A Proposed Load Balancer Using Naïve Bayes to Enhance Response Time on Cloud Computing

Tran Cong Hung*, Tran Duy Tien**, Le Ngoc Hieu***

*Faculty of Information Technology, The Saigon International University, Ho Chi Minh City, Vietnam

**Saigon University, Ho Chi Minh City, Vietnam

***Faculty of Information Technology, Ho Chi Minh City Open University, Ho Chi Minh City, Vietnam

tranconghung@siu.edu.vn, tranduytien.tranduytien@gmail.com, hieu.ln@ou.edu.vn

Abstract— **Cloud computing is a modern technology that brings challenges to all organizations worldwide. Response time is one of the most important challenges that cloud computing are facing. Improving response times for user requests is a big issue that can help reduce bottlenecks in cloud. With rapid growth as the number of applications grows, leading to the number of accessing the cloud is also rapidly increasing, easily cause the overload. This paper proposes an improvement using application of Naïve Bayes algorithm (based on Bayes theorem on probability theory to make judgments as well as classify data based on observed and statistical data) to improve the response time of VMs in the cloud, enhance the performance of cloud and cloud infrastructure. This new proposal is named as RCBA (Response Time Classification with Naive Bayes Algorithm). We simulated this proposed RCBA algorithm with the CloudSim engine and the result has improved from 4 popular load balancing algorithms: Round-Robin, FCFS, MaxMin and MinMin.**

Keywords— **Load balancing, response time optimization, cloud computing**

I. Introduction

Cloud Computing [1],[2],[3] is the trend of developing computer networks, inheriting previous networks and distributed computing concepts to integrate the existing network resources and provide the basement network for computers, storage, platform, and other services. These are based on demand of convenience and quickness, means that providers allow users to terminate their service, easily release their resources. With this approach, it can minimize interactions with providers and users only pay for the resources actually used (pay-by-use).

Cloud computing allows applications to be less dependent on network infrastructure, saving users money by not investing more hardware and infrastructure. All data will be uploaded onto the cloud, users can access and use it anywhere, anytime. On that basis, cloud computing technology has appeared and has been increasingly developing for commercial using, scientific research. Besides, the multimedia [4] has created a growing demand for high-performance computing infrastructures, the issues of exchanging and processing information, data security and especially load balancing in cloud computing. Among them, load balancing is one of the most challenges on cloud computing, which is a wide set of interests and focuses for researchers and cloud service providers.

In addition, quality of service (QoS) standard is also paid attention to, which is the basis for ensuring quality of cloud computing. According to the article [5], the QoS of cloud computing is mainly due to the allocation of resources for the applications running on it. To ensure the quality of service provided to customers, we must consider the space needs as well as in terms of response performance, availability, and reliability. According to the document [5], QoS includes the following models: Workload model, System model and applications of QoS model. From here, we can see that load balancing is one of the important factors in ensuring quality of service on Cloud Computing and it is one of the research directions to help Cloud Computing perform better and more developed.

For few years ago [6], if we store on DVD the amount of data transferred on the global network, the number of discs in line would be a few times the distance from earth to the moon. It is estimated that this amount of storage data will increase 44 times by 2020. The growth of data traffic with more than 5 billion people using mobile devices is one of the key factors driving the rapid growth of cloud computing. With the current speed of growth, we find the efficiency of cloud computing so overwhelming. To solve this problem, we need to combine several different approaches to load balancing on the cloud data centre. One of the methods is to reduce the response time of cloud services when users request to access their services. Load balancing aims to find best strategies to save computing resources and increase user service, directly impacting service provider's business to increase profitability.

With the purpose of find a strategy for load balancing on cloud computing with enhancing response time, this article is organized with 5 sections as follows: section 1 is the introduction of load balancing on cloud computing; Section 2 is about the related works including surveys, reviews from some recently publication research works on load balancing; section 3 is the proposed load balancing algorithm (RCBA); section 4 discusses about the simulation and results of the proposed algorithm; and finally section 5 is our conclusion.

II. Related Work

In 2017, Atyaf Dhari and Khaldun I.Arif [7] from India proposed the Load Balancing Decision Algorithm (LBDA) to manage and balance the load among VMs in the data centre, reducing the time of tasks / jobs completion (Makespan) and time of responding. In cloud environment, there are many data centres consisting of heterogeneous resources such as physical servers and virtual machines (VMs). Physical servers and VMs can have different configurations such as memory size, bandwidth, storage, and processing power. The authors had set up a set of virtual machines as VM {VM1, VM2, VM3, …, VMm}, Each VM has different parameters like VM state and processing speed in million instructions per second (MIPS - Microprocessor without Interlocked Pipeline Stages). All virtual machines are uninterrupted and independent operation. The coming tasks are independent, they are {T1, T2, T3, …, Tn}. Each task / job has different properties like id, length, start time and end time. The authors compared the proposed algorithm with Round Robin algorithm, Max-Min algorithm and SJF (Shortest Job First) algorithm in the same configuration environment. The simulation results show that their proposed algorithm performs well in all the cases compared to the Max-Min, SJF (Shortest Job First) and Round Robin algorithms by reducing the average response time, average and total execution time of all VMs.

In 2017, The article [8] by Louai Sheikhani, Yaohui Chang, Chunhua Gu, Fei Luo from China, proposed an algorithm, Modifying Broker Policy for Better Response Time in Datacentres. This new algorithm is to modify broker policy for better response time in data centre. In a multi-service cloud environment, the service broker controls and routes traffic between the user request and the data centre based on different service broker policies. Based on proximity service policies, it routes the request to the nearest data centres to handle. If there is more than one data centre in the same area, it randomly selects without considering any criteria such as workload, cost, response time or other parameters. A random selection of data centre is easily unsatisfactory to the customer. In this work, the authors propose to change the policy and apply a new scheduling algorithm that can improve response time, control load balancing. This article is based on cloud analysis, verified results suggest our solution is effective and can reduce response time and average response time.

In the same year 2017, the article [9] by S. K. Mishra et al., presents a new load-balancing approach to organize efficiently data centre and virtualization resources. In this study, the load on the virtual machine (VM) increases and decreases with the resource capacity of the VM. The proposed scheme (DLBA - Dynamic Load Balancing Algorithm) minimizes the system cost, maximizes the use of resources, and reduces the overall energy consumption. The authors experimented the approach in a CloudSim simulation environment, and it reduced waiting times compared to existing methods and optimized the resources and the costs of data centre. The authors compared to RoundRobin and FCFS, the overall makespan and waiting time are much better in the proposed DLBA.

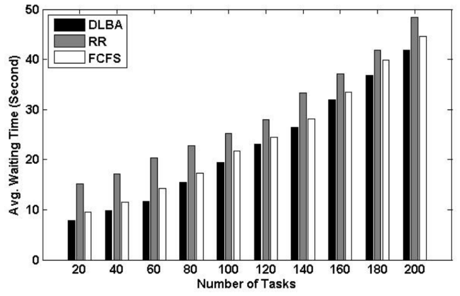

Figure 1. Comparison of average waiting time of RoundRobin, FCFS and DLBA [9]

In 2019, the article [10] by a group of authors Maria Maqsood, Saima Anwar Lashari, Murtaja Ali Saare, Sari Ali Sari, Yaqdhan Mahmood Hussein, Hatem Oday Hatem *"Minimization response time task scheduling algorithm"*, propose a task scheduling algorithm to minimize response time in a distributed computing system using a multi-agent cloud. In this paper, the authors focus on the distributed model of multi-layer system architecture. This leads to a situation where it is prudent to allocate virtual machines to the host machine to improve resource allocation efficiency. The focus for this is on reducing allocation time to optimize response time when using virtual machine locations and using an algorithm that allocates virtual machine accounts for the types of resources available between machine centres. owner. The algorithm is implemented using a cloud simulation tool, through this tool, the calculation results of the algorithm are performed better than that of the traditional algorithm, and the difference is presented by graphs for both algorithms. The results support the proposed methodology.

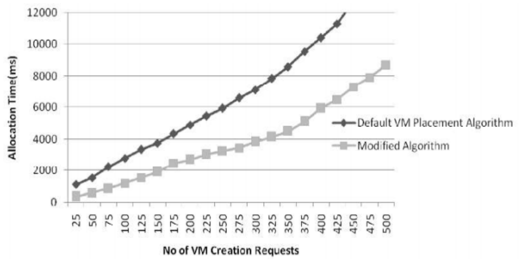

Figure 2. The time allocation of the default algorithms and the tuning algorithm [10]

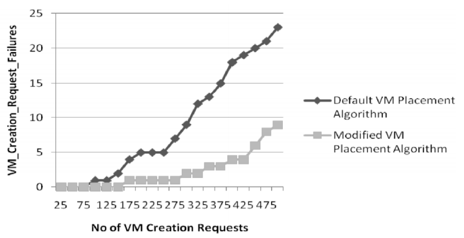

Figure 3. Ratios of VM with Request Failures [10]

In 2020, the article [11] by Nguyen Xuan Phi et al., have proposed a load balancing algorithm to reduce the response time in cloud computing. The main idea is based on the ARIMA prediction algorithm to predict the response time, thereby providing an efficient resource allocation solution based on the time threshold. To improve the performance of cloud computing, there are many parameters by which we can consider and allocate resources, meet resources, connect unused resources, and plan resource usage based on multiple parameters and response time is one of those. Users can easily figure out their response time requirements and it becomes one of the important QoSs requirements. As we learn and discover more about response times it can provide solutions for load-balanced distribution of resources with better efficiency. This is one of the most promising directions of research to improve cloud computing technology. Therefore, this paper proposes a load balancing algorithm based on the response time of cloud requests with the name APRA (Response Time Arima Prediction), the gist is to use Arima algorithm to predict upcoming response time, for a better way to efficiently deal with resource allocation with threshold value. The results of load balancing value experiment with predicted response time are very promising, it shows that predictive response time is an excellent direction for load balancing.

In the same year of 2020, Hieu Le Ngoc et al. [12] proposed MCCVA, an approach using SVM and Kmeans for load balancing on cloud computing. In this paper, the authors stated that the existing methods of load balancing which are currently using, are more about natural and heuristic methodology, just a few of applying the AI or Machine Learning technologies in load balancing due to the specific characteristics of cloud. They focus to reduce the processing time (makespan) on cloud computing, helping the load balancing work more efficiency. They used SVM to classify the coming Requests, K - Mean to cluster the VMs of the datacentre. The proposed LB will allocate the requests into the VMs in the most reasonable way. In this way, request with the least processing time will be allocated to the VMs with the lowest usage. The authors also have experimented and evaluated the algorithm in CloudSim. They also compared their work with the popular and well-known algorithms: Round Robin, MaxMin, MinMin and FCFS. The results showed that MCCVA has improved for their tests.

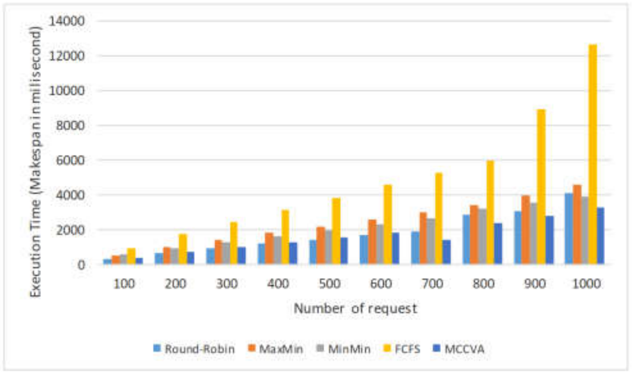

Figure 4. the execution time of 5 algorithms with 1000 Requests [12]

Sarita Negi, et al. [13] in 2020 also public their study to the world, it is CMODLB, an efficient load balancing approach in cloud computing environment. In this study, CMODLB is a hybrid of machine learning techniques (Supervised – Artificial neural network, unsupervised - clustering) and soft computing like interval type 2 fuzzy logic. They use the ANN-LB (artificial neural network) to cluster the VMs into underloaded and overloaded. In the scheduling stage, the proposal handles the underloaded tasks by multi-objective-based technique of order preference, it is like swarm optimization, they named it as TOP-SIS-PSO. In the research model, they add the VM manager to migrate the VM. With this migration, the interval type 2 fuzzy logic system (IT2FS) is used for make decisions. Everything of cloud is managed by the PMs (Pool Manager). They have conducted the application with their tuning parameters, and they selected the most significant params for their algorithms. To test and evaluate the proposed algorithms, the authors using CloudSim tool to carry on. Experimental results show that the CMODLB method takes 31.067% and 71.6% less completion time than TaPRA and BSO, respectively. It has maintained 65.54% and 68.26% less MakeSpan than MaxMin and R.R algorithms, respectively. The proposed method has achieved around 75% of resource utilization, which is highest compared to DHCI and CESCC. With the approach of this paper, we can see they have developed a fully model and take the most advantages of Machine learning techniques to generate their best fit of Load Balancing. This is one of the great paper that they know how to take the advantages of AI and ML in LB.

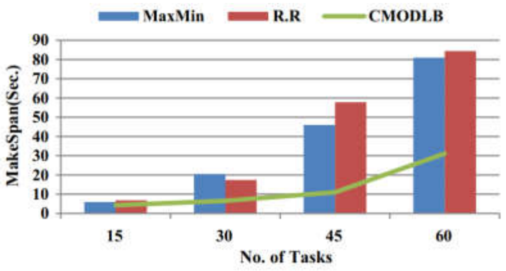

Figure 5. performance analysis for Test I, compare to MaxMin and RR [13]

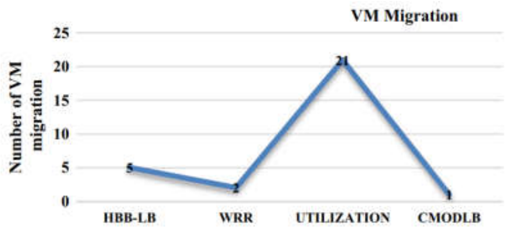

Figure 6. Comparation of number of migrations [13]

In 2021, T. J. B. Durga Devi et al. [14] proposed an application of neuro fuzzy inference system in load balancing. In this article, the authors also mention about the security of VM in cloud environment. NP-hard optimization problem corresponds to load balancing. They follow the Forbes about the introduction of General Data Protection Regulation, security in cloud continue to be an issue, the existing system uses a fuzzy based hybrid LB, but this is not satisfied. The authors focus on opportunities for improving CPU utilization and turnaround time and in terms of security, they proposed their work, MANFIS (Modified Adaptive Neuro Fuzzy Inference System). Parameters of MANFIS are optimized by introducing Fire-fly Algorithm. Security is imposed on user authentication by using the Enhanced Elliptic Curve Cryptography. This is a password-less mechanism to authenticate users. With the results, the authors tell us the improvement and the satisfactory of security on cloud, they show better performance with respect to resource utilization, cost and execution time, when compared with existing system, as shown by results of experimentation.

In 2021, an overview [15] by Dalia Abdulkareem Shafiq et al., the review study shows that Load Balancing is an important aspect on Cloud Computing environment, help to enhance the workload distribution and utilize the resources efficiently, especially improve the response time for cloud users. The paper tells us that there are a lot of issues related to LB, they are scheduling of tasks, migration, resource utilization, and so on. The authors survey and analyse the past six years research and studies on Load Balancing. This review also shows us the potential of intelligent approaches such as Artificial Intelligence, Machine Learning for LB on cloud. This study will be helpful for researchers to identify research problems related to load balancing, especially to further reduce the response time and avoid failures in the server. Another advantage of this study for our proposal, it is about the simulation tools and experimental environments. They also show that, CloudSim and CloudAnalyst are the best using for this field of study, it is the benefit of these tools.

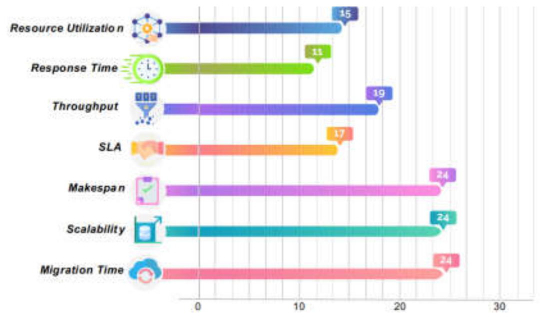

Figure 7. Evaluation Metrics of Reviewed Articles. [15]

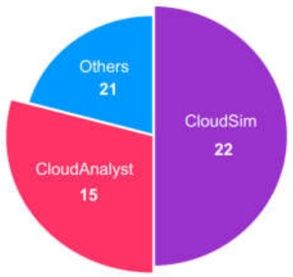

Figure 8. Simulation Tools of Reviewed Articles [15]

III. Proposal

In this article, we would like to propose an algorithm to reduce response time on the cloud, by improving response time for users (userbase) and data centre processing times. For Response Time Classification, we use Naive Bayes Algorithm, and for Virtual Machines Clustering we use K-Means. This new algorithm is named as RCBA (Response Time Classification with Naive Bayes Algorithm). The aim is to improve the response time by modifying the LB on cloud (cloud computing), and the application on the cloud environment (cloud in real time) by the scheduling policy used is Time-Share for virtual machines and tasks. The objective is to use the Naive Bayes classifier scheduling algorithm combined with K-means clustering to improve the response time of the load balancing system in the cloud environment.

A. *Proposed Algorithm:* RCBA

RCBA is a load balancing algorithm to reduce response time on cloud computing, it is based on available data on the time series of response time of requests from customers (client requests) and some other properties of the requests. We use Naive Bayes algorithm combined with K-means to predict the next response time (using NB classifier), thereby knowing how to allocate the most appropriate resources for the coming requests. The proposed algorithm consists of 3 main modules.

Module 1: Classify the coming requests using Naive Bayes algorithm. In this Module, Naive Bayes algorithm will rely on the properties (attributes) of requests from clients, it calculates the response time based on the historical data of responded requests, Naive Bayes will classify these coming requests. The attributes of the request are Response length, Max length, Size, etc.

$$\mathtt{RT_{new} = NB(X_1, X_2, X_3, …, X_n)}$$

Where X_i is the attributes of historical requests which have been sent to the cloud, and well handled by the VMs of cloud. RT_{new} stands for the prediction response time for the coming request.

From this perspective, we can divide into many groups of coming requests based on prediction RT (it can be 3 to 8 groups) or more depending on the variant of the requests.

Module 2: Cluster the virtual machines/servers/resources. This module will use K-Means clustering algorithm (with k = 3) to cluster virtual machines based on activity level, resource usage of virtual machines, including clusters such as: low, medium, high.

$$\mathtt{Cluster_i = Kmeans(RAM, Usage, CPU,…)}$$

Where: i = 1 is for the low group
i = 2 is for the mean group
i = 3 is for the high group

Module 3: Allocate virtual machine services. This module is responsible for allocating requests to the right virtual machines through the appropriate conditions defined by module 1 and module 2. If a request is sent, the request is classified by module 1, and the virtual servers in question even if this virtual server is not loaded are also clustered by module 2. Then the algorithm will calculate to find out which request is most suitable for which virtual machine through the return parameters of Naive Bayes and K-Means functions above. If the response time of the request (calculated from module 1) is the smallest, then this request will be processed on the virtual hosts with the furthest means (i.e., belonging to group 1, and having lowest usage). For requests that are not small or large, we can use calculation methods such as analogy or difference to calculate the allocation.

The module that allocates virtual machines is shown as follows

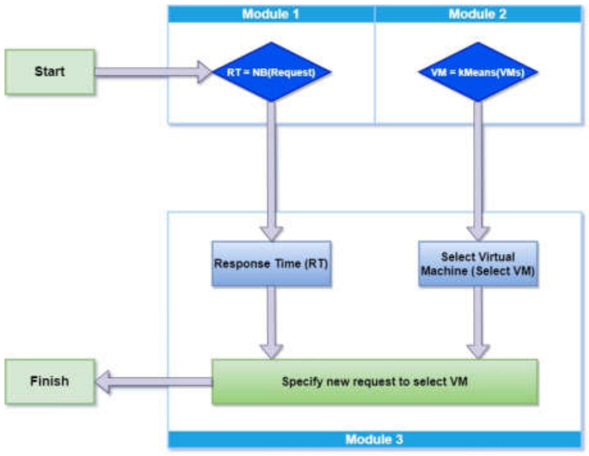

Figure 9. The schematic diagram of the RCBA algorithm

B. Pseudocode of RCBA

RCBA - Response Time Classification with Naive Bayes Algorithm

```
1.  For each Request in CloudRequests
2.      isLocated = false;
3.      RT_new = NB(RT1, RT2…..);  // Module 1
4.      For each VM in VMList
5.          VM_Cluster = kMeans(situation); //situation of VM, Module 2
6.          If isFitSituation(RT_new , VM_Cluster)
7.              AllocateRequestToVM(VM_Cluster, Request); // Module 3
8.              isLocated = true;
1.              break;
2.          End If
3.      End For
4.      If (!isLocated)
5.          VM = VMList.getMinFromMean(); // Module 2
6.          AllocateRequestToVM(VM, Request);
7.      End If
8.  End For
```

In the pseudocode above, the RCBA algorithm will use a loop to listen to all the *Requests* in the queue list of *Requests* sent to the load balancer (in this case, CloudRequests). Once this list is exhausted, it will no longer be distributed. In it, the algorithm uses the *isLocated* variable (logical type) to flag that the Request whether has been allocated or not. The first jump of the loop, the *isLocated* variable is set to false. Then, the algorithm calculates the new Response Time (predicted response time using Naïve Bayes), *RT_new* variable is to perform the Request in the current situation. This calculation is based on the historical data of previous requests *RT1, RT2, ... RTn* where *n* is the number of requests that have been saved in the LB. Corresponding to each machine, we use the K-Means to cluster the situation of that VM, we get the *VM_Cluster* variable. The algorithm considers whether the virtual machine matches the predicted RT or not, through the *isFitSituation(RT_new , VM_Cluster)* function. If it is satisfied, it will allocate the request under consideration to that virtual machine *AllocateRequestToVM(VM_Cluster, Request)*, and at the same time assign the variable *isLoacated = true*. If no matching virtual machine is found, the loop ends. At this point, the *isLocated* variable is still false, and now the *Request* has not been allocated. Therefore, the algorithm allocates this Request to the first VM which gets the nearest means, *VM = VMList.getMinFromMean()*. This allocation ensures that if any requests are predicted that are not in the data of the algorithm, they are still allocated and processed for the user.

The historical data is always update after the completion of a request processing. We limit a number of requests depending on the requirement and the characteristic of the cloud users.

IV. Simulation Results

In this article, we use the CloudSim library (version 4.0) and programmed in JAVA language with Eclipse IDE tool. The cloud simulation environment is from 5 to 15 virtual machines, and it creates a random request to the cloud services, including CloudSim's virtual machine provisioning, provisioning, and user-response service for testing.

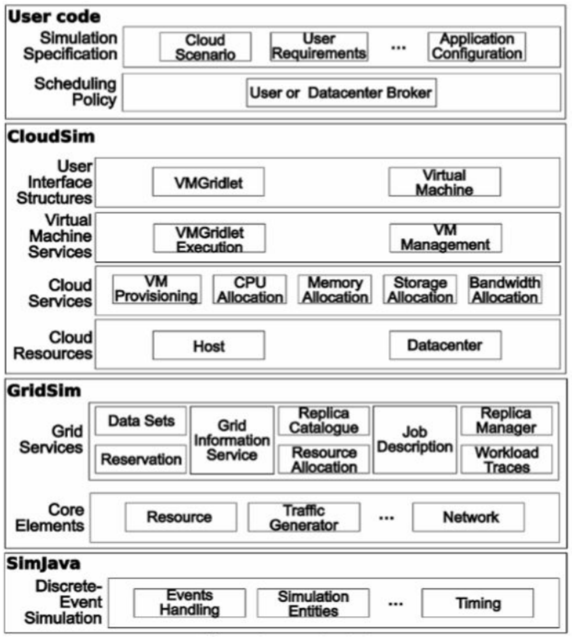

Figure 10. CloudSim Layer Architecture (cloudbus.org)

A. CloudSim Configuration

We run simulation experiments on CloudSim with 5 pre-built virtual machines to respond and serve the requests. Requests are initialized with random length and size, number of requests are from 20-50. To evaluate and to compare, we run these same requests with 04 algorithms Round-Robin, MaxMin, MinMin and FCFS.

TABLE 1. DATACENTER CONFIGURATION PARAMETERS

Datacentre information	Host information in Datacenter
- Number of hosts (hosts) in datacenter: 5 - Do not use Storage (SAN drives) - Architecture (arch): x86 - Operating system (OS): Linux - Processing (VMM): Xen - TimeZone: +7 GMT - Cost: 3.0 - Cost per Memory: 0.05 - Cost per Storage: 0.1 - Cost per Bandwidth: 0.1	Each host in the Datacenter has the following configuration: - CPU has 4 cores, each core has a processing speed of 1000 (mips) - RAM: 16384 (MB) - Storage: 1000000 - Bandwidth: 10000

Table 1 is the configuration of the datacenter that we use to simulate our proposal. We use these numbers based on the current physical datacenters we used to work with.

TABLE 2. VIRTUAL MACHINE CONFIGURATION

Size	100000 MB
RAM	1024 MB
MIPS	500
Bandwidth	1000
Number of CPUs (PES No.)	1
VMM	Xen

Table 2 is about the Virtual Machine configuration. We configure the VM with the above settings due to the request and the processing resources of the request respectively. In our research model, we just limit the request not too big and too complicated, just need 1 CPU (core) to handle, and the RAM usage is also not an issue.

TABLE 3. REQUEST PARAMETERS CONFIGURATION

Length	1700 ~ 3000
File Size	5000 ~ 45000
Output Size	450 ~ 750
Number of CPUs to process (PEs)	1

Table 3 is our requests settings. The Requests (mostly like web requests). We simulate the short and small requests as normal users, not special user like IT specialists of other users. These requests are represented by the *Cloudlet* in CloudSim and the size of the Cloudlets are initialized randomly using JAVA's random function. The number of Cloudlets is to 20 to 1000 respectively.

The proposed algorithm is built by creating the *RCBASchedulingAlgorithm* class, inheriting from the *BaseSchedulingAlgorithm* object (original in CloudSim), updating some methods and properties related to *predictRequestNB*, and adjusting the built-in functions to fit the proposed algorithm output. The adjustment are made as following:

```
// Module 1
public String predictRequestNB(Cloudlet req)
// Module 2
public CondorVM getMostFreeVM(String vmClass)
// Module 3
@Override
public void run()
```

B. Experiment and simulation results

The results of running simulation experiments on CloudSim with 5 pre-built virtual machines to meet the requirements, requests are initialized with random length and size, the number of Requests is from 20-1000 and compared with other load balancing algorithms. They are Round-Robin, MaxMin, MinMin and FCFS algorithms. We focus on the response time, and consider it as a key criteria to evaluate and compare between algorithms.

To understand more detail, we simulated with 4 circumstances. These 4 cases are different by the number of requests. They are 25, 50, 100 and 1000 requests respectively. With 4 cases, we store the historical data of last 100 requests which has been served by the cloud. These data are the data for classifying the next coming requests. For initialize, we already built a dataset of 100 requests, and it is updating while a request finished.

TABLE 4. EXPERIMENTAL RESULTS RESPONSE TIME WITH 25 REQUESTS

Algorithm	Number of requests				
	5	10	15	20	25
Round Robin	17.42	17.64	2.89	31.9	1.81
MaxMin	11.71	16.76	0.79	22.39	0.49
MinMin	11.37	20.54	0.75	114.6	0.48
FCFS	14.27	15.06	0.99	27.41	0.62
RCBA	16.04	12.23	0.83	12.72	0.53

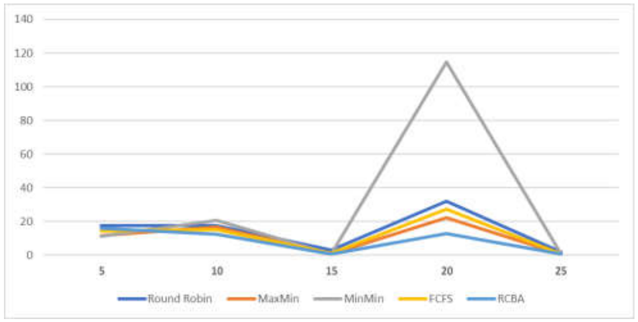

Figure 11. Response time of 5 algorithms with 25 Requests

Figure 11 shows the results of the 1st case, we run with 25 requests. The response time of RCBA is the lowest among the 25 requests, despite the debut is a bit higher. In this case, we can easily see that the 20th request may be complicated and bigger than other requests, causing that all algorithms' response time longer. But RCBA is the best in this processing.

TABLE 5. EXPERIMENTAL RESULTS RESPONSE TIME WITH 50 REQUESTS

Algorithm	Number of requests									
	5	10	15	20	25	30	35	40	45	50
Round Robin	23.13	18.39	18.04	18.09	18.17	18.33	18.17	18.35	18.14	20.08
MaxMin	14.84	73.43	14.1	14.75	13.84	40.79	39.76	14.96	57.43	1.6
MinMin	16.64	50.82	21.53	11.02	22.29	10.91	34.69	16.35	39.75	1.26
FCFS	20.55	28.14	29.04	10.91	11.05	10.92	28.63	16.22	32.5	14.16
RCBA	17.86	12.44	10.91	14.1	12.06	14.07	20.47	10.68	20.08	8.01

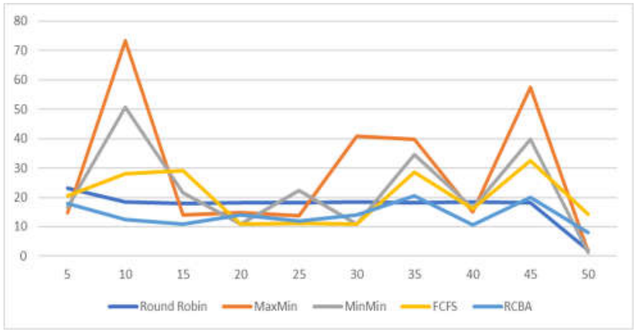

Figure 12. Response time of 5 algorithms with 50 Requests

In figure 12, we can see the 50 requests are not same and equivalent to each other. These 50 requests are different, leads to the different handling of VMs. The overall perspective shows that RCBA is the best one among them, RCBA keeps the response time lowest among the 5 algorithms, except some requests (no 20, 25 and 50). RCBA shows its stability and the suit for various changing of requests. Besides, we see that Round-Robin algorithm is dominant and fast processing, FCFS algorithm is also quite stable due to the small number of requests. MaxMin algorithm is does not perform well in this case.

TABLE 6. EXPERIMENTAL RESULTS RESPONSE TIME WITH 100 REQUESTS

Algorithm	Number of requests									
	10	20	30	40	50	60	70	80	90	100
Round Robin	14.36	35.49	13.84	16.04	14.69	15.9	37.21	18.7	11.69	2.91
MaxMin	50.07	39.25	38.8	38.59	38.56	38.99	38.88	19.51	39.68	3.03
MinMin	14.53	14.76	14.62	12.67	15.28	16.79	11.15	12.75	50.02	14.85
FCFS	17.5	33.1	13.46	32.85	29.76	13.83	40.57	8.86	13.73	1.37
RCBA	14.4	11.58	11.42	26.6	14.5	11.3	11.22	5.68	11.51	0.88

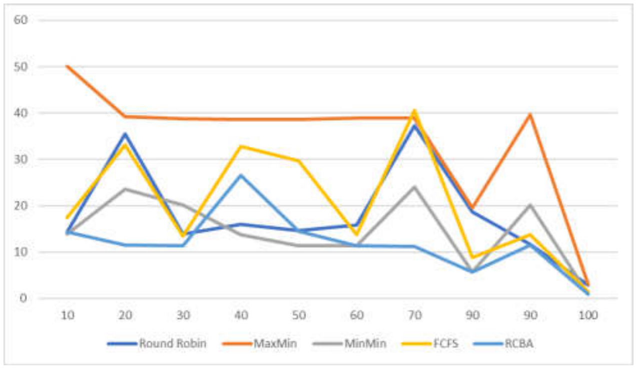

Figure 13. Response time of 5 algorithms with 100 Requests

To test with more request, the 3rd case is 100 requests. Figure 13 shows that RCBA is still the most stable algorithm among the 5 ones. RCBA is only not good at the 40th and 50th requests, but it is just a bit smaller than the smallest. In this case, among the 50 request, the variant are much more, the graph shows like the zigzag image of all algorithms, this kind of requests fits the reality of cloud users. In this case, due to the small number of requests, we still see MaxMin the worst one. MinMin shows better than the 2nd case, and RoundRobin is still the stable one besides RCBA. To see how stable the proposed RCBA is, we continue with the 4th case that we use 1000 requests.

TABLE 7. EXPERIMENTAL RESULTS RESPONSE TIME WITH 1000 REQUESTS

Algorithm	Number of requests									
	100	200	300	400	500	600	700	800	900	1000
Round Robin	14.53	14.76	14.62	12.67	15.28	16.79	11.15	12.75	50.02	14.85
MaxMin	24.25	16.41	19.17	16.19	13.38	18.94	13.27	13.16	13.63	3.18
MinMin	247.38	194.12	195.41	27.11	27.08	27.21	27.49	11.1	11.17	2.63
FCFS	31.07	11.1	11.1	38.57	24.57	28.32	28.09	28.03	40.34	6.77
RCBA	48.96	39.05	38.18	38.84	38.15	38.62	38.91	39.09	40.44	9.16

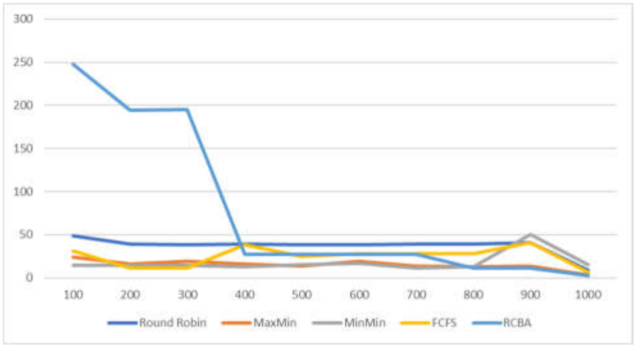

Figure 14. Response time of 5 algorithms with 1000 Requests

Figure 4 stands for the response time of 5 algorithms with the 4th case. In this case, we can se from the 1st request to the 400th request, RCBA is far away from the rests. But from the 400th onward, RCBA is stable and show its effectiveness. It slowly moves down and reach the smallest from 800th onward. This can tell us that RCBA will be stable for long term and much more request coming. We can see why it dose not fit the beginning due to the historical data storing. Very fast, RCBA can learn the trend, and it adjust for the best fit based on the historical data. In this case, we see MaxMin can perform very well, and always keep the stability during the process. With this number of requests, the non-dynamic and non-machine learning algorithms show the stable and perform not much different from each other's. But RCBA with Naïve Bayes and KMeans supports, it shows the advantages of Machine Learning and the effectiveness of it.

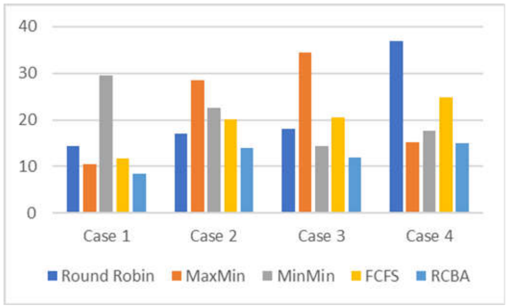

Figure 15. Average Response time of 5 algorithms in 4 cases

Figure 15 is the average response time (ART) of 5 algorithms in 4 cases. This figure is generated by the average of total requests, not only the selected requests like figure 11 to figure 14. In general, RCBA is the smallest average response time, performs best among the 5 algorithms. In this figure, we can see the characteristics of each algorithm: MaxMin performs well with more requests; FCFS 's ART is increasing by the increment of requests; RoundRobin and MinMin perform not really good at all.

Through 04 cases of different number of requests with the same input, we can see that the distribution of RBCA is quite stable and reasonable. The response time of virtual machines is satisfied compared to other algorithms on the cloud (in the case of few and many requests).

V. Conclusions

This simulation experiment is only simulating a group of virtual machines, not counting the expansion of the virtual machine pool (VM pool) to reduce the load in case of need, because it is assumed that this group of virtual machines can handle maximum how many requests, if exceeded, we will expand the pool. However, the simulation experiment with a large request of over 1000 requests require a more powerful computer and a better processor, so this is the limitation of this simulation experiment.

Presenting the simulated experimental model, the parameters as well as the given scenario are based on the request process of the browsers in the cloud environment. From there, record the forecast response time parameters of virtual machines of the cloud. Running simulation experiments with parameters of 5 virtual machines, under load from 25 to 1000 requests has shown relatively good results, the distribution of requests to virtual machines is quite uniform and feasible. In the bigger number of requests, our proposed RCBA shows the advantages of Machine Learning and Statistics, it can take the trend of the request, then behave the appropriate action to handle them. In future, we can continue study the ML and its application in LB, expand the requests and expand the cloud, including the VM migration and Pool manager.

Acknowledgment

We would like to express our big appreciation to the institutions: The Saigon International University and Ho Chi Minh City Open University in Vietnam for supporting us in our research on this topic. We wish to thank the students at the university who support us in this research experiment to achieve the desired results for this paper.

References

[1] Y. Wen and C. Chang, "Load balancing job assignment for cluster-based cloud computing," *2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2014, pp. 199-204, doi: 10.1109/ICUFN.2014.6876780.

[2] G. Shao and J. Chen, "A Load Balancing Strategy Based on Data Correlation in Cloud Computing," *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, 2016, pp. 364-368.

[3] B. T. Khiết, N. T. Nguyệt Quế, H. Đ. Hưng, P. T. Vũ, and T. C. Hùng, "Mô Hình Cấp Phát Tài Nguyên Điện Toán Đám Mây Công Bằng Dựa Trên Lý Thuyết Trò Chơi," *Fair - Nghiên Cứu Cơ Bản Và Ứng Dụng Công Nghệ Thông Tin* - 2017, 2017.

[4] Huỳnh Quyết Thắng, Điện Toán Đám Mây, NXB Bách Khoa Hà Nội, Hà Nội, pp 10-15, 2014.

[5] [4] Y. Zhang and M. R. Lyu, *QoS prediction in cloud and service computing: Approaches and applications*, 1st ed. Singapore, Singapore: Springer, 2017.

[6] Melvin M. Vopson Senior Lecturer in Physics, "The world's data explained: how much we're producing and where it's all stored," *The Conversation*, 20-Oct-2021. [Online]. Available: https://theconversation.com/the-worlds-data-explained-how-much-were-producing-and-where-its-all-stored-159964. [Accessed: 31-Oct-2021].

[7] A. Dhari and K. I. Arif, "An Efficient Load Balancing Scheme for Cloud Computing," *Indian Journal of Science and Technology*, vol. 10, no. 11, pp. 1–8, 2017.

[8] L. Sheikhani, Y. Chang, C. Gu and F. Luo, "Modifying broker policy for better response time in datacenters," *2017 3rd IEEE International Conference on Computer and Communications (ICCC),* 2017, pp. 2459-2464, doi: 10.1109/CompComm.2017.8322977.

[9] S. K. Mishra, M. A. Khan, B. Sahoo, D. Puthal, M. S. Obaidat and K. Hsiao, "Time efficient dynamic threshold-based load balancing technique for Cloud Computing," *2017 International Conference on Computer, Information and Telecommunication Systems (CITS)*, 2017, pp. 161-165, doi: 10.1109/CITS.2017.8035327.

[10] M. Maqsood, S. A. Lashari, M. A. Saare, S. A. Sari, Y. M. Hussein, and H. O. Hatem, "Minimization Response Time Task Scheduling Algorithm," *IOP Conference Series Materials Science and Engineering*, vol. 705, p. 012008, 2019, doi:10.1088/1757-899X/705-1/012018.

[11] N. X. Phi, L. N. Hieu, and T. C. Hung, "Load Balancing Algorithm on Cloud Computing for Optimize Response Time," *International Journal on Cloud Computing: Services and Architecture*, vol. 10, no. 3, pp. 15–29, 2020.

[12] H. L. Ngoc, T. N. T. Huyen, X. P. Nguyen, and C. H. Tran, "MCCVA: A New Approach Using SVM and KMeans for Load Balancing on Cloud," *International Journal on Cloud Computing: Services and Architecture*, vol. 10, no. 3, pp. 1–14, 2020.

[13] S. Negi, M. M. S. Rauthan, K. S. Vaisla, and N. Panwar, "CMODLB: an efficient load balancing approach in cloud computing environment," *The Journal of Supercomputing*, vol. 77, no. 8, pp. 8787–8839, 2021.

[14] T. J. B. Durga Devi, A. Subramani, and P. Anitha, "Modified adaptive neuro fuzzy inference system based load balancing for virtual machine with security in cloud computing environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 3, pp. 3869–3876, 2021.

[15] D. A. Shafiq, N. Z. Jhanjhi, and A. Abdullah, "Load balancing techniques in cloud computing environment: A review," *Journal of King Saud University – Computer and Information Sciences*, 2021.

Tran Cong Hung was born in Vietnam in 1961. He received the B.E in electronic and Telecommunication engineering with First class honors from Ho Chi Minh City University of Technology in Vietnam in 1987. He received the B.E in Information and Computer Engineering from Ho Chi Minh City University of Technology in

Vietnam in 1995. He received the Master of Engineering degree in Telecommunication Engineering from postgraduate department, Hanoi University of Technology in Vietnam in 1988. He received Ph.D at Hanoi University of Technology in Vietnam in 2004. His main research areas are B–ISDN performance parameters and measuring methods, QoS in high-speed networks, MPLS. He is currently Associate PhD. of Faculty of Information Technology, The Sai Gon International University, Ho Chi Minh City VietNam.

Tran Duy Tien is an IT specialist in banking field in Ho Chi Minh City, Vietnam. He works and develops the network with better performances for many famous banks in Ho Chi Minh City such as Vietnam EximBank, DongA Bank, etc. He is completing his Master degree in 2021, joining Ph.D Tran Cong Hung ‘s team in cloud computing.

Le Ngoc Hieu has been working in IT industry as an IT System Architect since 2010. In 2018, He completed master’s degree at Post & Telecommunication Institute of Technology. As now, currently, he is A PhD candidate in Information System at Post & Telecommunication Institute of Technology, Vietnam. He is now working at 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.