INTERNATIONAL JOUNAL OF: ENGINEERING RESEARCH AND TECHNOLOGY

TIMBOU-AFRICA ACADEMIC PUBLICATIONS
INTERNATIONAL JOURNAL, AUGUST, 2022 EDITIONS
VOL. 10 NO. 5 ISSN: 2707-5654

EFFICIENT HYBRID LOAD BALANCING ALGORITHM IN CLOUD COMPUTING

***BABANI SANI; **ABDULKADIR MAIGARI TURAKI; ***SALAMATU UMAR; & ****SAIFULLAH IBRAHIM WAZIRI**

**Department of Mathematical Sciences, School of Science, Bauchi State University, Gadau. **Department of Computer Science, Federal Polytechninc, Bauchi. ***Department of Computer Science, College fo Agriculture, Ganye, Adamawa State. ****Executive Director and other Revenue, Gombe State internal Revenue Services*

ABSTRACT

An essential requirement in cloud computing environment is the efficient algorithm that will allow the current jobs to be executed with the given constraints. The algorithms should order the jobs in a way where balance between improving the quality of services and at the same time maintaining the efficiency and fairness among the jobs. It is a

Introduction

Nowadays, many companies offering services to the customers based on the concept of "pay as a service", where each customer pays for the services obtained from the service providers. The cloud environment provides different platform by creating a virtual machine that assists users in accomplishing their jobs within a reasonable time and cost-effectively without sacrificing the quality of the services. The National Institute of Standards and Technology's (NIST) define a Cloud computing as "cloud computing is a

new solution and strategy to achieve high availability, flexibility, cost reduced and on demand scalability. However, cloud computing has many challenges such as poor resource utilization which has deep impact in the performance of cloud computing. These problems arisen due to the huge amounts of information sharing between users. So, there is a need for efficient and powerful cloud computing load balancing algorithms in order to improve the performance of cloud computing. This research work proposes a hybrid algorithm to improve the performance and efficiency in cloud computing environment. The hybrid algorithm has been evaluated and compared with other algorithms using CloudAnalyst simulator. The proposed algorithm has been tested and the experiments results showed that our algorithm improves the cloud system performance by decreasing the response and the data center processing time compared with other algorithms.

Keywords: *Cloud Computing, Load Balancing Algorithm, CloudAnalyst, Response, Time, and Processing Time*.

model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (Singh et al., 2016). Load balancing is considered as one of the challenges in cloud computing, it is the major factor to improve the performance of the cloud computing.

The main problem is to schedule the incoming request in a way so with minimum response time, efficient resource utilization and at the same time resources should not be underutilized

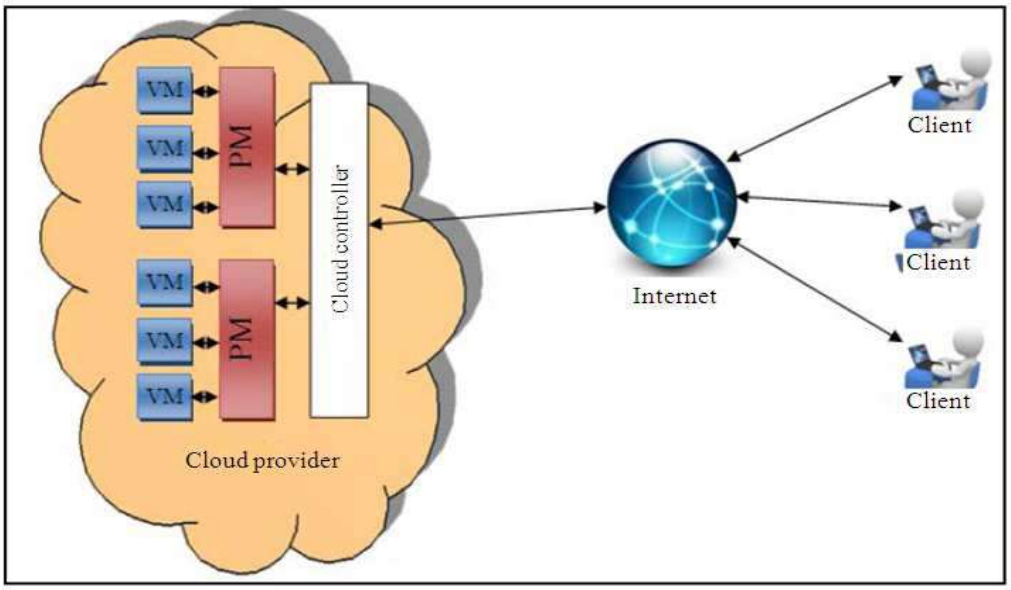


Figure 1: The proposed cloud framework (Source needed)

2.0 Related works

Many researchers work have been done to address the problems of load balancing and job scheduling in cloud computing environment; among which are the works of reviewed from the following researchers;

James et. al, (2013) provided an algorithm called weighted Active VMLoadBalencer to decrease the response time, and data processing time. In this algorithm they compute the power of VM's in the datacenter and use index table to store the count of requests that are currently allocated in VM. When a new request is received, the load balancer looks at the table and identifies the least VM loaded. Then the result return to the datacenter and the datacenter allocates the VM. Finally when the VM finishes, it will notify the datacenter and the datacenter will de-allocate it. The authors built index table to monitor each node in the system to quickly know the status of the node and to allocate the best VM. Although this algorithm decrease the response time, but they need to compare their results with other algorithms such as Equally Spread Current Execution (ESCE) and Throttled in order to evaluate the results.

Sethi et. al, (2014) introduce a load balancing algorithm using fuzzy logic with Round Robin (RR) algorithm. The algorithm is based on various parameters such as processor speed, and assigned load in VM and etc.

The algorithm maintains the information of each VM and numbers of requests currently allocated to VM. When a new request is received, the load balancer searches for the least loaded VM and allocate it, but if there are more than one VM, the selection will be based on processor speed and load in VM using fuzzy logic.

This algorithm enhanced the performance of load balancer and decreased the response time. In addition, the results referred that its performance is better than RR algorithm. The drawback of this approach that authors had focused only on how to decrease the response time of job scheduling and they ignored talk about processing cost. In addition, the researchers compared their results with only RR algorithm which had been enhanced and improved by many researchers before.

Sharma et. al, (2013) propose a new enhancement scheduling algorithm EEA, the main purpose of the algorithm is to find the expected response time of each VM to achieve the maximum throughput and decrease response time to avoid overhead. They compared their algorithm with three algorithms, Round Robin (RR), Equal Spread Current Execution scheduling and throttled algorithm. The result shows that overall response time and data center processing time is improved as well as cost is reduced in comparison to the existing scheduling parameters. However, this approach is limited on enhancement of response time and how to achieve the maximum throughput but does not handle the fault tolerance and the problem caused by deadlocks and server overflow.

Hu et. al, (2009) propose a new algorithm to enhance job scheduling using a genetic information. The algorithm uses a historical data and current state of the system. It makes a mapping relationship between the set of physical machines and the set of VMs. It chooses the least-affective solution by computing ahead influence of the system after the deployment of the needed VM resources. They used some equation to find the best scheduling solution using population. The experimentation results show an improvement in the utilization of resources. On the other hand, the proposed algorithm has high cost to store and retrieve the historical data of the system nodes, and this may also increase the response time and the processing cost.

Fang et. al, (2012) try to obtain high resource utilization and meet dynamic requirements of task by providing a two level task scheduling mechanism based on load balancing in cloud computing. They paper improve the response time, resources utilization by mapping task to VMs and then VMs to host resources. They use the first level of scheduling (from user's application to the VM) to create a description of VM including the task of computing resources, network resources, storage resources, etc. and used the second level scheduling (from the VM to host resources) to find appropriate resource for VM. This approach may have succeeded in improving the resource utilization, but we think that using two levels of task scheduling would increase the response time compared with other load balancing algorithms.

Methodology

Cloudsim Simulator (*CloudAnalyst)*

The proposed algorithm should be tested in a cloud computing environment. For that purpose we have two choices, either use a real test beds such as Amazon EC2 or use simulation tools to simulate a cloud environment. In our work we prefer to use simulator for several reasons:

1. The experiments on real cloud environment will be limited to the cloud provider configurations, so tests cannot examine different configurations.
2. The conditions prevailing in the Internet-Base environment are beyond the control of the tester and this may affect the tests results.
3. Applying tests on real cloud incurs payments.

There are varieties of simulators for modeling cloud computing but CloudSim is one of the best. CloudSim is an open source Java-based simulation toolkit developed by the GRIDS Laboratory at University of Melbourne (Pakize, *et al* 2014). It is an extensible simulation toolkit that enables modelling and simulationof Cloud computing environments (Buyya et al 2014). The CloudSim toolkit supports modeling of infrastructures containing Cloud Data Centers, Virtual Machines (VM), users, user workloads, and pricing models (Buyya et al 2014)

Simulation Environment Cloudsim

The proposed algorithm is tested in a cloud computing environment. We have two choices to test it, the first choice is to use a real test such as Amazon EC2, and the second is to use simulation tools to simulate a cloud environment. In our work we prefer to use a simulator, because using real test limits the experiments to the scale of the infrastructure, and makes the reproduction of results an extremely difficult undertaking, Buyya, *et al*, (2009). Also it is very difficult and time consuming to measure performance in real cloud environment, (Ray and De Sarkar, 2012).

CloudAnalyst Metrics

One of the most important features of CloudAnalyst is that it gives a final report for the simulation result. This report summarizes the results of some metrics which are used for evaluation process. Those metrics are listed below (Ray and De Sarkar, 2012):

- **Overall response time: Minimum, Maximum and Average in ms.**

This metric presents the overall response time for the whole cloud system. The Average score present the average response time for all user bases, while the Minimum and the Maximum present the highest and the lowest response time between user bases. For evaluation process, researchers considered the average response time to compare their works with others

- **Overall processing time in the data center: Minimum, maximum and average in ms.**

The same as response time CloudAnalyst calculate the average processing for all the data centers. Also it presents the highest and the lowest processing time scored between all the data centers.

- **Response time per user: Minimum, maximum and average**

For every user base, the minimum, maximum and average response time is calculated and presented in a summery table.

- **Minimum, maximum and average time per data center.**

For every data center, CloudAnalyst calculate the minimum, maximum and average processing time and present it in a summery table.

- **Cost of execution in $**

The final result report gives details about the cost of execution considering: Virtual machine total cost, Cost per VM of Data Center, Cost of data in each data center, and Total cost in each data center.

Evaluation Metrics

There are various metrics used to evaluate different techniques. In our work we used two metrics to measure the performance as follow:

1. Average response time: It can be measured as, the time interval between sending a request and receiving its response. It should be minimized to boost the overall performance (Pakize, S.R., 2014).

2. Average processing time: It is the amount of time actually needed to process a task.

We measured these two metrics for our algorithm and then compare them with the response and processing time for three popular load balancing algorithms which are: Round Robin (RR), ESCE, and Hybrid algorithms, in order to evaluate the improvement of our algorithm.

Experiments and Results

We studied the problem without the effect of network delay, then we tested the algorithm in heterogeneous environment of hosts, and each machine has different number of CPUs and speed. Finally, we testes the impact of effect of network delay on the hybrid algorithm with considering the capacity of CPU factor and in the heterogeneous environment of hosts; each machine has different number of CPUs and speed. We compared the hybrid algorithm with Round robin, Equally Spread Current Execution (ESCE) algorithms, respectively.

Experiment 1: Test the hybrid algorithms without considering The Processing Power

Data Center Configuration

One Data Center is used with 50 Virtual Machines and 20 Physical HW Units all of them have the same number of processors and the same processor speed. Table 4.1 present the Data Center configurations while

TIJERAT
ISSN: 2707-5654

Table 4.2 shows the configuration of the Physical Hardware of the Data Center.

Users are grouped by a factor of 1000, and requests are grouped by a factor of 100. Each user request requires 250 instructions to be executed. The simulation duration took one day. We used the response time and processing time metrics to compare the algorithm with other current algorithms.

Table 4.1 Application development Configuration used in Experiment 1

Data Center	#VMs	Image Size	Memory	BW
DC1	50	10000	1024	1000

Table 4.2 User Bases Configuration for Experiment 1

Name	Region	Requests per user per Hr.	Data Size per req.(Bytes)	Peak Hours Start(GMT)	Peak Hours End(GMT)	Avg. Peak Users	Avg. Offpeak Users
UB1	0	12	100	13	15	400000	40000
UB2	1	12	100	15	17	100000	10000
UB3	2	12	100	20	22	300000	30000
UB4	3	12	100	1	3	150000	15000
UB5	4	12	100	21	23	500000	50000
UB6	5	12	100	9	11	800000	80000

Table 4.3 Data Centers Configuration for Experiment 1

ID	Memory (Mb)	Storage (Mb)	Available BW	Number of Processors	Processor Speed	VM Policy
0	204800	1000000000	1000000	4	200	TIME_SHARED
1	204800	1000000000	1000000	5	500	TIME_SHARED
2	204800	1000000000	1000000	2	2000	TIME_SHARED
3	204800	1000000000	1000000	2	5000	TIME_SHARED
4	204800	1000000000	1000000	2	90000	TIME_SHARED

TIJERAT
ISSN: 2707-5654

Table 4.3 shows that the VM policy used are shared-time.

Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing (Marisol *et al*., 2014). Table 4.4 shows the results for when the capacity of the CPU was not considered

Results

Table 4.2.1(A) Results for Response Time Results without Considering Capacity of CPU factor

Response Time For Experiment One				
Algorithms		**Avg.(ms)**	**Min(ms)**	**Max(ms)**
Round Robin	Response Time	262.16	37.69	64.89
ESCE		262.11	37.69	607.9
Hybrid		210.06	27.06	41.12

In this experiment we will test the performance of the proposed algorithm along with Round Robin and ESCE algorithms. The response time and the processing time of the Hybrid algorithm had compared with response time and the processing time of the RR and Random Algorithms and that Hybrid algorithm recorded the best response time as well as the processing time as indicated in Fig. 4.1(A) and Fig. 4.1(B) respectively.

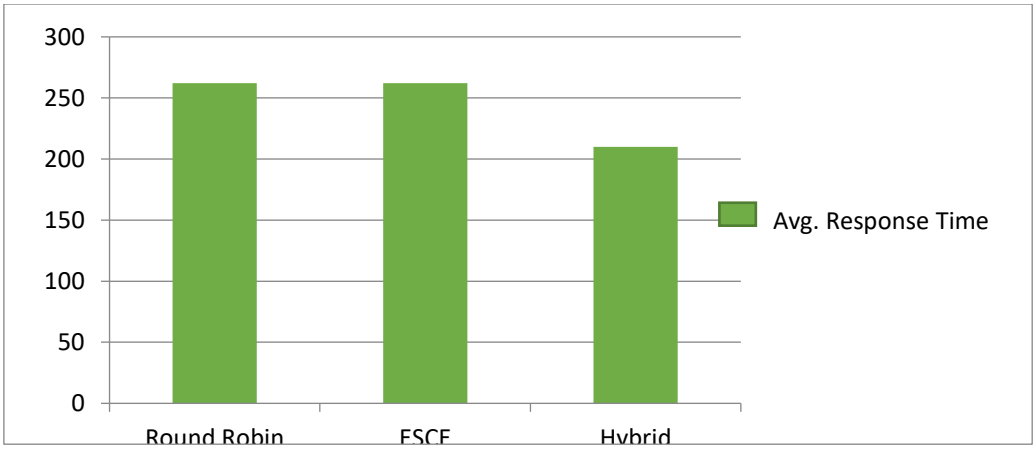


Figure 4.2.2 (A) Response Time without Considering Capacity of CPU at the Regions

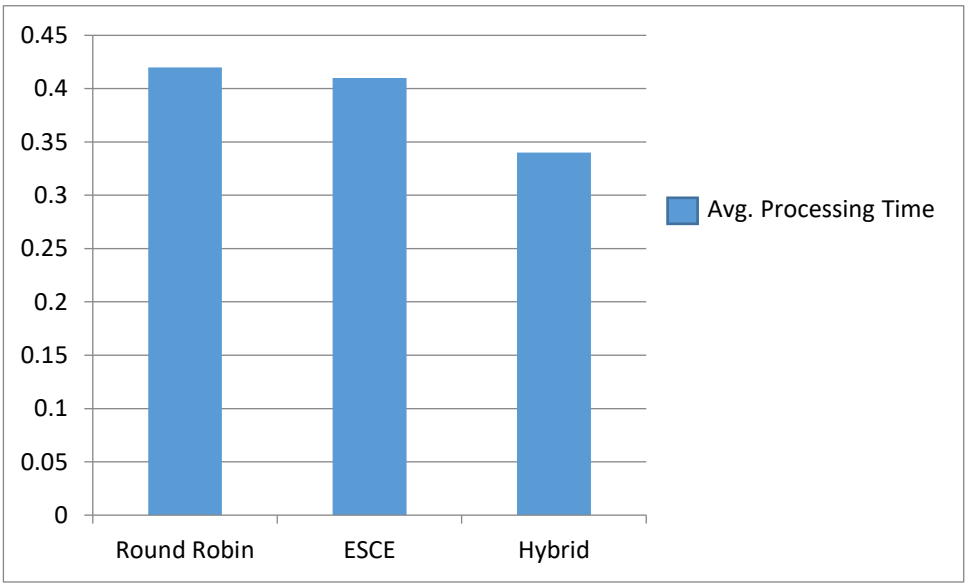
Table 4.2.3 (B) Processing Time Results without Considering Capacity of CPU at the Reigion

Processing Time For Experiment One				
Algorithms	**Processing Time**	**Avg.(ms)**	**Min(ms)**	**Max(ms)**
Round Robin		0.42	0.09	1.78
ESCE		0.41	0.09	1.49
Hybrid		0.34	0.02	0.61

Processing Time

The results show that Hybrid algorithm has the best processing time compared with Round Robin and ESCE. Hybrid algorithm recorded the Average Processing Time of 0.34 (ms), ESCE recorded 0.41(m.s) and Round Robin recorded 0.42 (ms). These results can be better represented in a graphical form for easy analysis

Figure 4.2.4 Results For The Average Processing Time Comparison When Capacity of Factor CPU Considered at the Region



Discussion

From fig. 4.2.1 (A) and Fig. 4.2.2(B), It showed that the Hybrid algorithm had better result than the Equally Spread Current Execution (ESCE) and Round robin algorithm. Also we found that the Equally Spread Current

Execution (ESCE) is better than Round Robin algorithm because it is very simple and does not have the overhead computation as Equally Spread Current Execution (ESCE).
In addition the Hybrid recorded the best min. response time 210.06 (ms) and the best min. processing time 0.34(ms). Because the Hybrid does not need a complex computation to takes decision to allocate VM.

Experiment 2: Test the effect of considering the capacity of CPU.

Table 4.3.1 Test the effect of considering the capacity of CPU in Experiment 2

Name	Region	Requests per user per Hr.	Data Size per req.(Bytes)	Peak Hours Start(GMT)	Peak Hours End(GMT)	Avg. Peak Users	Avg. Off-peak Users
UB1	0	12	100	6	15	20000	2000
UB2	1	12	100	15	17	1000	1000
UB3	2	12	100	17	22	14000	1000
UB4	3	12	100	1	6	1500	1500
UB5	4	12	100	21	23	5000	500
UB6	5	12	100	5	11	8000	800

In this experiment we have test the performance of the Hybrid algorithm with different input values to test the effect of CPU capacity. In addition we have study the effect of the instruction length on the performance of hybrid algorithm by comparing the response time and the processing time with RR and Random Algorithms. This experiment studied the effect of considering the Capacity of CPU factor with the three algorithms in heterogeneous environment; each machine has different number of CPUs and speed.

Configuration

The configuration settings are as in experiment one in table 4.2.1, 4.2 and 4.2.2.

Results

After successful execution of the experiment, the results obtained were recorded and compared with our benchmark algorithms. Table 4.5 and

in Figure 4.2 showed that our proposed algorithm outperformed that of Round Robin with 35% and Equally Spread current Execution with 34.5% by 30% respectively.Table 4.3.4 (B1) Response Time Results for Testing the Effect of Capacity of CPU Factor

Response Time For Experiment Two				
Algorithms		Avg.(ms)	Min(ms)	Max(ms)
Round Robin	Response Time	132.37	39.4	716.26
ECSP		130.91	38.96	713.59
Hybrid		115.24	21.01	423.12

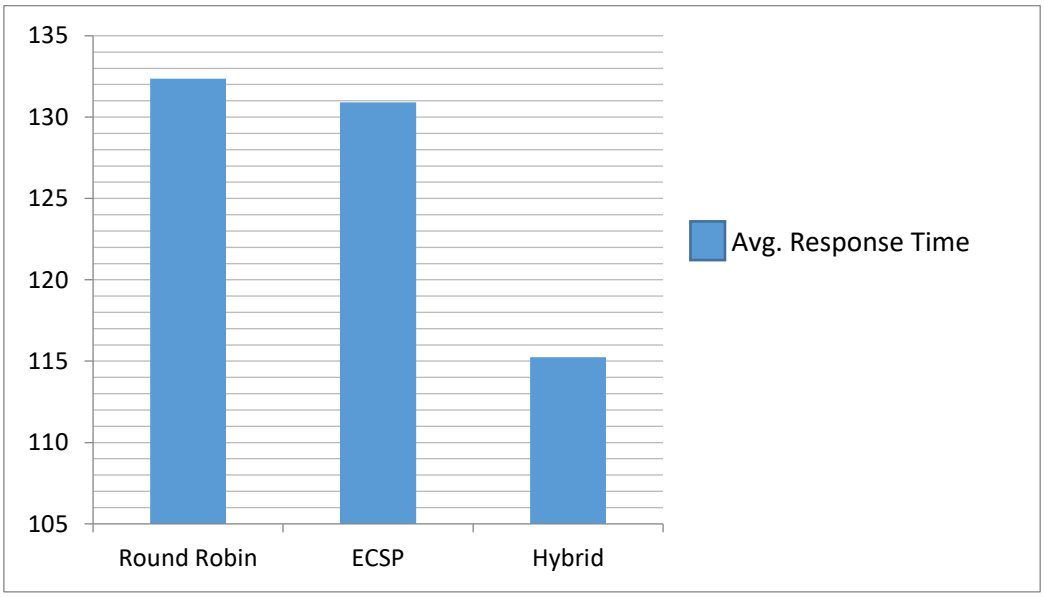


Fig 4.3.5(B1) Response Time Results for Testing the Effect of Capacity of CPU Factor

These results can be better represented in a graphical form for easy analysis

Table 4.3.6 (B1) Processing Time Results for Testing the Effect of Capacity of CPU at the region

Processing Time For Experiment Two				
Algorithms		Avg.(ms)	Min(ms)	Max(ms)
Round Robin	Processing Time	80.31	0.44	648.92
ECSP		78.83	0.44	645.25
Hybrid		51.4	0.12	512.18

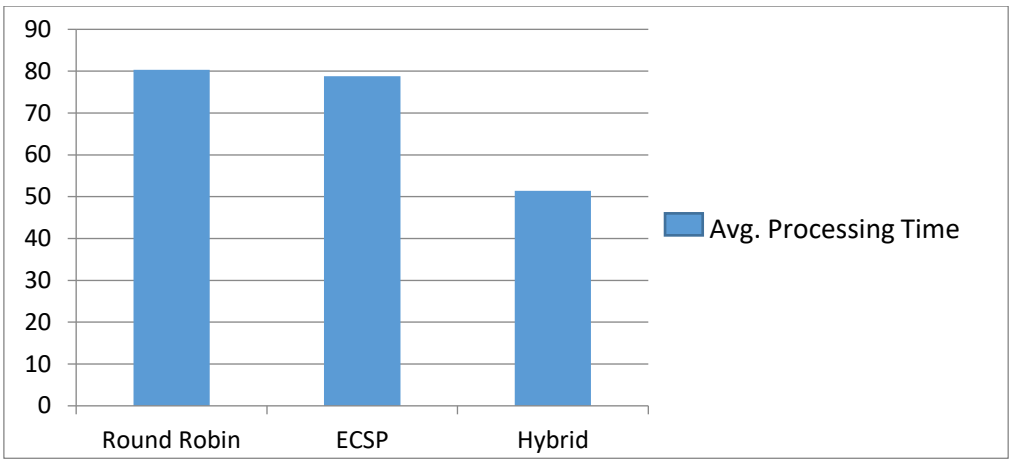


Fig 4..3.7(B1) Processing Time Results for Testing the Effect of Capacity of CPU Factor

Figure 4.2 Results for the Comparison When Testing the Effect Capacity of CPU at the region

Discussion

In this experiment the VMs distributed on the hosts according the hosts qualification and the CPU capacity, the results showed that when considering the CPU capacity factor, the host that has best qualification has more VMs than other hosts, so when we select K nodes randomly from the VMs and choose the least loaded one from the selected VMs, the response time will be improved because most of VMs selected will be in the qualified host.

The proposed hybrid algorithm returned average response time of **115.24** (ms) and average processing time of 51.40 (ms). This result is better than that of Round Robin which had average response time of **132.37** (ms) and the average processing time of **80.31** (ms)**.** This means minimizing the number of VM to 15 and with considering the CPU capacity factor, the hybrid algorithm decreased the overhead computation. The hybrid algorithm adds a significant improvement on average response time and on processing time compared with Round Robin and Equally Sprayed Current Execution Algorithms.

Experiment 3: Response time and processing time results for testing the effect of network delay

This experiment tested the effect of network delay on the hybrid algorithm with considering the Capacity of CPU factor in the

heterogeneous environment of hosts. Each machine has different number of CPUs and speed.

Configuration

The configuration files are in Table 4.6. And in order to test the effect of network delay we distributed the user's base in 6 regions, UB1 N-America, UB2 in S. America, UB3 Europe, and UB4 in Asia, UB5 Africa and UB6 in Oceania as in Table 4.6 shows the user bases configuration setting.

Table 4.5.3 User Bases Configuration Used in Experiment3

Name	Region	Requests per user per Hr.	Data Size per req.(Bytes)	Peak Hours Start(GMT)	Peak Hours End(GMT)	Avg. Peak Users	Avg. Offpeak Users
UB1	1	12	100	13	14	400000	40000
UB2	2	12	100	9	10	100000	10000
UB3	3	12	100	20	21	300000	30000
UB4	4	12	100	1	2	150000	15000
UB5	5	12	100	5	6	500000	50000
UB6	6	12	100	17	18	800000	80000

Result: After the successful execution of the experiment, the results obtained were recorded and compared with our benchmark algorithms. Table 4.8 shows the Response timer and processing time results for testing the effect of network delay.

Table 4.5.4(A) Response Timer and Processing Time Results for Testing The Effect of Network Delay

Responce Time For Experiment Three				
Algorithms	Response Time	Avg.(ms)	Min(ms)	Max(ms)
Round Robin		280.26	161.14	370.64
ESCE		300.06	237.06	369.12
Hybrid		262.13	37.69	607.9

TIJERAT
ISSN: 2707-5654

Table 4.5.5 (B) Processing Timer Results for Testing The Effect of Network Delay

Processing Time For Experiment Two				
Algorithms	**Processing Time**	Avg.(ms)	Min(ms)	Max(ms)
Hybrid		0.23	0.09	1.78
Round Robin		0.31	0.02	0.66
ESCE		0.34	0.02	0.61

These results can be better represented in a graphical form as shown below:

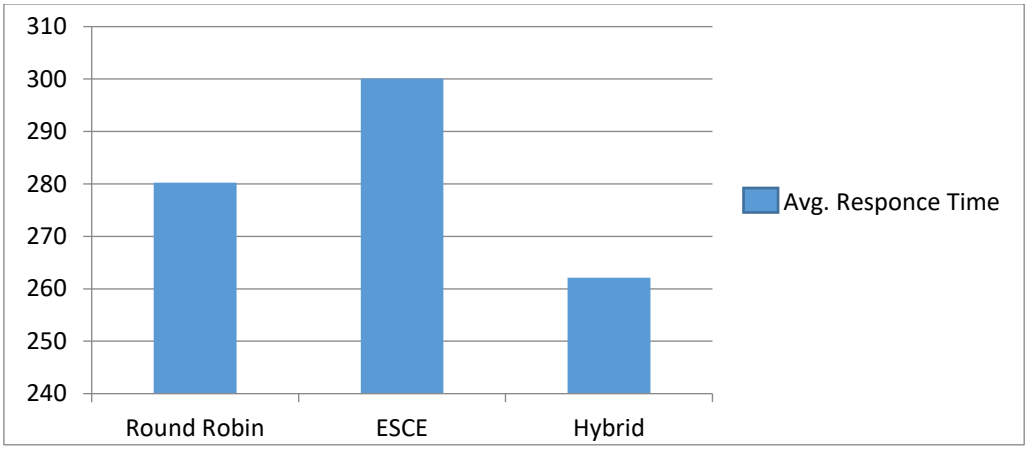


Figure 4.5.6 Response Timer Results for Testing the Effect of Network Delay

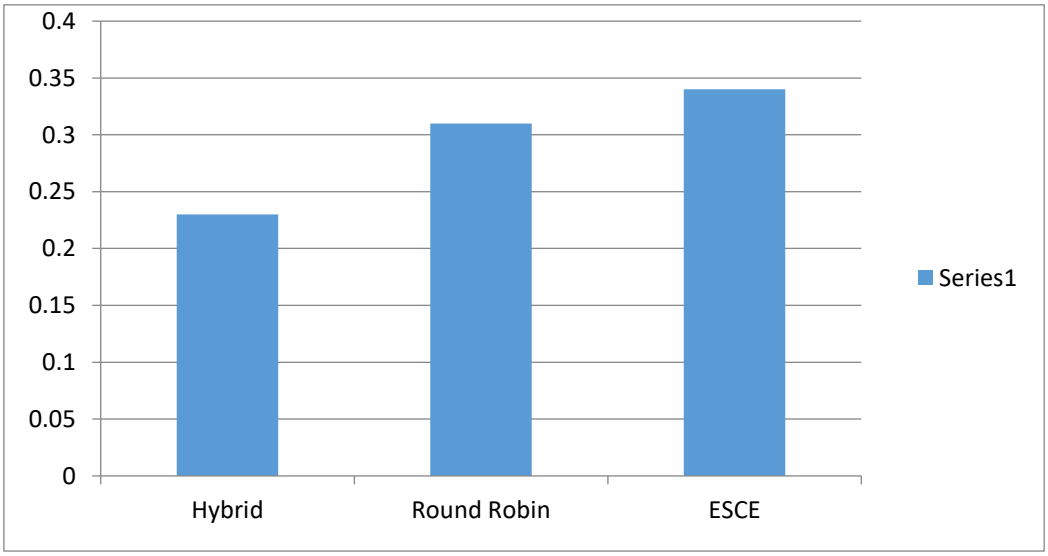


Figure 4.5.7 Response Timer Results for Testing the Effect of Network Delay

Table 4.5.7 shows the results obtained from the experiments when network delay was taken into consideration. It clearly shows that hybrid algorithm has Response time of 262.13 (ms) and Processing time of 0.23 (ms), Equally Spread Current Execution (ESCE) has response time of 300.06 (ms) and Processing time of 0.34 (ms) and Round Robin has response time of 280.26 (ms) and Processing time of 0.31 (ms) respectively.

References

Florence, A.P. and Shanthi, V., (2013). *Intelligent Dynamic Load Balancing Approach for Computational Cloud.* International Journal of Computer Applications:

Zhang, Q., Cheng, L., and Boutaba, R., *Cloud computing: state-of-the-art and research challenges.* Journal of Internet Services and Applications, **1**(1): pp. 7-18,(2010).

Shah, J., *Cloud Computing:* (2014). *The Technology for Next Generation.* International

Journal of Advances in Computer Science and Technology. Sharma, T. and Banga, V.K., (March 2013). *Efficient and Enhanced Algorithm in Cloud Computing.* International Journal of Soft Computing and Engineering (IJSCE).

Dave, S. and Maheta, P., *Utilizing Round Robin Concept for Load Balancing Algorithm at Virtual Machine Level in Cloud Environment.* International Journal of Computer Applications, **49**(4),(2014).

Ratan, M. and Anant, J., *Ant colony Optimization: A Solution of Load Balancing in Cloud.* International Journal of Web & Semantic Technology (IJWesT), **3**(2): pp. 33-50,(2012).

Singhal, U. and Jain, S., *A New Fuzzy Logic and GSO based Load balancing Mechanism for Public Cloud.* International Journal of Grid Distribution Computing, **7**(5): pp. 97-110,(2014).

Singhal, U. and Jain, S., *A New Fuzzy Logic and GSO based Load balancing Mechanism for Public Cloud.* International Journal of Grid Distribution Computing, **7**(5): pp. 97-110,(2014).

Mia, N., Zhangb, Q., Riskac, A., Riskac, E., and Riedel, E., *Performance impacts of autocorrelated flows in multi-tiered systems.*

Pakize, S.R., Khademi, S.M., and Gandomi, A., *Comparison Of CloudSim, CloudAnalyst And CloudReports Simulator in Cloud Computing.* International Journal of Computer Science And Network Solutions, **2**: pp. 19-27,(2014).

Ratan, M. and Anant, J., *Ant colony Optimization: A Solution of Load Balancing in Cloud.* International Journal of Web & Semantic Technology (IJWesT), **3**(2): pp. 33-50,(2012).

Alakeel, A.M., *A Guide to Dynamic Load Balancing in Distributed Computer Systems.*International Journal of Computer Science and Network Security (IJCSNS), **10**(6): pp. 153-160, (2010).

Calheiros, R.N., Ranjan, R., De Rose, C.A.F., and Buyya, R., *CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services*. pp. 1-9, (2009).

Pakize, S.R., Khademi, S.M., and Gandomi, A., *Comparison Of CloudSim, CloudAnalyst And CloudReports Simulator in Cloud Computing.* International Journal of Computer Science And Network Solutions, **2**: pp. 19-27,(2014).