}\left(\rho\tau_{ijx} + \frac{1-\rho}{F_k}\right) \cdot \frac{1}{CT_{ijx}+DT_{ijx}+TI_{ijx}+\beta_{ijx}}} \tag{15}$$

3.4. Optimizing cost

Each virtual machine of IaaS providers is hired for hours and SaaS vendors must pay a fixed fee for the rental hours, if they do not use all their one-hour of hiring time, they also have to pay for a whole hour. This promotes a demand for effective positioning of costs for requests. Each vendor x can accept multiple requests, the advantage of validity period of the lease within one hour of request is taken in the same vendor to provide the highest return for SaaS providers.

The period of validity within an hired hour is called as the advance time of both requests and defines the set T_i including every request of the same provider with request t_i and put the advantage on request $t_{i.}$ All these requests can share the same virtual machine.

$$T_i = \{t_l | d_l \geq d_i \text{and} a_l < d_i\} \tag{16}$$

After identifying set T_i, the overlapping time will be calculated. t_{iljx} is defined as the effective time to calculate the request t_l after completing the request of t_i on virtual machine j of resource provider x. The value of t_{ijlx} depends on the speed of virtual machines, arrival time, deadline, and workload of t_i and t_l. t_{iljx} is calculated as follows:

$$t_{iljx} = \begin{cases} \min\left(D - U_{il}, d_l - a_l\right) \text{if} a_l - a_i \geq \frac{w_i}{s_{jx}} \\ \min\left(D - U_{il}, d_l - a_l\right) \text{if} a_l - a_i \geq \frac{w_i}{s_{jx}} \\ d_l - \left(a_i + U_{il}\right) \text{if} a_l - a_i < \frac{w_i}{s_{jx}} \text{and} d_l - a_i < D \end{cases} \tag{17}$$

Therein $U_{il} = \frac{w_i}{s_{jx}} + \max\left(a_l - d_i, 0\right)$, s_{jx} the speed of virtual machine is mapped to request t_i

3.5. ACACO algorithm

Input:

- Creation of pheromone evaporation value initial of ρ is 0.05; the value of pheromone deposit is 0.01; Number of ant (k) is used in the proposed algorithm is 10.

- $T = \{t_1,\ t_2, \ldots, t_N\}$: The set of user requests sent in, each request t_i is a 7_tuple $< a_i, d_i, b_i, p_i, w_i,\ in_i,\ out_i >$

- $X = \{x_1,\ x_2, \ldots, x_Y\}$, $VM_x = \{vm_{1x},\ vm_{2x}, \ldots, vm_{Mx}\}$: set of IaaS providers and the virtual machines of provider, every virtual machine vm_{jx} is a 5_tuple $vm_{jx}(t_{jx}, p_{jx}, s_{jx},\ Dtp_{jx}, Dts_{jx})$.

-$S = \{\}$: the set of accepted requests to schedule.

Output:

The scheduling list S contains all approved requests by SaaS provider, each request i is mapped to the virtual machine j of provider x.

Description algorithm (Algorithms 1).

Algorithm 1 Function ACACO() and Function ADMISSION_CONTROL($t_i \in T, X, VM_x$)

function ACACO()
  for Each t_i in T **do**
    s_i= **ADMISSION_CONTROL** (t_i,X,VM_x);
    if (s_i==Reject) **then**
      Inform the users that the request has been rejected.
    else
      $S = S+\{s_i\}$;
    end if
  end for
  Return S
end function

function ADMISSION_CONTROL($t_i \in T, X, VM_x$)
  ST={}
  for each ant k **do**
    for each provider x in X **do**
      Calculating heuristic information for request t_i on virtual machines vm_{jx}

$$\eta_{ijx} = \frac{1}{CT_{ijx} + DT_{ijx} + TI_{ijx} + \beta_{ijx}}$$

      Find the value of current pheromone: τ_{ijx}
      Pheromone update: $\tau_{ijx} = \rho\tau_{ijx} + \frac{1-\rho}{F_k}$
      Calculate the probability for request t_i mapped into virtual machine vm_{jx}:

$$P_{ijx} = \frac{\left(\rho\tau_{ijx} + \frac{1-\rho}{F_k}\right) \cdot \frac{1}{CT_{ijx}+DT_{ijx}+TI_{ijx}+\beta_{ijx}}}{\sum_{x=1}^{Y}\sum_{j=1}^{M}\left(\rho\tau_{ijx} + \frac{1-\rho}{F_k}\right) \cdot \frac{1}{CT_{ijx}+DT_{ijx}+TI_{ijx}+\beta_{ijx}}}$$

      From the probability on virtual machines vm_{jx}, find the virtual machine which has the highest probability, but the cost $\leq b_i$ and processing time $\leq d_i$ if found then save the request t_i, virtual machine vm_{jx} in the list ST
    end for
  end for
  if t_i is not found in the list ST **then**
    Return Reject
  else
    Find the best solution of s_i by analyzing the ants in scheduling list of ST.
    Return s_i
  end if
end function

Correctness of the algorithm

- T. Stutzle and M. Dorigo [12] proved the convergence of the ACO algorithms which ensure the convergence of proposed algorithm ACACO.

- In the cloud computing environment there is available data center, each data center has servers, each server creates multiple virtual machines. So the set of input data X and VM_x is fully determined.

- The formation of two sets T and S certainly has limit because of the requests to schedule by batch and follows a periodic basis; two sets of requirements not yet scheduled are used, this set requests scheduling then other request keeps accepting the unscheduled request and store in queue, when this request done in scheduling then the system would process scheduling for the next in queue and keep iterating.

- ACACO algorithm maps the request i into the virtual machine j of provider x (vm_{jx}) base on the probability of P_{ijx}:

$$P_{ijx} = \frac{\left(\rho\tau_{ijx} + \frac{1-\rho}{F_k}\right) \cdot \frac{1}{CT_{ijx}+DT_{ijx}+TI_{ijx}+\beta_{ijx}}}{\sum_{x=1}^{Y}\sum_{j=1}^{M}\left(\rho\tau_{ijx} + \frac{1-\rho}{F_k}\right) \cdot \frac{1}{CT_{ijx}+DT_{ijx}+TI_{ijx}+\beta_{ijx}}}$$

Therefore, whenever the time T_{ijx} lessens then the minima function F_k and heuristic information η_i are larger, this leads to the pheromone and the probability of selecting the virtual machine vm_{jx} getting higher. So, when mapped request gets into the high speed virtual machine, that makes the total time of the system decrease to the exact goal at formula (8).

3.6. MACO algorithm

Input:

$ST = \{\}$: contains a set of scheduled requirements.

UST = ACACO(): set the request has been accepted by the SaaS provider.

Output: An optimal schedule to map the request to virtual machine.

Description algorithm (Algorithm 2).

Correctness of the algorithm

The resource rental period is D-minute, therefore t_i completes its task on itself with the lesser time than D-minute, but to pay the fee of D-minute. MACO algorithm takes advantage of this effective time interval to process the next request, which makes the implementation costs of the entire system reduced, according to the objectives of SaaS providers.

3.7. Evaluation of algorithm complexity

- ACACO algorithm uses the ants to browse M virtual machine on Y provider to find resources. Thus, the complexity of algorithm is O(k*M*Y) where k is the number of ant, in the proposed algorithm, the fixed number of ant is 10, therefore k is considered as the constant.

Algorithm 2 Function MACO(UST)

function MACO(UST)

Sort all requests in UST accordingly to the provider, then all requests of the same provider will be in the same group.

for Each provide x in UST **do**

PUSH(t_i);// *Save* t_i *into the stack,* t_i *is the first request of the provider* x

ST=ST+$\{t_i\}$; S = S-$\{t_i\}$;

for Each request of t_i in the provider x **do**

t_i=POP();// *Take* t_i *from Stack*

Find $T_i = \{t_l | d_l \geq d_i$ *and* $a_l < d_i$ *and* t_l *is in the same group with*$t_i\}$

Calculate t_{iljx} for the requests in T_i as in formula (17). t_{iljx} is calculated on the virtual machine j of the provider x is mapped by the request t_i.

$$t_{iljx} = \begin{cases} \min\left(D - U_{il}, d_l - a_l\right) & \text{if } a_l - a_i \geq \frac{w_i}{s_{jx}} \\ D - U_{il} & \text{if } a_l - a_i < \frac{w_i}{s_{jx}} \text{ and } d_l - a_i \geq D \\ d_l - (a_i + U_{il}) & \text{if } a_l - a_i < \frac{w_i}{s_{jx}} \text{ and } d_l - a_i < D \end{cases}$$

Find max(t_{iljx}), t_l has the largest overlapping time as the next request.

Base on max(t_{iljx}) to find w_l reload all request status of t_l

PUSH(t_l);

ST=ST+$\{t_l\}$; S = S-$\{t_l\}$;

end for

end for

Base on ST to produce the mapped schedule onto the request of resources.

end function

- MACO algorithm use Quick Sort algorithm to sort the requests according to the provider, which then the complexity of Quick Sort algorithm is $O(2.N.\log_2^N)$, N is the number of requests. Then MACO algorithm browses through Y provider, for each provider browsing through the unscheduled request in it, for each unscheduled request in each provider, the researchers find the set T and calculate max(t_{iljx}) to figure out the request of t_l. Thus, the complexity of the three-loop is $O(Y * N^2)$. The output of the ACACO algorithm is the input of the MACO algorithm, so the complexity of MACO algorithm is $O(2.N.\log_2^N + Y * N^2)$. This complexity is still maintaining the polynomial time for the proposed algorithm.

3.8. Simulation and evaluation of algorithms.

The algorithms are installed in the Java language simulation (NetBean 7.1.1, JDK 6), CloudSim tools package [14] with the following parameters: Use 4 Datacenter, 10 physical hosts, 100 virtual machines and the number of requests change from 1000 to 5000. The parameters of users and resource providers are identified below:

3.8.1. On the user side

The inherited Cloudlet class is to create the users' requests and parameters: arrival time, workload, budget, penalty rate and deadlines. These parameters are defined as follows: The time to be taken at random from 1 to 500, the deadline is generated randomly between (d_l,d_u) minutes and the different values of d_l and d_u are limited from 10 to 1500, deadline must be greater than arrival time. Workload is taken at random from $8*10^4$ MI to 10^5MI, based on the workload the required budget is estimated, the remaining parameters are taken as implicit in CloudSim.

3.8.2. On the resource provider's side.

The researchers simulate upon four resources providers; each resource provider has a number of virtual machines, costs, speed, and different bandwidth. In simulation installation, the Vm class of CloudSim is inherited to create a virtual machines with the parameters of speed and cost are defined as follows: speed is taken at random from 10^3 to $5*10^3$ MIPS corresponding with the costs which are real numbers taken at random from 0.001 to 0.01, other parameters of the virtual machine as a virtual machine initialization time, bandwidth, etc. took default values of CloudSim.

3.8.3. Simulation Results

A. *Analyze the total cost and execution time (makespan) as fixed requirements*

Figures 2 and 3 show the total cost and the processing time of four algorithms EDF, ACACO, MACO and sequential using 100 virtual machines and 1000 requests. The values of simulation are the results of 5 tests and the average results are obtained.

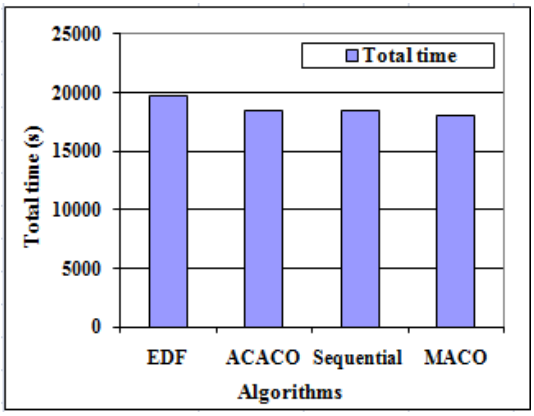

Figure 2: Total time of EDF, ACACO, sequential and MACO algorithm with fixed requests

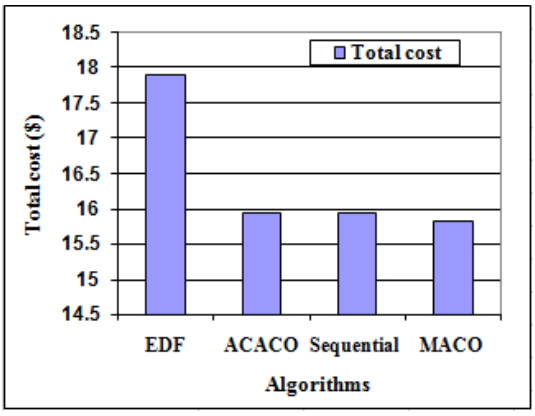

Figure 3: Total cost of EDF, ACACO, Sequential and MACO algorithm with fixed requests

Simulation results show that the total execution time of algorithms MACO and ACACO are almost as same as sequential algorithms, but the cost of MACO is always lower than EDF, ACACO, and sequential algorithms. Because the ACACO algorithm is responsible for admission control, accepting its satisfied deadline and budget requests, so after the ACACO algorithm has done processing, the accepted requests are got with less time consuming, this requesting set is the input data of MACO scheduling algorithm. MACO scheduling algorithm continues taking advantage of the advanced time

interval of requests onto IaaS resource provider, which lead to the total of processing fee reduction as Figure 3.

Sequential algorithm does not consider the overlapping period between requests, but uses exhaustive algorithm to find the resource, so there will be many cases that the request can't use all the rental time, this will make the cost of the sequential algorithm increase and take a huge amount of time to make a schedule. As EDF algorithms only consider the ratio used: $U = \sum_{i=1}^{m} \frac{C_i}{T_i} \geq 1$ (where C_i is the execution time and T_i corresponded deadline) [5,6] to map the request to the resource, thus EDF algorithm only ensures the request to complete before the deadline, but not interested in the budget of request.

B. *Analyze the total of requests that the SaaS provider is penalized*

Most of all the EDF algorithm's requests are non-penalized, because the algorithm selects the resources to complete before the deadline of request, and sequential algorithms, ACACO and MACO consider additional penalty costs while still having profit for SaaS providers then such requests are acceptable. Figure 4 shows the total number of requests that providers get fine due fixed requests is 1000.

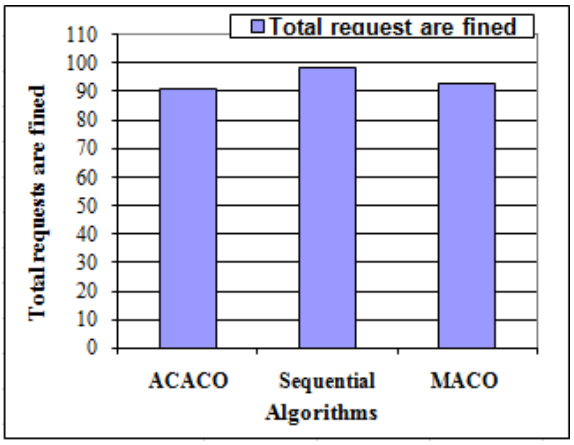

Figure 4: Total number of requests are fined for ACACO, Sequential and MACO as fixed requests

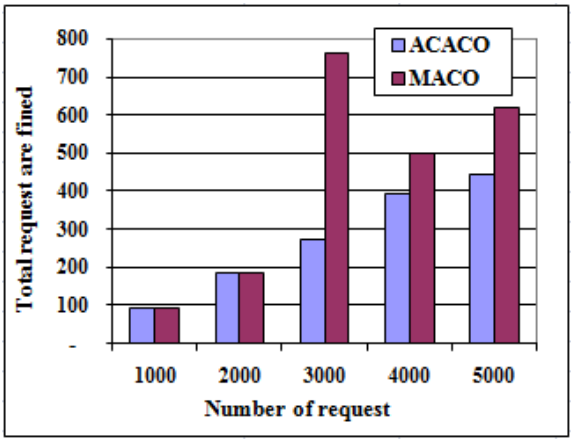

Figure 5: Total number of requests is fined for ACACO and MACO algorithm as changing requests

C. *Analyze total cost, total execution time and total number of requests that SaaS providers are fined while maintaining a fixed number of virtual machines and changing number of requests*

This section presents the results of the total cost, total time and total number of requests that are fined because of changing the number of requests from 1000 to 5000 and also maintaining the fixed number of virtual machines as 100 of the three algorithms, as shown in Figures 5, 6, and 7.

Sequential algorithm uses the exhaustive algorithm to find the resource, therefore the larger the requests are, the more time used for scheduling the complexity of algorithm will be exponential, so the sequential algorithm is not considered in this section.

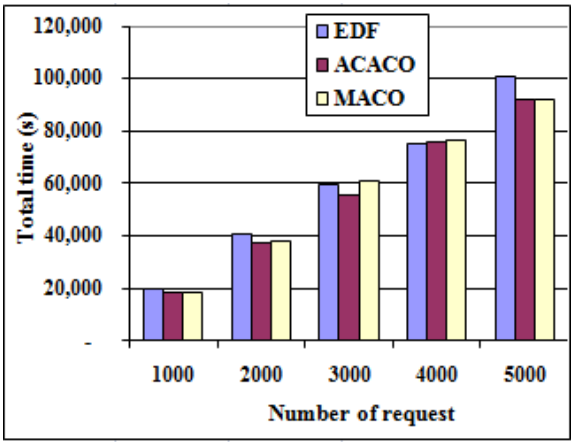

Figure 6: The total time of the EDF, MACO and ACACO algorithm when requests change

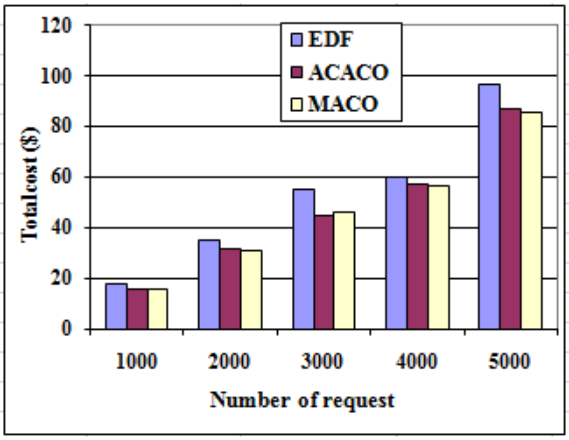

Figure 7: The total cost of the EDF, MACO and ACACO algorithm when requests change

When the number of requests increases, it will have many requests that can't use all the rental time, while MACO algorithm would be able to use all of such rental time. This will lead to the total cost of MACO algorithm is much smaller than EDF and ACACO algorithms, as shown in Figure 7. However, in some cases when the number of penalized request is larger like the case in Figure 5, then the provider must pay the higher penalized cost. This will lead to higher total cost, such as of the 3000 requests shown in Figure 7. So, the total cost of both ACACO and MACO algorithms has to be compared to decide which algorithm is best one.

As shown in Figure 6, when the number of requests gets larger, the total of processing time will increase, the total processing time of both ACACO and MACO are nearly equal, while EDF algorithm only considers to use the ratio of U to find resources, not considering in finding the best resources, so that the total execution time of the EDF is often greater than the total execution time of ACACO and MACO.

4. CONCLUSION

The article focuses on researching the admission control algorithm and scheduling for users' requests with QoS constraints, in each request, the researchers check all factors such as arrive time, cost, deadline, budget, wordload, rates of penalization, input and output file sizes; each virtual machine has speed, cost and different bandwidth. This paper proposes an ACACO algorithm to find the resources with high speed in order to bring the fastest execution time to users. In combining with ACACO algorithm, the MACO algorithm is proposed to use up all the time that the requests rented for bringing the most profit to SaaS provider.

According to the analysis and strategically experimental results of the same samples and using some CloudSim simulations show that ACACO and MACO algorithms have impressive improvement of cost and time compared to the sequential and EDF algorithm.

REFERENCES

[1] O. Elzeki, M. Reshad, and M. Elsoud, "Improved max-min algorithm in cloud computing," *International Journal of Computer Applications*, vol. 50, no. 12, pp. 22–27, 2012.

[2] M. Dorigo and T. Stützle, *Ant Colony Optimization.* A Bradford Book, The MIT Press, Cambridge, Massachusetts London, England, 2004.

[3] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in *Chinagrid Conference (ChinaGrid), 2011 Sixth Annual.* IEEE, 2011, pp. 3–9.

[4] J.-s. Lin and S.-h. Wu, "Fuzzy artificial bee colony system with cooling schedule for the segmentation of medical images by using of spatial information," *Res. J. Appl. Sci. Eng. Technol*, vol. 4, no. 17, pp. 2973–2980, 2012.

[5] A. Burns, R. I. Davis, P. Wang, and F. Zhang, "Partitioned EDF scheduling for multiprocessors using a C = D task splitting scheme," *Real-Time Systems*, vol. 48, no. 1, pp. 3–33, 2012.

[6] Ł. Kruk, J. Lehoczky, K. Ramanan, S. Shreve *et al.*, "Heavy traffic analysis for EDF queues with reneging," *The Annals of Applied Probability*, vol. 21, no. 2, pp. 484–545, 2011.

[7] J. Deng, Y. Zhao, and H. Yuan, "A service revenue-oriented task scheduling model of cloud computing," *Journal of Information and Computational Science*, vol. 10, no. 10, pp. 3153–3161, 2013.

[8] M. Mao, J. Li, and M. Humphrey, "Cloud auto-scaling with deadline and budget constraints," in *11th IEEE/ACM International Conference on Grid Computing (GRID), 2010.* IEEE, 2010, pp. 41–48.

[9] K. H. Kim, A. Beloglazov, and R. Buyya, "Power-aware provisioning of cloud resources for real-time services," in *Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science.* ACM, 2009, pp. 1–6.

[10] N. Ramkumar and S. Nivethitha, "Efficient resource utilization algorithm (ERUA) for service request scheduling in cloud," *International Journal of Engineering and Technology (IJET)*, vol. 5, no. 2, pp. 1321–1327, 2013.

[11] S. Irugurala and K. S. Chatrapati, "Various scheduling algorithms for resource allocation in cloud computing," *The International Journal of Engineering and Science (IJES)*, vol. 2, pp. 16–24.

[12] T. Stutzle and M. Dorigo, "A short convergence proof for a class of ant colony optimization algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 358–365, 2002.

[13] K. Kousalya and P. Balasubramanie, "An enhanced ant algorithm for grid scheduling problem," *Int J Comput Sci Netw Secur*, vol. 8, no. 4, pp. 262–271, 2008.

[14] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in *International Conference on High Performance Computing & Simulation, HPCS'09.* IEEE, 2009, pp. 1–11.

Received on November 11 - 2014
Revised on June 25 - 2015