Dynamic Task Scheduling in Cloud Computing Based on Naïve Bayesian Classifier

Fatemeh Ebadifard
Department of Computer (QJLQHHULQJ
University of Kashan
Kashan, Iran
Hmail: ebadifard.fatemeh@gmail.com

Seyed Morteza Babamir
Department of Computer (QJLQHHULQJ
University of Kashan
Kashan, Iran
Hmail: babamir@kashanu.ac.ir

***Abstract*—the issue of task scheduling in a cloud environment is one of the most important issues that must be considered by the cloud platform providers in data centers. The use of the right solution to solve this problem enables cloud platform providers to have the most use of available resources; and also increase the customer satisfaction by providing quality of service parameters. In this paper it has been tried to provide a dynamic scheduling algorithm using machine learning techniques and naïve-Bayes classifier in a cloud environment. The proposed method is one of the dynamic task scheduling methods and load distribution at any moment is conducted according to the latest information from previous and current server status. The distinction of this method with previous studies is the use of data mining techniques (classification) in load distribution. Since this classification method has higher accuracy and speed compared with other methods, therefore this classifier helps us to achieve the optimal solution in less time. Simulation results show that the proposed method has a good improvement in terms of Makespan time and load balancing degree.**

Keywords—task scheduling; cloud computing; naïve-Bayes classifier; Virtual machine; Makespan

I. INTRODUCTION

Cloud environment provides a huge context of servers in the data center, so that when users request resources, provides them in shared mode. To enable the applications to use the resources in accordance with their requirements, an appropriate mechanism to distribute requests to virtual machines is required. That is why the scheduling of the requests is one of the most important issues in cloud environment and in recent years it has attracted much attention of researchers. Task scheduling in cloud computing means optimal allocation of requests to the computing resources in the data center [1]. In scheduling, tasks are allocated to different types of virtual machines with regard to the limitations specified by the user and service provider. One of the major challenges in task scheduling is the equitable distribution requests on the resources according to application requirements. Providing scheduling algorithm with the aim of load balancing can reduce makespan time and increase productivity of machines.

In this paper, a dynamic task scheduling algorithm has been proposed to increase the load balancing in the cloud environment. Using Naïve Bayesian classifier technique, the proposed algorithm has tried to put the requests on the machines in a balanced way; as in addition to the reduction of makespan time, increases the efficiency of resources. The advantage of the proposed algorithm in the above method is to reduce the overload and increase resource utilization. Simulation results show that this method performs well in completion time of the longest task and increasing the level of load balancing. In summary it can be said that our main focus in this article includes the following:

1. Providing an appropriate method for scheduling the requests using data mining techniques to reduce makespan time and increase resource utilization;
2. The use of classification techniques for equitable distribution of requests;
3. Targeted analysis to show the effectiveness of the proposed algorithm, compared to previous algorithms;

The remainder of this paper consists of the following sections: Section 2 reviews related work, in section 3 after the introduction of Naïve Bayesian classifier and formulation of the problem, details of the proposed method is described in detail. In Section 4, simulation and evaluation of the proposed method is presented and finally, in Section 5, conclusions and future works are expressed.

II. RELATED WORK

Different studies are carried out in conjunction with load balancing in cloud environment, load balancing algorithms are generally divided into two categories: static and dynamic: In the static method, the allocation of tasks to virtual machines is based on the functionality of virtual machine and initial condition of each machine; in other words, this process is only based on the data of the nodes and their features. This information includes the amount of processing power, internal memory and storage capabilities and the power of communications between other virtual machines. An important feature of static algorithms is that, these algorithms do not consider changes occurred dynamically on virtual machines at any moment. In addition, they do not have the ability to adapt to changing workloads on each virtual machine over time. Some static algorithms are Round Robin (RR) algorithms or weighted RR or load balancing algorithm using ant colony algorithm [2] or procedures based on the amount of the resources of physical machines [3].

Unlike static algorithms, dynamic method of distribution in addition to the basic functionality of each virtual machine,

assigns tasks to virtual machines based on the current status of the machine and the workload on it. Such algorithms are required to review each moment of machines and based on the results of this review, the requests are transferred from one machine to another. These methods, however, are of greater complexity than the static method, but they are more efficient [4]. A lot of related work has been done on dynamic load balancing and each has been presented with different objective such as response time[5,6,7], scalability[6,8], reducing migration time in requests [9,10] and etc. since our objective in load balancing is reducing the response time, we mention some new studies in this field. Nakai et al. have presented a load balancing mechanism in 2014 [6] based on reserve policy to distribute requests between replicated servers. This allowed overload servers to reserve a part of the capacity of remote servers before receiving a new request and if requests were higher than the amount of shared capacity of remote server, some of requests were discarded. Simulation results show that the proposed method reduces the response time and increases load balancing. Even though the proposed method reduced the response time, still some of the requests would be discarded and that is why this method was not appropriate for our work.

Yuan et al. [7] tried to improve the performance of load balancing algorithm in 2015. They considered the network structure in addition to technical factors of load balancing. Thus, they provided a method which was efficiently applicable in network and reduced the level of overhead in network in addition to reducing response time. This proposed algorithm was also not suitable for cloud environment due to low productivity and lack of scalability.

Mittal et al. [16] provided a method for scheduling requests in 2016 with the objective of reducing makespan time using load balancing algorithm in cloud platform. They compared their algorithm with some of the most well-known algorithms of load balancing such as MIN-MIN [13], MAX-MIN [11] and improved algorithms of MAX_MIN [14, 15] and RASA [12]. Simulation results show that their algorithm has better performance than other listed algorithms. We have also compared our proposed algorithm with this algorithm.

III. PROPOSED METHOD

A. *Naïve Bayesian Classification Method*

Naïve Bayesian is a statistical method for classification. Research shows that although this method compared with other methods such as classification decision tree and selected neural network classifiers have equal efficiency; in contrast has higher accuracy and speed than other methods [17].

Assume that, there is m classes called $\{Y1, Y2, \ldots, Ym\}$ and tuple X have been given as input. Using the Classifier it is predicted that X belongs to the class with the highest posterior probability, in other words, X belongs to the class Yi if and only if the Eq. (3-1) is true.

$$P(Yi|X) \geq P(Yj|X) \quad j \in 1,..m \quad (3\text{-}1)$$

In Eq. (3-11) probability is calculated using Eq. (3-2).

$$P(Yi|X) = \frac{P(X|Yi)\times P(Yi)}{P(X)} \quad (3\text{-}2)$$

In the Eq. (3-2), P (X) is constant for all classes and only the values P (X | Yi) × P (Yi) should be highest value. To calculate P (X | Yi) under the assumptions of Naïve Bayes, it is assumed that class conditional is independent and based on previous training is obtained according to Eq. (3-3).

$$P(X|Yi) = \prod_{k=1}^{n} P(x_k|Yi) \quad (3\text{-}3)$$

In Eq. (3-3) if the values of P (X | Yi) × P (Yi) is more than most other classes in Yi class, this class is selected.

B. *Details of the proposed method*

Suppose VM = {VM1, VM2… VM_m} is a set of virtual machines used to host user requests. Also Task = {T1, T2... T_n} is a set of tasks that are intended to be run on virtual machines. Details of the proposed method for assigning requests are as follows:

1. First for initial allocation of the requests on virtual machines MAX-MIN method is used. As from the request queue the request with the highest runtime is selected and put on the machine where the runtime of this request is less. Then the requests in the queue waiting and the runtime of requests on each machine are updated. And it will continue until the completion of all requests.
2. After the initial allocation of the requests to the method by Max-Min, in this step, the load balancing status of the system is investigated. To do this, the standard deviation of the system load is obtained and thereby the load balancing of the system will be evaluated. To calculate the standard deviation of the load in the system the Eq.(3-4) can be used:

   $$\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(PT_i - PT)^2} \quad (3\text{-}4)$$

   Where m is the number of machines in the system and PT_i is the processing time of virtual machine i and PT is the processing time of the system. If the standard deviation is greater than the desired threshold, the system is unbalanced and requires a request transfer from overloaded machines to under-loaded machines. For this, the machines are divided into three categories: overload, balanced and under load and the request is then removed from the overload set and then in accordance with the following procedure they will transfer to the members of the set of under-load machines.
3. To transfer the requests from the overload class to the under-load class, Naïve Bayesian Classification Method is used. For this purpose, each of the machines of under-load class is considered as a class. Request t_k is selected from the overload class, and then Eq. (3-5) is calculated for it.

$$P(VMi|tk = compatible) = \frac{P(tk = cmpatible|VMi) \times P(VMi)}{P(t_{k=compatible})} \quad (3\text{-}5)$$

For all virtual machines $P(t_{k=compatible})$ is constant and is obtained according to Eq. (3-6).

if(Global Capacity on all VM > Requested Capacity for Task)

$$P(t_{k=compatible}) = 1$$

else

$$P(t_{k=compatible}) = \frac{-(\text{Global Capacity on all VM} - \text{Requested Capacity for Task})}{\text{Requested Capacity for Task}} \quad (3\text{-}6)$$

$P(VMi)$ For each machine is based on the utilization of the machine. Since the utilization of each machine is less, it is better to be selected, to achieve this goal, the probability of the machine with less utilization increases; therefore $P(VMi)$ is obtained according to the Eq. (3-7):

$$P(VMi) = 1 - Ucpu \quad (3\text{-}7)$$

Where $Ucpu = \frac{\text{used CPU Capacity}}{\text{ALL CPU Capacity}}$

The amount of $P(tk = cmpatible|VMi)$ is also calculated according to Eq. (3-3) based on the product of the probability of previous requests put on that machine.

As a result, using Naïve Bayesian Classification Method for transfer of requests from overload machine to under-load machines, a machine from under-load machines is selected which its compatibility with the considered request is higher than other machines and it has less utilization. This trend has continued until load balancing in the whole system. Figure (1) shows the pseudo-code of the proposed algorithm.

IV. RESULT EVALUATION

In this section we will discuss the details of the simulation of the algorithm presented in the previous section. Then through the charts the proposed method are evaluated. This fact that, the environment is software as a service and also our tools for simulation is CloudSim software [18], would be useful. This simulator allows us to create a virtualized environment and supports the allocation of resources based on the request. In fact, the core of the simulator for modeling our method is extended. Cloud computing has been simulated to assess this sector of a data center consisting 3 hosts with virtualization capabilities. In fact, it is assumed that the virtual instruments such as Xen have been installed, which can share resources. The properties of each host are according to Table (1).

16 virtual machines with different characteristics are put on this data center. Each virtual machine runs some applications with variable number of instructions between 500 and 4500. As mentioned earlier, we aim to provide comprehensive and appropriate algorithms for request scheduling in the cloud platform, as it has the minimum makespan time and maximum load balancing degree with equitable distribution requests on the virtual machines.

Input :list of tasks output : list of Vm Id for running any task;
1. Allocate Tasks to VMs Base on MAX-MIN Algorithm.
2.Calculate Standard deviation for load of all VMs
If (Standard deviation>threshold)
Group VMs based on load as UVM, BVM and OVM
if (OVM≠ φ)
Select VM by maximum processing Time in OVM
TaskID: Select task by minimum processing time in Selected VM in OVM
Calculate Probablity value Of any VM in UVM based on equation (3-5)
Sort Probablity value any VMs in UVM by ascending order.
Select VM by maximum Probablity value
VMID=selected VM id
Allocate selected task to VMID
Else
break;
3. Repeat step 2 for load of all VMs

Figure 1. pseudo code of the proposed method

TABLE I. HOST SPECIFICATION

HostId	Number of processing Cores	Processing speed(MIPS)	Ram (MB)	Hard (MB)	Bandwidth (Mbps)
1	*4*	*50000*	*204800*	*1048576*	*102400*
2	*2*	*25000*	*102400*	*1048576*	*102400*
3	*1*	*10000*	*51200*	*1048576*	*102400*

If Makespan is calculated according to the Eq. (4-1); Figure (2) indicates a comparison between the Makespan in the Round Robin method and scheduling of paper [16] and improved algorithm MAX-MIN [18] and the proposed algorithm.

$$\text{Makespan} = \max\{CT_{ij} | i \in T, i = 1,2,\dots,n \text{ and } j \in VM, j = 1,2,\dots,m\} \quad (4\text{-}1)$$

Figure. (2) Shows the use of proposed method has relatively better Ref. [16] at Makespan time. Since in ref. [16] the combination of algorithms MIN-MIN and RASA are used and in these algorithms only the runtime of the request on the virtual machine is considered but in the proposed algorithm in addition to the selection of algorithm MAX-MIN, load balancing algorithm is also used for suitable primary distribution and use of its benefits; and with the displacement of requests from overload machines and putting them on the under-load machines, based on the utilization of the machine and compatibility of the request with machine with classification techniques, the amount of Makespan decreases.

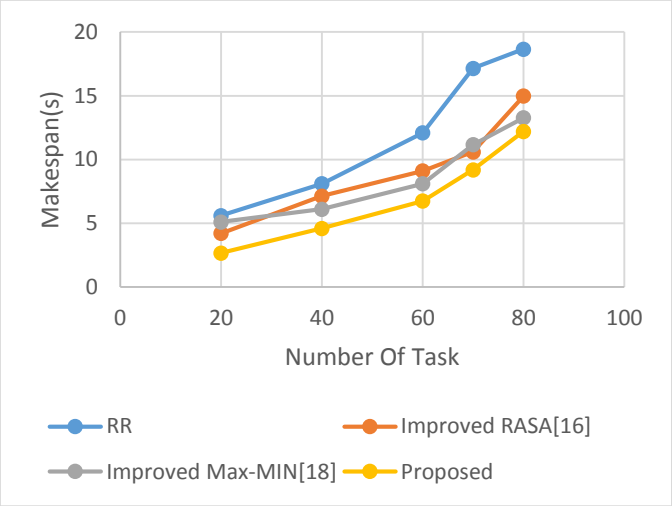

Figure 2. Makespan Comparison

Figure (3) indicate a comparison between the average response time for the requests, between the proposed method and Round Robin method and task scheduling of paper [16].

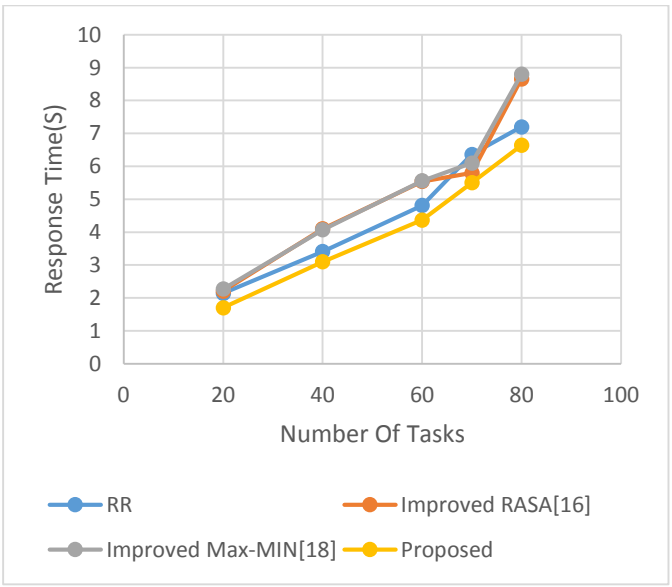

Figure 3. Response time Comparison

Another criterion which is important is the degree of imbalance defined as the Eq. (4-2) [5].

$$DI = \frac{T_{max} - T_{min}}{T_{avg}} \quad (4\text{-}2)$$

In Eq. (4-2), T_{max} and T_{min} are the highest and lowest runtimes between virtual machines and T_{avg} is the average runtime among all virtual machines. Figure. (4) Shows the comparison between the degree of imbalance in the method provided in algorithm of paper [16] and the proposed algorithm. The horizontal axis shows the number of requests and vertical axis shows the degree of imbalance.

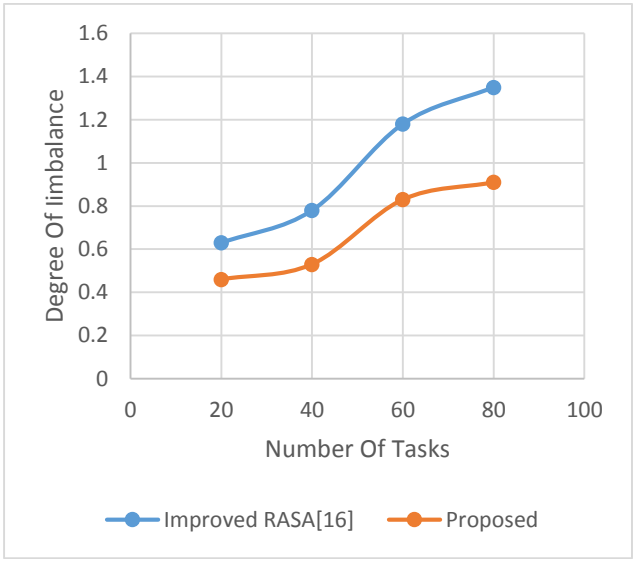

Figure 4. degree of imbalance Comparison

It is clear that the use of load balancing methods reduces the degree of imbalance and thus the proposed method has lower load balancing degree than other methods. This is due to the reduction in the runtime of the longest duration and balancing the requests between virtual machines in the proposed method. As indicated in the Figure (4), if the number of requests is lower, due to fewer machines with overload and equal longest runtime in both methods, the degree of load imbalance is close to each other, with the increasing requests of the proposed method, and decreasing runtime of the longest task and balanced distribution of requests, the degree of imbalance decreases. There are many classification methods available including linear classifiers, support vector machines, decision trees and neural networks. Figure. (5) Shows the comparison between the makespan for proposed method by using support vector machines Classification and proposed method by using Naïve Bayesian Classification.

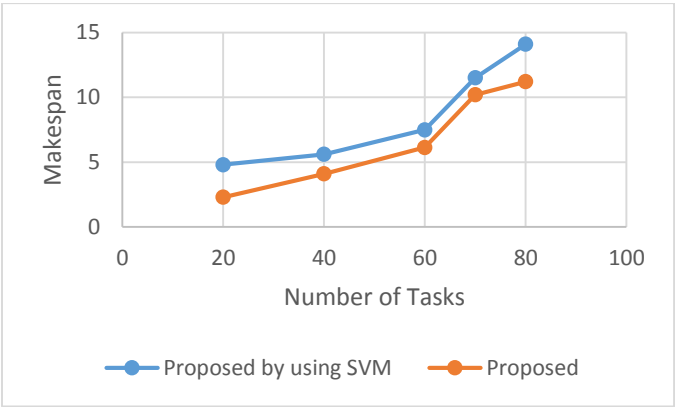

Figure 5. Makespan Comparison

V. Conclusions and future works

In this paper we have provided a task scheduling method with Naïve Bayesian Classification Method aim at creating load balancing. This algorithm with classification of virtual machines and selection of suitable machine for the existing requests reduces the makespan time and increases the level of load balancing. In the following, the proposed method is compared with the basic Round Robin method and the scheduling method [16] in terms of the criteria of Makespan, response time and load balancing level. In the future we plan to do this proposed method for workflow scheduling; and also consider other criteria such as reduced cost for the service providers.

REFERENCES

[1] K. A. Nuaimi, N. Mohamed, M. A. Nuaimi, J. Al-Jaroodi, "A survey of load balancing in cloud computing: Challenges and algorithms," in Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on, pp. 137-142, 2012.

[2] S. K. Chaharsooghi, A. H. M. Kermani, "An effective ant colony optimization algorithm (ACO) for multi-objective resource allocation problem (MORAP)," Applied Mathematics and Computation, vol. 200, pp. 167-177, 2008.

[3] M. Ajit, G. Vidya, "VM level load balancing in cloud environment," in 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), pp. 1-5, 2013.

[4] L. Guo, S. Zhao, S. Shen, C. Jiang, "Task scheduling optimization in cloud computing based on heuristic algorithm," Journal of Networks, vol. 7, pp. 547-553, 2012.

[5] P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," Applied Soft Computing, vol. 13, pp. 2292-2303, 2013.

[6] A. Nakai, E. Madeira, L. E. Buzato, "On the Use of Resource Reservation for Web Services Load Balancing," Journal of Network and Systems Management, vol. 23(3), pp. 502-538, 2014.

[7] Daraghmi, E. Y. and S.-M. Yuan. "A small world based overlay network for improving dynamic load-balancing." Journal of Systems and Software, vol. 107, pp. 187-203, 2015.

[8] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. Larus, A. Greenberg, "Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services," Performance Evaluation, vol. 68(11), pp. 1056-1071, 2011.

[9] F. Ramezani, J. Lu, F. K. Hussain, "Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization," International Journal of Parallel Programming, vol. 42(5), pp. 739-754, 2013.

[10] Y. Fang, F. Wang, J. Ge, "A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing," Web Information Systems and Mining: International Conference, WISM 2010, Sanya, China, October 23-24, 2010. Proceedings. F. L. Wang, Z. Gong, X. Luo and J. Lei. Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 271-277, 2010.

[11] T. Kokilavani, G. Amalarethinam,"Load Balanced Min-Min Algorithm for static Meta-Task Scheduling in Grid Computing," International Journal of Computer Applications (0975-8887), vol. 20(2), 2011.

[12] S. Parsa, R. Entezari-Maleki, "RASA: A New Grid Task Scheduling Algorithm," International Journal of Digital Content Technology and its Applications, vol. 3(4), pp. 91-99, 2009.

[13] G. Amalarethinam, V. Kfatheen, "Max-min Average Algorithm for Scheduling Tasks in Grid Computing Systems," International Journal of Computer Science and Information Technologies, vol. 3(2), 3659-3663, 2012.

[14] O. M. Elzeki, M. Z. Reshad, M. A. Elsoud, "Improved Max-Min Algorithm in Cloud Computing", International Journal of Computer Applications (0975 – 8887), vol. 50(12), pp. 22-27, 2012.

[15] U. Bhoi, P. N. Ramanuj, "Enhanced Maxmin Task Scheduling Algorithm in Cloud Computing," International Journal of Application or Innovation in Engineering & Management, ISSN2319-4847, vol. 2(4), 2013.

[16] S. Mittal, A. Katal, "An Optimized Task Scheduling Algorithm in Cloud Computing," in 2016 IEEE 6th International Conference on Advanced Computing (IACC), pp. 197-202, 2016.

[17] H. W. Lijuan Zhou, W. Wang, "Parallel Implementation of Classification Algorithms Based on Cloud Computing Environment," Indonesian Journal of Electrical Engineering and Computer Science, vol. 10(5), pp. 1087-1092, 2012.

[18] R. Kuar, P. Luthra, "Load Balancing in Cloud System using Max Min and Min Min Algorithm", International Journal of Computer Applications, National Conference on Emerging Trends in Computer Technology (NCETCT-2014).