LOAD BALANCING ALGORITHM ON CLOUD COMPUTING FOR OPTIMIZE RESPONE TIME

Nguyen Xuan Phi, Le Ngoc Hieu, Tran Cong Hung

Posts and Telecoms Institute of Technology, Vietnam

ABSTRACT

To improve the performance of cloud computing, there are many parameters and issues that we should consider, including resource allocation, resource responsiveness, connectivity to resources, unused resources exploration, corresponding resource mapping and planning for resource. The planning for the use of resources can be based on many kinds of parameters, and the service response time is one of them. The users can easily figure out the response time of their requests, and it becomes one of the important QoSs. When we discover and explore more on this, response time can provide solutions for the distribution, the load balancing of resources with better efficiency. This is one of the most promising research directions for improving the cloud technology. Therefore, this paper proposes a load balancing algorithm based on response time of requests on cloud with the name APRA (ARIMA Prediction of Response Time Algorithm), the main idea is to use ARIMA algorithms to predict the coming response time, thus giving a better way of effectively resolving resource allocation with threshold value. The experiment result outcomes are potential and valuable for load balancing with predicted response time, it shows that prediction is a great direction for load balancing.

KEYWORDS

Load balancing, Response time, Cloud computing, Optimize, Threshold.

1. INTRODUCTION

Cloud computing helps us share data and provide more resources for users. Users only need to pay for what they use. So cloud computing stores data and distributes resources in an open environment. The amount of data stored in the cloud is increasing rapidly on this open environment. Therefore, load balancing is the biggest challenge on cloud computing. Load balancing helps to distribute loads dynamically through network nodes to ensure that no nodes are overloaded. This optimizes the resources, improves system performance. Many algorithms have been proposed to load balancing and optimize resources. There are many types of load used on cloud computing such as: memory, CPU and load on the network. Load balancing is considered to be a process of finding out overloaded network nodes and hence switching over to other nodes that are loading little or no load. In the cloud environment [1], load balancing requires reallocating all active loads between all nodes, load balancing which enables the cloud to achieve the best allocation of resources, flexible, scalable to avoid bottlenecks to improve productivity and maximize cloud utilization.

The main factors that load balancing needs to resolving is [24]: poor infrastructure, poor network traffic management, low network reliability,...resulting imbalance in server clusters. This is essential for researchers to develop load balancing algorithms for complex network systems. The main purpose of load balancing is to distribute traffic between balanced nodes in the cluster to improve network quality, including: Enhance the reliability and satisfaction of services for users; Increased use of resources; Reduced execution time and waiting time to handle jobs coming from

DOI: 10.5121/ijccsa.2020.10302

many different positions.; Increase service performance; Maintain the stability of the server cluster; Build fault tolerance system, extensible and easy to modify.

The main result of load balancing is minimizing resource use (minimizing resource consumption), implementing fail-over, enabling enable scalability, avoid bottlenecks (eg bottlenecks), over-provisioning. In addition, improve the load balancing performance indispensable models of related parameters forecast based on data mining, hidden Markov models, probability methods, machine learning. In the works [25], [26], [27], [28], [29], [30] the authors have identified that predicting parameters directly affecting load balancing mechanism is very important. and introduced new algorithms and mechanisms to overcome the current disadvantages of existing models.

Methods is to reduce the response time of cloud services when users request access to services with aims to find strategies to save computing resources and increase user service, which directly affects the service provider's business to make a profit. It can be said that response time is a very important parameter on cloud computing. The main objective of this paper is to reduce response time on cloud computing. This is a parameter that directly affects the quality of service when users access computing resources. Improving response time means increasing cloud performance, increasing user satisfaction, thereby solving the problem of load balancing access to computing resources. This is the impetus for us to study and propose a method of load balancing to reduce response time presented in this paper.

The article includes the following sections: Secsion 1, Introduction; Section 2, Related Works; Section 3, Theoretical Basic; Section 4, Proposed Algorithm; Section 5, Simulation Results; VI, Conclusions.

2. RELATED WORKS

Syed Hamid Hussain Madni [2] has studied and evaluated the resource allocation techniques in the cloud environment. This article also outlines the importance of allocating resources in the cloud, requiring resource allocation policies, strategies, and algorithms to distribute and migrate resources to best support both both suppliers and users. According to the literature [3], the author has come up with a work load balancing approach based on QoS criteria. This article uses the Load Balanced Resource Scheduling (LBRS) algorithm to improve the quality of service on the Cloud. The response time (the time it takes to send requests and wait for replies in the queue until the first CPU is used is considered until the algorithm allocates the requests.

Agraj Sharma [4], presentation algorithm for load balancing is based on response times. The load balancing algorithm based on the response time of the server, which determines which subsequent requests will be processed by which server (VMs). Experimental results show that the proposed model is a natural dynamic load balancing method, taking into account current responses and its variables to determine the allocation of new requirements. Response time on cloud computing is very interested in [5 -14]. With such usability and performance, cloud computing has become an indispensable trend. In the future, the increase in the number of cloud users requires service providers to meet the needs of users with minimal response time. Therefore, load balancing methods in cloud computing are increasingly being developed, when number of servers or server configurations increasing is only temporary method. Effective use of resources on the "cloud" is a necessity. This is also a huge challenge in the field of cloud computing. In order to meet the above requirements, the establishment of an efficient load balancing algorithm and how to use resources in a reasonable manner is the goal that cloud computing wants to achieve.

Response time is a very important parameter, it affects the performance of the cloud as well as the QoS of the computing service providers. Therefore, there are many load balancing algorithms proposed with the goal of optimizing the response time on cloud computing [15 - 20]. Load balancing algorithm takes into account the response time of each request [15], this algorithm is not only flexible, but also reduces communication and additional calculations on each server. The scheduling algorithm [16], is designed to minimize response time by optimizing the workload weight on time slots of different Ci virtual clusters.

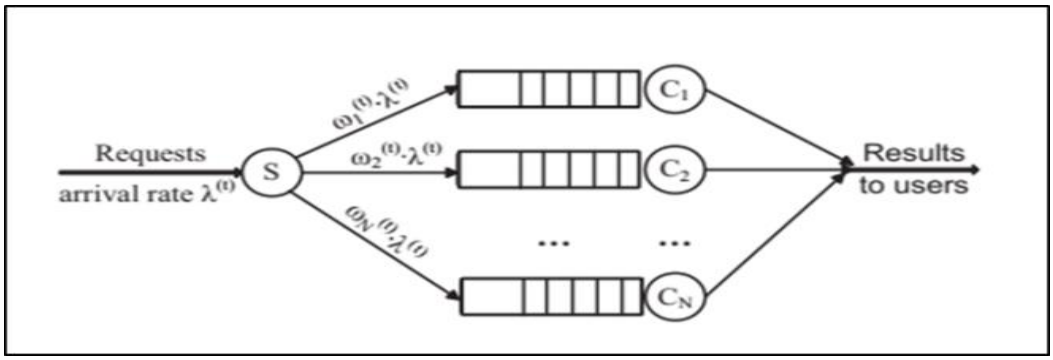


Fig 1. Cloud workload scheduling model [16].

With the ratio of requirements to the average of the time slot t modeled by distribution of Possion $\lambda^{(t)}$ và $\omega_1^{(t)}$ are corresponding weights [16].

In the WMaxMin load balancing algorithm [18], the author combined two Max - Min and Weighted Round Robin Algorithm algorithms to balance load effectively by considering two parameters: response time and waiting time. The main idea of WMaxMin is to have no overloaded or under loaded virtual machines, the load is balanced between all virtual machines and they will be delivered to the specified machine or The machine intended to handle the task, if it had to wait for any task, it would be assigned to the scheduler again to schedule the task according to the availability of the virtual machine. Simulation through CloudSim shows that the algorithm has improved response time.

In this paper [32], the author propose a multi-tier-oriented job scheduling and allocation technique. The scheduling and allocation process is formulated as a problem of assigning jobs to the resource queues of the cloud computing environment, where each resource of the environment employs a queue to old the jobs assigned to it. The scheduling problem is NP-hard, as such a biologically inspired genetic algorithm is proposed. The computing resources across all tiers of the environment are virtualized in one resource by means of a single queue virtualization.

Paper [33] presents an approach for service-level driven load scheduling and balancing in multi-tier environments. Joint scheduling and balancing operations are employed to distribute and schedule jobs among the resources, such that the total waiting time of client jobs is minimized, and thus the potential of a penalty to be incurred by the service provider is mitigated. A penalty model is sed to quantify the penalty the service provider incurs as a function of the jobs' total waiting time. A Virtual-Queue abstraction is proposed to facilitate optimal job scheduling at the tier level.

3. THEORETICAL BASIS

3.1 Load balancing problem

- **Definitions**

Load balancing problems occur when multiple servers handle a set of jobs or requests. The assumption is that all servers are identical and can be used to serve any request [21].

- **Problem:**

  - There are m machines $\{M\} = \{M_1, M_2, M_3,..,M_m\}$.
  - There are n jobs $\{J\} = \{J_1, J_2, J_3,\ldots.., J_n\}$, for each job J_i has processing time $t_j > 0$.

- **Requirement:**

Assign each job J to an M machine so that the load on all M machines achieves the optimal level of balance possible.

Let A(i) be the set of tassk J assigned to M_i. So M_i needs to work in total time [21]:

$$T_i = \sum_{j \in A(i)} t_j$$

And Ti is called the makespan (which is the time required to complete the implementation of all input tasks of the system) and we also consider it the load on Mi machines [21]. The goal of the problem is that minimizing the makespan quantity with the makespan is the maximum load on any machine, $T = \max_i T_i$. And the minimization of the makespan is a problem in the NP-hard class [21].

3.2 Response Time Definition

Response time is the total time required to respond to a service request. Within the framework of the thesis, without losing the generality, we can skip the transmission time, the response time is the sum of service time and waiting time. Service time is the time required to perform the requested job. Research suggests that response time is a key factor that has a significant impact on cloud performance. To improve the performance of cloud services, resource management faces fundamental issues including resource allocation, resource response, resource connectivity, resource discovery. unused resources, map corresponding resources, model resources, provide resources, and plan resource usage. In particular, planning for resource use based on service response time is very important. From there we can study response time to give solutions for the distribution, load balancing of resources. This is one of the promising research directions that make cloud technology more and more complete and advanced.

3.3 Solve Load Balancing Problem By Approximation Method

Load balancing problem is a problem of NP-Complete [22], there is currently no optimal solution, but can be solved through approximation algorithms. The approximation method is to search for near optimal solutions, within a reasonable amount of time. To implement the stated load balancing problem, the assumption is that all servers are identical and can be used to serve any requirement.

Considering a simple load balancing algorithm Greedy-Balance [21] to illustration the problem of load balancing by approximation method. This algorithm assigns job j to the machine with the smallest load.

Greedy-Balance:

1. *Start with no jobs assigned*
2. *Set Ti = 0 and A(i) = Ø for all machines Mi*
3. *For j = 1, . . . , n*
   Let Mi be a machine that achieves the minimum mink Tk
   Assign job j to machine Mi
   Set A(i) ← A(i) ∪{j}
   Set Ti ← Ti + tj
4. *EndFor*

This Greedy-Balance procedure runs for 6 jobs with the corresponding size of each job: 2, 3, 4, 6, 2, 2 and 3 virtual machines M1, M2, M3. From Figure 2 we see that initially virtual machines have a makespan of 8.5,6 (makespan system = 8) and after running this makespan has reduced to 6, 7, 6. The algorithm has better load balancing and has optimized load system (makespan of system = 7).

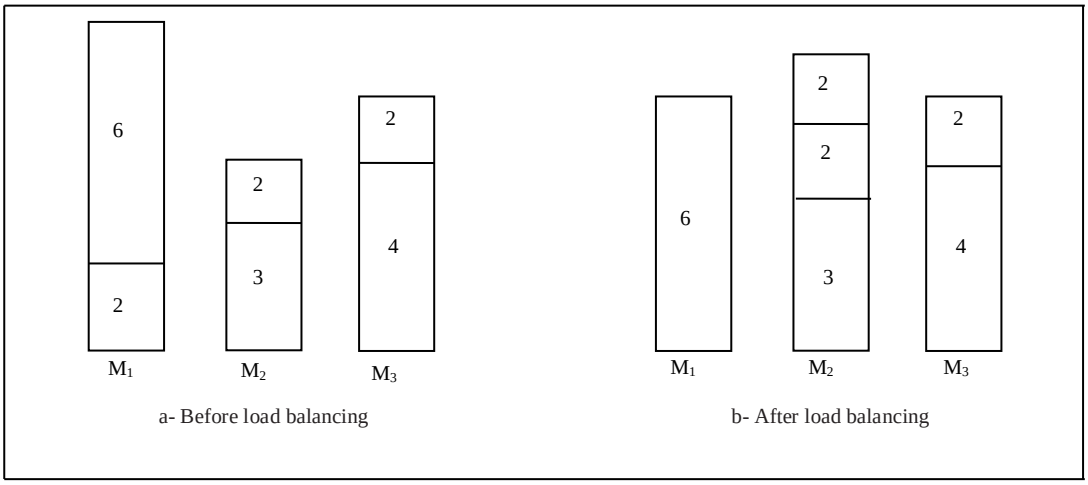


Fig 2. Results of running Greedy-Balance algorithm on three work-size machines 2, 3, 4, 6, 2, 2.

Assuming T* is the optimal makespan value, then task of load balancing algorithms is to find T with expectation that T is not much greater than T*. We cannot know value T* but it always has the lower limit. There are many ways to define the lower limit, but one of them is based on the total processing time is $\sum_j t_j$.

Considering Sorted-Balance algorithm [21], this algorithm has increase the above rate.

Sorted-Balance:

1. *Start with no jobs assigned*
2. *Set Ti = 0 and A(i) = Ø for all machines Mi*
3. *Sort jobs in decreasing order of processing times tj*
4. *Assume that t1 ≥ t2 ≥ . . . ≥ tn*

5. *For j = 1, . . . , n*
   Let Mi be the machine that achieves the minimum mink Tk
   Assign job j to machine Mi
   Set A(i) ← A(i) ∪ {j}
   Set Ti ← Ti + tj
6. *EndFor*

4. PROPOSED ALGORITHM

We present a proposed algorithm for the load balancing to reduce response time on cloud computing using the ARIMA [23] predictive algorithm to forecast response time, which efficiently allocate requests.

4.1 The Goal of the Proposed Algorithm:

- Minimize the process of transmitting information between load balancers and virtual machines with idle resources with servers.
- Reducing the response time of requests from users
- Minimize the load imbalance between virtual machines, prevent and warn before losing load balance.

4.2 Assumptions:

- The load balancer will know in advance which services are running on VMs at any time.
- Here focusing on Web Services (Web Service), web servers will anticipate the response time of each service running on the web and on each VM.
- If the two VMs have the same configuration of RAM, processor, and I / O, the execution time of the services will not be much different.

4.3 Introducing The ARIMA Algorithm

According to [23], ARIMA is an Auto Regression Integrated Moving Average, developed from the ARMA (Auto Regression Moving Average) regression model. This is a development model based on known time series data and forecasting data in the near future. Real-time data or time series is a sequence of values of a given quantity that is recorded as time. Any time series data is generated by a random process. The values of the time series of quantities X are denoted as X1, X2, X3, ..., Xt, ..., Xn where X is the value of X at time t.

Determining ARIMA model (also known as Box-Jenkin method) means determining p, d, q in ARIMA (p, d, q), because Box-Jenkins model only describes stop chains or chains Because of the variance, the ARIMA model (p, d, q) represents non-stop series of data, which have been differentiated (here, d indicates the degree of variance).

According to [31], the Box - Jenkins method includes general steps:

- Confirmation of testing model: First, we need to identify the testing model. In which: d is the integrated degree and p, q will be determined by a dedicated function, called Correlogram. Arima models can be presented in different formats. The method of determining models is often carried out by researchers through studying the changing direction of the whole or partial correlation function.

- Parameter estimation: Arima model has 2 types: ARMA (p, q) and Arima (p, d, q). For the ARMA form (p, q) there will be d = 0, so we can transform back to Arima (p, 0, q). In the parameter estimation process, we need to pay attention to how to determine p q in the Arima model. To determine these two figures, one will use the Correlogram graph. Specifically, p will be the order of the AR graphs. From the first lag, which bar is outside the limit line and after a significant reduction after a lag, the partial autocorrelation coefficient is p. Similarly, q will be the order of MA. In order to estimate the parameters, we need an initial estimate for the parameters a0, a1, ..., ap, b1, ..., bq of the original model. Then based on the estimated parameters, construct the final estimates through an iterative process.

- Verification by diagnosis: After the parameters of the general model have been built, the accuracy and relevance of the model with the established data will be checked. Consider whether the error is pure random? If so, that model is satisfactory, otherwise we will have to repeat the above steps.

- Forecasting: In this final step, when the model matches the found data, we will make the forecast at the next time. Therefore, the model of ARMA(p, q):

$$y(t+1) = a0 + a1y(t) + \ldots + apy(t - p + 1) + e(t+1) + b1e(t) + \ldots + bqe(t - q + 1)$$

We can see the advantage of ARIMA with a time series data in a near future prediction. And the response time in load balancing is also a time series data in discrete space. What we need to have a near future prediction, means forecasting in a next few seconds/ minutes only. ARIMA can adapt this better than other methods. Combining the 02 characteristics of ARIMA and response time, we are inspired to build this proposal to achieve a better forecast of response time. In a short time, if we can predict more exact so we can have a better load balancing

4.4 Research Models

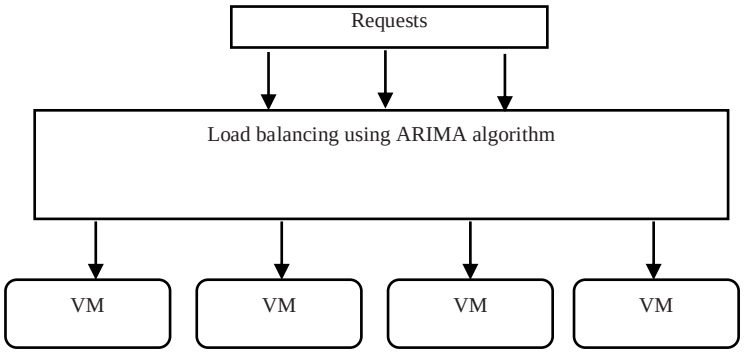


Fig.3. Load balancing model using ARIMA algorithm

In the research model, the load balancer will run the load balancing algorithm as shown in Figure 3, the load balancer with a list of virtual machines and services provided by the cloud. The load balancer knows which services are running on which virtual machines and can allocate new services on a new VM as required. In this model we use a parameter called the time threshold, based on this threshold, which anticipates the next response time. With VM has a predicted response time less than the threshold calculated from the ARIMA algorithm [23], and that the treatment with virtual machines is the same in the allocation of requests. If there are no virtual machines in the pool (a set of virtual machines) considering the threshold conditions, then the request will be allocated to the next virtual machine pool. Virtual machines with the lowest average response time and lowest predictor will be selected to process the next request.

4.5 If No Vms Meet the Threshold Conditions, We Can Handle the Following:

+ If a VM does not load, or the pool does not load, these VM can be used to handle the requests, but must meet the threshold.

+ If no VM have not load, or the pool have not load, or all of them do not meet the threshold, then we will allocate that running service to the virtual machine with the expected response time closest to the threshold.

The model apply for scheduling new requests to avoid load imbalances. This algorithm reduces the communication load between the VM and the available resources, thus reducing unnecessary bandwidth and throughput for the user.

4.6 Proposed Algorithm Based on Response Time

Based on existing time series data of response time, we use the ARIMA algorithm to predict the next response time, so we know how to allocate resources for the next request. according to the.

The proposed algorithm consists of 3 modules as follows:

- **Module 1**: Calculation of threshold by ARIMA algorithm:

In this module, the threshold is the cloud expected response time, will be calculate by expected response time with ARIMA algorithm, and will increase or decrease depending on the response time according to the time series data. The new threshold is a expected time response in the file of VM expected in most 100 request.

$$\text{New Thereshold} = \text{TNew} = \text{ARIMA}(RT1, RT2, \ldots, RT100)$$

Where Rti is the recordable response time sequence of the cloud (only within the last 100 requests).

- **Module 2**: Predict next response time for each VM:

In this module, use the ARIMA algorithm to predict the next response time of the virtual machine. The next response time prediction is based on the response time data of the last 50 requests of the virtual machine considered through the getPredictedRT () function. This module also provides a function that calculates the nearest predicted value of VMs against the input threshold through the AllocateRequestToVM (VM, Request) function;

$$\text{PRTi} = \text{Prediected Response Time} = \text{Expected response time of VMi}$$

- **Module 3***:* Select Virtual Machine

This module is allocating requests to VMs that meet the time threshold condition.

If a request is sent to the VM being considered and the VM does not load, then this request will be forwarded to the VM directly, and we get the response time value. If the VM's estimated response time (calculated from module 2) is smaller than the next response time of the cloud (calculated from module 1), the request will be processed on the VM.

This module is responsible for allocating requests to VM that meet the time threshold requirement. If a request is sent to the VM under consideration and the VM does not load, then this request is forwarded directly to the VM, and we have response time value. If the VM's estimated response time (calculated from Module 2) is less than the next response time of the cloud (calculated from module 1), the request will be processed on the VM. In contrast, no VM meets the threshold condition (the VM's predicted response time is not less than the predicted response time of the cloud), the request will be allocated to the virtual machine with the forecast closest to the threshold.

In which, the threshold calculated is the maximum response time in a set of VMs, but there will be some changes, or put into some parameters, depending on the experimental results.

5

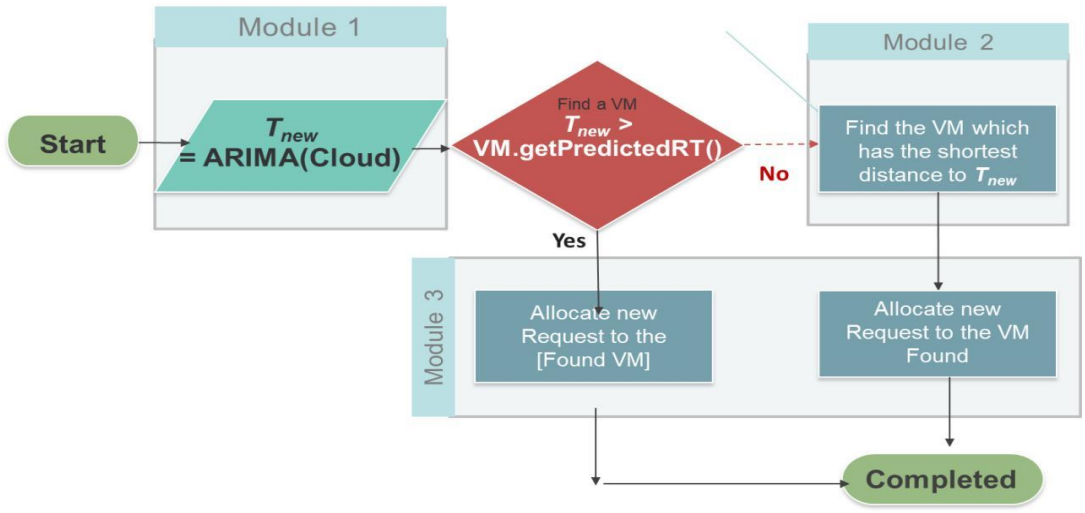


Figure 4. The process flow of proposed algorithm using ARIMA

5.1 Simulation Setup

The cloud simulation environment uses the CloudSim library and programming in JAVA language, has between 3 and 10 VMs, and creates a random request environment for these cloud services. This includes virtual machine provisioning, cloud service provisioning, and user response provisioning for simulation environment.

Setup the ARIMA algorithm on the simulation environment, and verifies the result. Similarly, set up the author's algorithm [4], and compare the results between the two algorithms. The proposed algorithm is installed on the JAVA language and uses the NETBEAN IDE to test and render the results using the STS IDE with the SPRING BOOT framework. Simulation environment with the CloudSim 4.0 open source library (provided by http://www.cloudbus.org/). The simulation environment consists of a data center with the following parameters:

Table 1. Data Center Configuration

Datacenter	Host
- Number of host in datacenter: 5 - OS architecture: x86 - OS: Linux - VMM: Xen - TimeZone: +7 GMT - Cost: 3.0 - Cost per Memory: 0.05 - Cost per Storage: 0.1 - Cost per Bandwidth: 0.1	- CPU: 4 cores, each core with processing speed is 1000 (mips) - Ram: 16384 (MB) - Storage: 1000000 - Bandwidth: 10000

Requests are represented by the cloudlet in cloudSim and the size of the Cloudlet is initialized randomly using the JAVA random function. Number of Cloudlet is 100→ 1000.

Table 2. Parameters of Request

Length of Request	File Size	Output File Size	Number of CPU (PEs)
3000 ~ 1700	5000 ~ 45000	450 ~ 750	1

The proposed algorithm was built by creating the ArimaDatacenterBroker class, inheriting from the DatacenterBroker object, updating some methods and properties related to Predicted Response Time, and adjusting the built in functions to match the algorithm proposed:

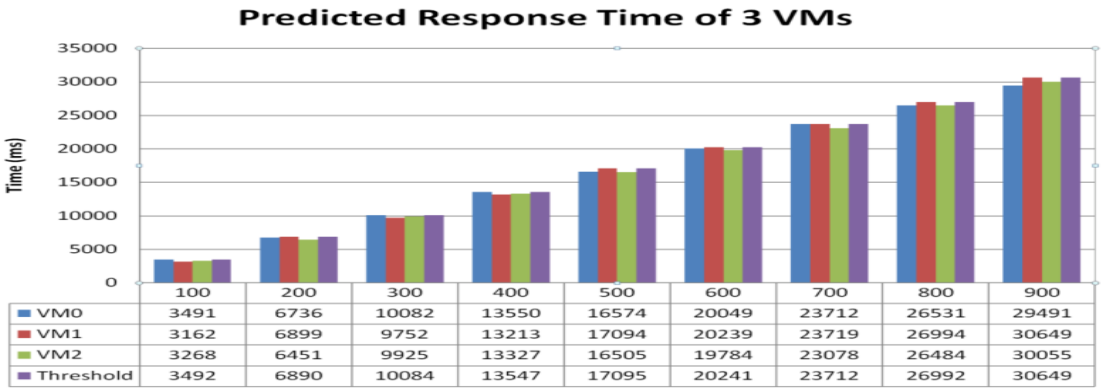
processResourceCharacteristics(SimEvent ev)
createVmsInDatacenter(int datacenterId)
processVmCreate(SimEvent ev)
processCloudletReturn(SimEvent ev)

Experiment the cloud with the above parameters, and run the CloudSim load balancing algorithm available, and run the proposed algorithm, the same input, output comparison, especially the response time parameter. The predicted response time of VM as well as the predicted response time of the cloud with the lower error is the better the algorithm's performance.

In this paper, we temporarily name the proposed algorithm as APRA (*ARIMA Prediction of Response Time Algorithm*).

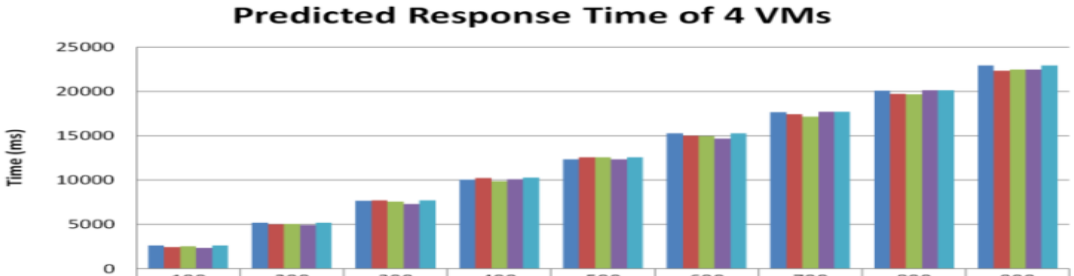
5.2 Simulation results

The results simulation on the CloudSim correspond to case 3, 4, and 5 VMs are built to meet requests, requests are initialized with random length and size, requests are 100, 200, ... to 900. The predicted response time is shown in figure 5, figure 6, and figure 7. With the three cases, we can see the threshold (predicted response time of cloud) is always stable and moving around the response time of all VMs. The predicted response time is quite smooth, it is not the maximum but not also a minimum value. This shows the reasonability of ARIMA in prediction time series data with the near future.



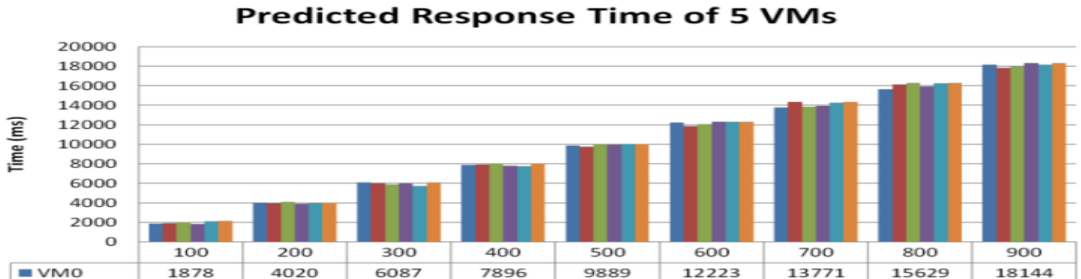
	100	200	300	400	500	600	700	800	900
VM0	3491	6736	10082	13550	16574	20049	23712	26531	29491
VM1	3162	6899	9752	13213	17094	20239	23719	26994	30649
VM2	3268	6451	9925	13327	16505	19784	23078	26484	30055
Threshold	3492	6890	10084	13547	17095	20241	23712	26992	30649

Fig 5. Predicted Response Time of 3 VMs



	100	200	300	400	500	600	700	800	900
VM0	2635	5194	7654	10047	12324	15291	17650	20066	22941
VM1	2443	4986	7700	10251	12564	14984	17411	19700	22334
VM2	2534	5069	7584	9846	12582	14964	17151	19653	22469
VM3	2353	4928	7291	10110	12328	14685	17704	20149	22468
Threshold	2628	5192	7701	10258	12582	15291	17702	20153	22941

Fig 6. Predicted Response Time of 4 VMs



	100	200	300	400	500	600	700	800	900
VM0	1878	4020	6087	7896	9889	12223	13771	15629	18144
VM1	1904	3922	6031	7913	9752	11875	14346	16117	17819
VM2	2042	4097	5874	8045	10019	12080	13852	16300	17967
VM3	1828	3913	5987	7820	9934	12311	13939	15929	18336
VM4	2114	3975	5724	7760	10033	12311	14279	16270	18174
Threshold	2133	4028	6089	8048	10033	12315	14347	16299	18338

Figure 7. Predicted Response Time of 5 VMs

After testing the proposed algorithm (APRA) and observing the predicted response time of the system, we would like to compare the results with the 3 well-known algorithms which represented for the Greedy-balance (Max-Min) and sorted balance (Round-Robin). We test with 5 VMs and the same request input, the results are shown in Figure 8, Figure 9 and Figure 10 with respect to case of 24 requests, 100 requests and 997 requests.

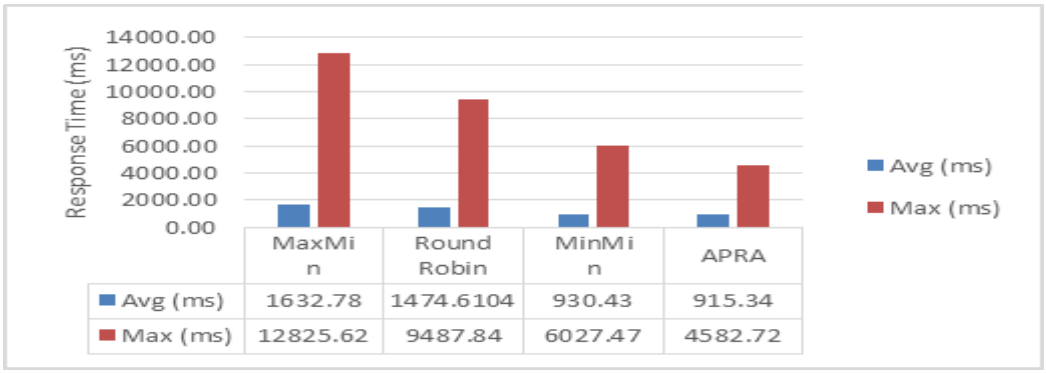


Figure 8. Response Time Comparison with 24 requests

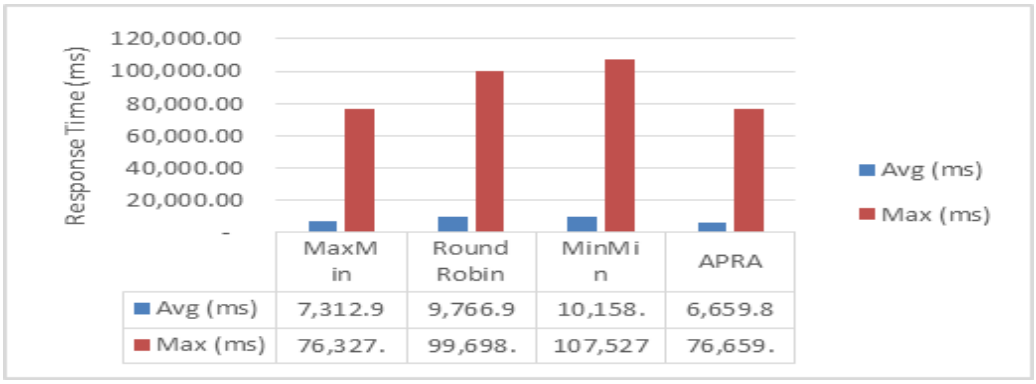


Figure 9. Response Time Comparison with 100 requests

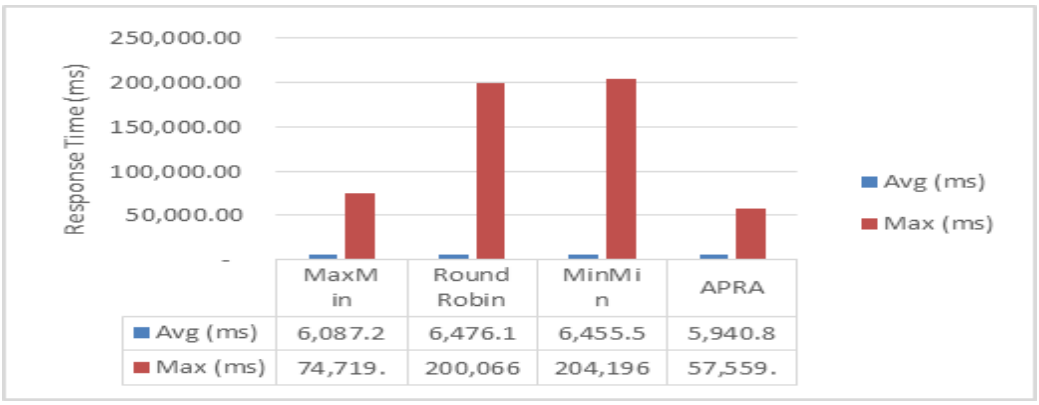


Figure 10. Response Time Comparison with 997 requests

According to the comparison charts above, we can see our proposal APRA is quite stable and potential. Despite we did not take the experiment with many other cases, but we can see the potential and we can develop more.

5.3 Algorithm evaluation

By comparing the predicted response times of virtual machines with the threshold of computation (for the case of 3 VMs, 4 VMs and 5 VMs), we can see that the distribution is quite stable algorithm's, response time is not too different from the cloud's forecast time (ie, threshold).

We can see the low prediction error of the ARIMA algorithm, which makes it to allocate the corresponding requests to VMs in the most effectively. This experiment is just a simulation of a group of virtual machines, not to mention the expansion of the VM pool to reduce the load in case of necessity, assuming the maximum number of virtual hosts request, if it exceeds the newly expanded pool. However, simulation experiments with large requests over 1000 requests require more powerful computers and better processors, so this is a limitation of this simulation experiment.

- **Contributions of the proposed algorithm:**

The ARIMA algorithm predicts the response time of VMs (model 2) not to exceed the threshold (module 1) of the cloud. Therefore, the required distribution capacity of the algorithm is quite stable, the response time is not too different from the forecast of the time cloud (that is, the threshold). We can see the low predictive error of the ARIMA algorithm, making it most efficiently allocate requests to virtual machines.

6. CONCLUSIONS

A new algorithm (APRA) for cloud load balancing using the next response time prediction model has been proposed and experimentally simulated with a small model. The proposed algorithm (APRA) uses an ARIMA algorithm for load balancing based on response times. In particular, the more accurate the predicted response time, the higher the algorithmic efficiency. This algorithm generally approaches and develops the idea of forecasting and time series processing, typically the ARIMA algorithm. Thus, the proposed algorithm has a fairly new approach in load balancing in the cloud environment, while achieving some pretty positive simulation results, showing the good development direction of the algorithm.

The development of the proposed algorithm is a more accurate measurement and calibration of the forecasting time by combining ARIMA with machine learning, unattended or supervised learning by setting peak periods or low point of cloud. The development of better algorithms and deeper, more experimental simulation on the computer is more powerful, large-scale simulation. In addition, setting up algorithms on the actual cloud environment will allow us to research in more depth and detail, since the actual cloud environment will generate issues related to response times, which thus correcting more reasonably and effectively.

REFERENCES

[1] Ghani & Naghmeh Niknejad & Seung Ryul Jeong, (2015), “Energy saving in green cloud computing data centers: a review”, Journal of Theoretical and Applied Information Technology, ISSN: 1992-8645, pages 16-30.

[2] Syed Hamid Hussain Madni, (2016), “Recent advancements in resource allocation techniques for cloud computing environment: a systematic review”, Springer Science+Business Media New York 2016, DOI 10.1007/s10586-016-0684-4.

[3] Ritu Kapur, (2015), “A Workload Balanced Approach for Resource Scheduling in Cloud Computing”, 978-1-4673-7948-9/15/$31.00 ©2015 IEEE.

[4] Agraj Sharma & Sateesh K. Peddoju, (2014), “Response Time Based Load Balancing in Cloud Computing”, 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), pp. 1287-1293.

[5] Atyaf Dhari, Khaldun I. Arif (2017), “An Efcient Load Balancing Scheme for Cloud Computing”, Indian Journal of Science and Technology, Vol 10(11), DOI: 10.17485/ijst/2017/v10i11/110107.

[6] Bharat Khatavkar, Prabadevi Boopathy (2017), “Efficient WMaxMin Static Algorithm For Load Balancing In Cloud Computation”, DOI: 10.1109/IPACT.2017.8245166, International Conference on Innovations in Power and Advanced Computing Technologies [i-PACT2017], Vellore, India.

[7] Jananta Permata Putra, Supeno Mardi Susiki Nugroho, Istas Pratomo (2017), "Live Migration Based on Cloud Computing to Increase Load Balancing", 2017 International Seminar on Intelligent Technology and Its Application, 10.1109/ISITIA.2017.8124096, Publisher: IEEE, 28-29 Aug. Surabaya, Indonesia.

[8] Nguyen Xuan Phi, Cao Trung Tin¬, Luu Nguyen Ky Thu¬ and Tran Cong Hung, "Proposed Load Balancing Algorithm To Reduce Response Time And Processing Time On Cloud Computing", International Journal of Computer Networks & Communications (IJCNC) Vol.10, No.3, DOI : 10.5121/ijcnc.2018.10307,pp 87-98, May 2018.

[9] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov ,C´ esar A. F. De Rose and Rajkumar Buyya (2010), "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Software: Practice and Experience (SPE), Volume 41 Number 1, pp.23-50.

[10] Rashmi. K. S, Suma. V, Vaidehi. M (June 2012), "Enhanced Load Balancing Approach to Avoid Deadlocks in Cloud", Special Issue of International Journal of Computer Applications on Advanced Computing and Communication Technologies for HPC Applications – ACCTHPCA, pp.31-35.

[11] Rajwinder Kaur, Pawan Luthra (2014), "Load Balancing in Cloud System using Max Min and Min Min Algorithm", International Journal of Computer Applications Proceedings on National Conference on Emerging Trends in Computer Technology (NCETCT- Number 1), pp.31-34.

[12] Navtej Singh Ghumman, Rajwinder Kaur (2015),"Dynamic Combination of Improved Max-Min and Ant Colony Algorithm for Load Balancing in Cloud System", 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT).

[13] Hafiz Jabr Younis (2015), "Efficient Load Balancing Algorithm in Cloud Computing", Islamic University Gaza Deanery of Post Graduate Studies Faculty Of Information Technology

[14] Shubham Sidana, Neha Tiwari (2016), "NBST Algorithm: A load balancing algorithm in cloud computing", International Conference on Computing, Communication and Automation (ICCCA), pp. 1178 – 1181, IEEE Conference Publications.

[15] Agraj Sharma, Sateesh K. Peddoju (2014), "Response Time Based Load Balancing in Cloud Computing", 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), DOI: 10.1109/ICCICCT.2014.6993159, IEEE, Kanyakumari, India

[16] Xiaoming Nan, Yifeng He and Ling Guan (2013), "Optimization of Workload Scheduling for Multimedia Cloud Computing, IEEE International Symposium on Circuits and Systems (ISCAS2013), DOI: 10.1109/ISCAS.2013.6572478

[17] S. Boyd and L. Vandenberghe (2004), "Convex optimization". Cambridge University Press.

[18] Bharat Khatavkar, Prabadevi Boopathy (2017), "Efficient WMaxMin Static Algorithm For Load Balancing In Cloud Computation", International Conference on Innovations in Power and Advanced Computing Technologies [i-PACT2017], DOI: 10.1109/IPACT.2017.8245166, IEEE, Vellore, India.

[19] Louai Sheikhani,Yaohui Chang, Chunhua Gu and Fei Luo (2017), "Modifying Broker Policy for Better Response Time in Datacenters", 2017 3rd IEEE International Conference on Computer and Communications (ICCC), DOI: 10.1109/CompComm.2017.8322977, Chengdu, China.

[20] Kripa Sekaran, K R Kosala Devi (2017), "SIQ Algorithm for Efficient Load Balancing In Cloud", International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), IEEE, DOI: 10.1109/ICAMMAET.2017.8186673.

[21] Jon Kleinberg, Eva Tardos (2006), "Algorithm Design", Conell University, Copyright 2006 by Pearsion Education Inc, ISBN 0-321-29535-8, Publisher: Addison-Wesley.

[22] Sambit Kumar Mishra, Bibhudatta Sahoo, Priti Paramita Parida (2018), "Load Balancing in Cloud Computing: A big Picture", Journal of King Saud University - Computer and Information Sciences, January 17.

[23] Ross Ihaka. Time Series Analysis, Lecture Notes for 475.726, Statistics Department, University of Auckland, 2005.

[24] Sidra Aslam , Munam Ali Shah (2015), "Load Balancing Algorithms in Cloud Computing: A Survey of Modern Techniques", 2015 National Software Engineering Conference (NSEC 2015) on 17-17 Dec. 2015, DOI: 10.1109/NSEC.2015.7396341, IEEE 2015

[25] Liang-Teh Lee, Hung-Yuan Chang, Gei-Ming Chang and Hsing-Lu Chen, "A Grey Prediction Based Load Balancing Mechanism for Distributed Computing Systems", 2006. ISCIT '06. International Symposium on Communications and Information Technologies, 2006.

[26] Jay W.Y. Lim, Poo Kuan Hoong, Eng-Thiam Yeoh,"Neighbor's Load Prediction for Dynamic Load Balancing in a Distributed Computational Environment". TENCON 2012 - 2012 IEEE Region 10 Conference, page 1 – 6, 19-22 Nov. 2012, Cebu, Philippines.
[27] Iniya Nehru, Venkatalakshmi.B, Ranjithflalakrishnan, Nithya.R,"Neural Load Prediction Technipue for Power Optimization in Cloud Management System", Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013).
[28] Yingchi Mao, Daoning Ren, Xi Chen, "Adaptive Load Balancing Algorithm Based on Prediction Model in Cloud Computing", ICCC'13, Decem ber 1– 2, 2013, Wuhan, China
[29] Foram Kherani, Mr. Jignesh Vania (2014),"Dynamic Future Prediction Load Balancing Algorithm For Virtual Machine Instances In Cloud", IJIRT 100026 Internationnal Journal of Innovative Research in Technology 55, ©2014 IJIRT | Volume 1 Issue 1 | ISSN: 2349-6002.
[30] Sambit Kumar Mishra, Bibhudatta Sahoo, Priti Paramita Parida (2018), "Load balancing in cloud computing: A big picture", Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2018.01.003.
[31] Roy Batchelor. Box-Jenkins Analysis. Cass Business School, City of Lodon.
[32] Husam Suleiman and Otman Basir (2019), "QoS-Driven Job Scheduling: Multi-Tier Dependency Considerations", Distributed, Parallel, and Cluster Computing (cs.DC), pp.133-155, Volume 9, DOI: 10.5121/csit.2019.90912
[33] Husam Suleiman and Otman Basir (2019), "Service Level Driven Job Scheduling in Multi-Tier Cloud Computing: A Biologically Inspired Approach", Distributed, Parallel, and Cluster Computing (cs.DC), pp.99-118, Volume 9, DOI: 10.5121/csit.2019.90910

AUTHORS

Nguyen Xuan Phi was born in Vietnam in 1980. He received Master in Posts & Telecommunications Institute of Technology in Ho Chi Minh, Vietnam, 2012, major in Networking and Data Transmission. Currently, he is a PhD candidate in Information System from Post & Telecommunications Institute of Technology, Vietnam. He is working at the Information Technology Center of AGRIBANK in Ho Chi Minh city, Vietnam. His main research areas are load balancing on cloud computing, optimizing the performance of cloud computing.

Le Ngoc Hieu has been working in IT industry as a IT System Architect since 2010. In 2018, He completed Master Degree at Post & Telecommunication Institute of Technology. As now, he is working as an IT lecturer for HCMC Open University. His major study is about cloud computing and cloud efficiency for better service; his minor study is about education, especially education in IT line .

Tran Cong Hung was born in Vietnam in 1961. He received the B.E in electronic and Telecommunication engineering with first class honors from HOCHIMINH University of technology in Vietnam, 1987. He received the B.E in informatics and computer engineering from HOCHIMINH University of technology in Vietnam, 1995. He received the master of engineering degree in telecommunications engineering course from postgraduate department Hanoi University of technology in Vietnam, 1998. He received Ph.D at Hanoi University of technology in Vietnam, 2004. His main research areas are B – ISDN performance parameters and measuring methods, QoS in high speed networks, MPLS. He is, currently, Associate Professor PhD. of Faculty of Information Technology II, Posts and Telecoms Institute of Technology in HOCHIMINH, Vietnam.