

Applicability of MMRR load balancing algorithm in cloud computing

Abiodun Kazeem Moses, Awotunde Joseph Bamidele, Ogundokun Roseline Oluwaseun, Sanjay Misra & Adeniyi Abidemi Emmanuel

To cite this article: Abiodun Kazeem Moses, Awotunde Joseph Bamidele, Ogundokun Roseline Oluwaseun, Sanjay Misra & Adeniyi Abidemi Emmanuel (2021) Applicability of MMRR load balancing algorithm in cloud computing, International Journal of Computer Mathematics: Computer Systems Theory, 6:1, 7-20, DOI: [10.1080/23799927.2020.1854864](https://doi.org/10.1080/23799927.2020.1854864)

To link to this article: <https://doi.org/10.1080/23799927.2020.1854864>



Published online: 14 Dec 2020.



Submit your article to this journal [↗](#)



Article views: 19








View related articles [↗](#)



View Crossmark data [↗](#)



Applicability of MMRR load balancing algorithm in cloud computing

Abiodun Kazeem Moses ^a, Awotunde Joseph Bamidele ^b, Ogundokun Roseline Oluwaseun ^a, Sanjay Misra ^c and Adeniyi Abidemi Emmanuel ^a

^aDepartment of Computer Science, Landmark University, Omu-Aran, Nigeria; ^bDepartment of Computer Science, University of Ilorin, Ilorin, Nigeria; ^cDepartment of Electrical and Information Engineering, Covenant University, Ota, Nigeria

ABSTRACT

One of cloud computing's fundamental problems is the balancing of loads, which is essential for evenly distributing the workload across all nodes. This study proposes a new load balancing algorithm, which combines maximum-minimum and round-robin (MMRR) algorithm so that tasks with long execution time are allocated using maximum-minimum and tasks with lowest execution task will be assigned using round-robin. Cloud analyst tool was used to introduce the new load balancing techniques, and a comparative analysis with the existing algorithm was conducted to optimize cloud services to clients. The study findings indicate that 'MMRR has brought significant changes to cloud services. MMRR performed better from the algorithms tested based on the whole response time and cost-effectiveness (89%). The study suggested that MMRR should be implemented for enhancing user satisfaction in the cloud service.

ARTICLE HISTORY

Received 27 August 2020
Accepted 13 November 2020

KEYWORDS

Grid computing; cloud computing; scheduling algorithm; distributed computing

1. Introduction

Cloud computing [27] is an expansive research area that brings excellent value to the costs of businesses around the world. Cloud computing [11] is a service model of information technology, using the world's previous networks and distributed computing concepts that assist users getting access to different resources and data in the cloud. The ability to process complex data quickly and efficiently has made several organizations to support it such as Google, Facebook etc. [13,29]. Cloud computing is established on virtualization technology [28], through network services to provide users with the necessary resources, application platform, software and other services. Cloud computing is changing the IT industry, changing the way we use software and hardware [17]. Major companies like Google, HP, Amazon, Oracle and others are embracing Cloud computing as a route to computer ability to do many things. Cloud clients nowadays can purchase resources of different natures from various providers. They can lease infrastructure resources, platform resources or software resources, possibly all three types simultaneously. In the cloud environment, we have four distribution models in the cloud; they are private, public, hybrid and community. Proper management of cloud resources is maintained by the proficient and accessible features of cloud computing. Most of the resources in the cloud are in an easily accessible form, which is another crucial cloud system characteristic. The Cloud Service Provider (CSP) offers services to users on a rented basis [19]. The job of CSP to provide the

facilities to the customer is a unique one with the array of available virtual cloud resources. Consequently, scholars have given more consideration towards the balancing of the load. The impact of load balancing on system performance is enormous.

Balancing of load [27] is a process that ensures the job between various nodes in the given setting are not overwhelmed or is idle for any moment. A practical algorithm for balancing of loads will ensure that all the system nodes do almost the same amount of jobs. The algorithm for load balancing is controlling the mapping of the tasks that are assigned to cloud territory, to unused assets, so overall response time available is enhanced as well as the useful resource usage is given. Harmonizing the load has become a key concern in cloud computing because the number of requests in the cloud environment that are issued within a second is not foreseeable. The unpredictability is because of the cloud's ever-changing behaviour.

Static and dynamic algorithm are the standard ways of classifying resource scheduling algorithm. The static algorithm anticipates prior information on the structure and diverse factors of the framework, like restrictions on a storage device, memory etc. An example of a static approach is round-robin (RR) algorithm. The algorithm that can change as they know about the framework is referred to as Dynamic algorithms. The demand of the customer can be effortlessly accomplished with vibrant procedures. However, algorithms that are dynamic works well when equated with static algorithms; the challenging task is in designing and developing an algorithm for the cloud environment that is dynamic [21].

The dynamic algorithm classifies further based on offline mode and online mode. The task to be performed are done at some pre-characterized times for offline mode, and the circumstances are selected based on possible ending time of more significant tasks. The approach is call batch method because the planning is completed in groups. For online mode, the job is allocated when the tasks began to arrive. The critical goal of balancing load in the cloud sphere is to allot jobs dynamically between nodes to please user necessities and offer optimum resource usage by classifying the whole obtainable jobs to different nodes.

The remaining part of this paper includes Section 2 that reviews related works of literature; Section 3 discusses the approach and the performance metrics adopted for the study and also presents the proposed hybridization of MMRR algorithm. Section 4 gives results and discussion of simulation outcomes. Lastly, Section 5 recapitulates the study results and proposal for work in the future.

2. Literature review

Individuals, companies and organizations have gained significant benefits and transformation from cloud computing [1,5], which offers secure personalized and cost-effective services across the internet to all aspects of life daily. Cloud derives from the grid computing, utility computing and simulation concept [31]. Resources of computers in the cloud setting provide the ability to access computing power, bandwidth, storage, etc. it gives users a variety of IT services [28]. The rapid progress in developing cloud systems and applications into cloud computing environments is a very challenging task, according to [7] and requires capable algorithms for efficient resources distribution. Round Robin is another algorithm used in cloud computing to assign resources to the Server; for a specific time, slice, the request is performed in an RR fashion based on the community sharing order [12].

There are several articles on the study of various algorithms and cloud simulation applications that address the relative merits of various options [3,26]. Such research relates to investigative and nature-inspired algorithms [22,30] and evaluating the advantages of cloud analyst for investigating the various algorithms [23]. Contrasted the various variations of RR algorithm like modified round-robin and time slice-based priority-based round-robin (TSPBRR). These algorithms are evaluated in terms of various parameters such as performance, processing time, waiting time and response time. The results findings show that TSPBRR offers a more robust solution with high throughput and better response time.

RR algorithms had the drawback that when the VMs are assigned to the application request of a user, its position would not be maintained, which allows the algorithm to be performed for the request of similar demand again. The authors in [17] have built an enhanced RR which manages the state every time the server runs an application request. Unique data structures achieved it called hash maps to hold VM information allocated to a particular demand for user applications and VM state list that retains the VM status (busy or free). The simulation results showed that this algorithm increased the response time compared with the normal one.

The authors in [8] built an updated algorithm for load balancing called throttled. The algorithm makes sure the best vacant VM is assigned according to the response and processing time to an application request. Here the state of each VM is maintained by 'throttle VM load balancer'. On request to the 'data centre manager' for an application, it requests VM load balancer for throttle to select the right VM based on the application specifications. A technique for biologically inspired bee colony optimization in distributing load in a cloud hosting web services is defined in [20].

Under this methodology, the allocation of web services varies accordingly as the amount of requests for web services varies. The servers are here considered to be virtual servers that contain a list of demands for service. The server calculates a measure called 'profit' which gives us the application request service, founded on performance metrics such as CPU time, etc. The 'advert board' also shows the idle servers if they need any services. Depending on various metrics, the net income of the associated network is measured. The servers will act the character of bees here, and lazy servers will be like the bees waiting for the nectar to fetch.

The ant colony optimization (ACO) updated technique [20] is another biologically inspired load balancing technique. The ACO methods are established on the ant movements in a specific direction using pheromone. ACO is related to the balancing of cloud loads by seeing the system module as ants and the virtual machines (VMs) as nodes. One node will initially be chosen, as the Regional Load Balancing Node (RLBN), which performs as the head node. Ants will emerge from this RLBN and will be able to travel towards the nodes in any direction, according to a load of nodes.

In ACO approach, a table of pheromone is preserved, which keeps the information about loads on the nodes. The main goal of ant is to find and assign the least loaded node to the client, thus distributing the node in the network equally. Established on the standard of analytical hierarchy process, the author in [10] provided a load balancing process called priority-based task scheduling in cloud. The algorithm flow built with three prioritization stages, like resource, scheduling and job levels. The downside of this work was the complexity of the system, the incoherence and the time of consumption.

In [9], the author implemented a new updated HEFT (heterogeneous earliest finish time) algorithm to resolve the HEFT drawbacks. It works on assigning the inbound job amongst different resources in two steps, which reduces the makes-pan [15]. Introduced elastic cloud max-min (ECMM) algorithm with significant enhancements in the basic max-min algorithm. This algorithm maintains a table of all tasks with status. It locates the current job for each server, and the estimated time for each task to be completed – the table details for specific tools to be used when allocating various tasks.

ECMM process is evaluated using cloudsim with other algorithms, and the result shows that it increases the job pending time in comparison with max-min and RR algorithms. Still, the max-min algorithm is faster than ECMM.

NBST algorithm [25] proposed a method that attempts to strike a balance with the load by arranging the VM according to their processing power and the cloudlets according to their length. The list of VM and cloudlets is then submitted to the broker for allocation. Broker assigned through mid-point algorithm and divides the VM list and cloudlet list till we have at most one cloudlet or VM in the inventory and then the allocation of resources is done.

Arulkumar and Bhalaji [4] present the evaluation result of the load balancing algorithm inspired by nature in a cloud environment. The analysis concludes that water wave algorithm is at least 2.5% better than particle swarm optimization; 4.5% better than ACO and 6.9% better than genetic algorithm.

2.1. Load balancing algorithms types used in cloud

The load balancing algorithms that are currently being employed in cloud computing is described below, along with specific considerations:

(1) *Random Algorithm*

The nature of the algorithm is static [12], typically specified in system design or implementation. It randomly selects the node by using a random number generator [16] for each processor; the procedures for the delivery guideline are retained. Although the process works well with similarly loaded systems, when the loads are of diverse computational complexity, there may be some problem. Another problem is that there is no deterministic approach to the algorithm.

(2) *FCFS*

FCFS algorithm [6] is mostly used for parallel processing, and it selects a task coming in and aimed at resource with the shortest waiting queue time. The load balancing strategy that each load balancer has a work queue in which the job is waiting to be executed for its turn. FCFS's advantages are due to its quick and comfortable design.

(3) *Round Robin*

Allocates the nodes in a way that it gives time to each job equally to satisfy the requests in the time-sharing process, i.e. based on the locally defined process distribution directive. RR gives the benefit of a quick response in case workload is distributed equally among the methods. The job cycle time for various systems, however, is not the same. Many nodes will, therefore, be hugely overburdened, whereas some others will keep on idle [12].

(4) *Weighted RR*

The algorithm share weight for each machine according to a define ways [16], the algorithm considers proficiencies of the VM resource. It gives a greater quantity of jobs to the higher capacity VM according to the weightage given to each of the systems. This system works efficiently, nevertheless has an issue in the beginning due to the static interpretation of the weights. It also failed to consider the length of the tasks to select the appropriate VM.

(5) *Dynamic RR*

The difference of dynamic RR compared with weighted RR is that on a continuous cycle, the servers are tracked, and the weights continue to change. That is a form of active load balancing. Dynamic RR uses numerous facets of instantaneous server capabilities scrutiny, as the recent quantity of links per node or the quickest reaction time for the links to be distributed. The only problem with this algorithm is that, because of its complicated existence, load balancing is rarely used [16].

(6) *Least connections*

The load balancing algorithm least connections modes for pool members, an algorithm that is dynamic and allocates links to the pool member (node/server) that is presently handling the fewest open links at the time when a new link request is received. The program transfers a unique link to the server using this algorithm that has the minimum number of new connections [16]. The algorithm is well suited in environs where identical capabilities exist for the servers or load balancing of other materials. The application that uses this is scarcely possible in a simple load balancer.

(7) *Observed*

Although it is not common, the system calculates a vibrant ration value used to allocate connections along with available pool members. It is the alignment of least connections and fastest algorithm. Servers with a better combination between the smallest networks and quickest reaction time earn a larger share of the systems. The observed mode balances the load of the networks by the ratio values given to individual pool participant, favouring the pool participant with the highest ratio value [16].

(8) *Equally spread current execution load (ESCE)*

Communication occurs among the load balancer and the Datacentre Controller for bringing up-to-date the index table leading to overhead in the ESCE algorithm. Besides, this overhead leads to a delay in getting a response to the arrived requests. Continuous monitoring of jobs is required for this algorithm [18] that is available for execution to schedule the tasks to various VM. The task is arbitrarily issued by testing the size and then giving the job to a VM that is not loaded much or can perform this function effortlessly and uses a lesser amount of time, thus providing full throughput.

(9) *Throttled load balancing*

Within this system [2], a suitable VM is looked for to perform the correct operation when a user desire is received. The algorithmic process begins withholding a list of all accessible VMs. The list is much and to make it faster in the method of lookup; indexing is carried out. A user's request is approved when a suitable match is identified depending on the machine's dimension and accessibility. The Server will get a virtual machine assigned. On the other hand, the demand will be lined up by the load balancer.

(10) *Min-Min Algorithm*

The algorithm [12] starts by looking for the shortest completion time for all jobs. Then the quickest time in all the tasks are pick for all resources and scheduled based on the equivalent device. Afterwards, the time use of the allocated task is joined with the completion times of other jobs on that system to update the finishing time on that machine. The task completed is then removed from the machine's list of tasks yet to be assigned. This process is repeated; however, again till every job are dispersed on the resources. The biggest shortcoming of this technique is starvation [14].

(11) *Max-Min Algorithm*

The max-min algorithm [14] is like the min-min process, but the maximum value is chosen after finding the total execution times. The highest completion time on any resources in the jobs, based on what the job is the plan on the equivalent machine. Subsequently, the completing time of the apportioned task is joined to the performance periods of new tasks on that machine to update the performance period on that machine. The apportioned job is removed from the list of jobs yet to be allotted to the devices. The process is followed again until every task is allocated to the resources.

(12) *Token Routing*

The algorithm was designed to minimize the cost of the system by shifting tokens around the device. Because of communication bottleneck, agents cannot have enough workload distributing information. With the support of token-based load balancing heuristic approach, the downside of this algorithm can be eliminated, thus providing quick, efficient routing decisions. The proxies might not have comprehensive information about the workload of their international state and neighbours; however, they make their specific conclusions about places to transfer the token by producing their particular knowledge base. Thus, the information base used is generated from the tokens received before, and so no unnecessary cost is created.

3. Methodology

The new emerging technology is cloud computing in the academics and industry. In the proposed work, cloud job scheduling and balancing of load strategy were used for allocation of the virtual machine to different user bases requests. In this process, the user base deployed in diverse regions of the world that transmit their request to different datacentres allocated by the third-party cloud server. The cloud analyst tool was used for development and simulation of cloud environment before actual deployment to the real-world application. In the processing of cloud analyst, numerous provision broker policies and policies of load balancing have been used for response of different users.

In the Max–Min Algorithm, the pseudocode is described below:

```

For i = 1 to K // K represents task numbers to be scheduled.
  For j = 1 to N //N represents the virtual machine numbers.
    TCOij = TETij + TRTj // CTij represents Task Completion Time
    // TETij denotes Task Execution Time
    // TRTj denotes Task Ready Time of job i on VM j
  End For
End For
Do until all the unscheduled tasks are exhausted.
  For each unscheduled task
    Find the maximum completion time of the task and virtual machine that
    obtains it.
  End for
  Locate the task tp with the maximum completion time
  Allocate task tp to the virtual machine that gives the maximum completion time
  Delete task tp from the pull of unscheduled tasks
  Update the ready time of the machine that offers the maximum completion time
End for

```

3.1. Proposed algorithm

This paper advocates a new fusion method for load balancing using maximum-minimum and RR algorithm to assign VM to different user base requests called MMRR Load balancing. The proposed hybrid will overcome the problem of maximum and minimum that usually restrict job with least completion time from being executed on time. In contrast, the job with the highest finishing time will be accomplished first. RR that assigns tasks without priority. The proposed flow of work of the new hybrid algorithm is shown in Figure 1.

Figure 1 showing the systematic approach to implement the new algorithm. The data centres and user base are first initializing, then the characteristics of the cloud environment and its bandwidth are set. The new hybrid algorithm is then applied, and the result of the simulation is noted. The downside of maximum and minimum procedure shows the implementation of jobs with the highest execution period may first escalate the overall response period of the system [24]. Eventually, it may cause a slowing down in implementing tasks with smallest execution period, therefore, the reason to work on the new maximum and minimum procedure to lessen the delay in implementing jobs with lowest completion time.

3.2. Performance evaluation parameters

These parameters are used for performance evaluation of the proposed approach.

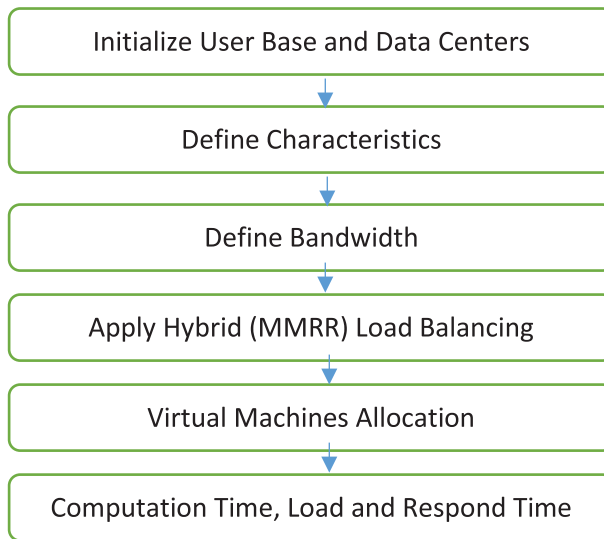


Figure 1. Flow step for the hybrid.

- (i) Over All Response Time: provide information about time use by the CSP to respond to requests.
- (ii) Datacentre Request servicing time: give information on a single data centre for serving requests of different user bases
- (iii) Cost Analysis: provide information about different load balancing approaches using data transmission cost and total VM cost occurred.

The pseudocode for MMRR

```

Minimum → 5
Calculate expected completion time, for all job in the job set
If the job set is empty then
end the program
else
Find a job with least execution time and job with high execution time
    Calculate the difference between high execution time and least execution time
End if
If difference < = least then
Assign job with high execution time with max-min
Else
Assign job with least execution time accordingly
End if
Remove the job from the job set
Update the ready time for the selected resource and expected completion time for all tasks
End
  
```

The pseudocode shows the implementation methods of MMRR; with the assigned task of the shortest execution time process first; the job with the highest expected time will be allocated with RR so that it does not starve the job with minimum execution time tasks. This approach enhances the maximum-minimum algorithm while the RR algorithm will only be used on the task with maximum execution task.

3.3. Configuration of simulation software

The implementation of the algorithms is achieved using Cloud Analysts, a graphical user simulator for the cloud environment. The Eclipse Java EE IDE for Web Developers is used as a container for it. The study adds a new algorithm MMRR to Cloud Analyst source code. In the proposed work, six user bases and two data centres have been defined in the cloud computing environment. They have been allocated in different regions for processing. Different colour regions have identified region boundaries.

Figure 2 displays the distribution of the data centres and the user base for this simulation. We have six user base spread across the continent and two data centres.

Table 1 represents the user base configuration that has been used for the simulation of the cloud computing environment.

Table 2 represents a datacentre configuration that has been used for simulation of the cloud computing environment.

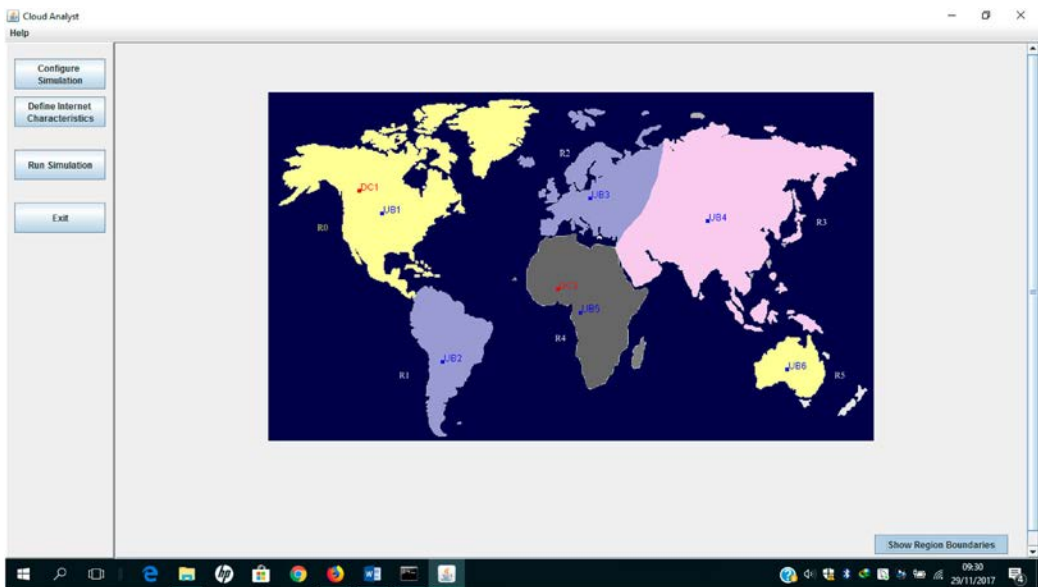


Figure 2. User base and DC allocation in cloud analyst.

Table 1. Configuration for the userbase.

Name	Regions	Request/h	Size	Peak/h	Peak Users	Normal users
UB1	0	600	100	1–3	1500	100
UB2	1	1600	1000	3–5	500	1000
UB3	2	2000	1000	5–7	500	1000
UB4	3	1000	1000	7–9	1600	600
UB5	4	1000	1000	9–11	2000	700
UB6	5	7000	300	11–12	2000	800

Table 2. Configuration of the datacentre.

DC name	R	Cost Vm/h	Memory cost/s	Data transfer/Gb	Speed	H/w units
DC-1	0	0.1	0.5	0.1	10,000	2
DC-2	4	0.1	0.8	0.1	10,000	3

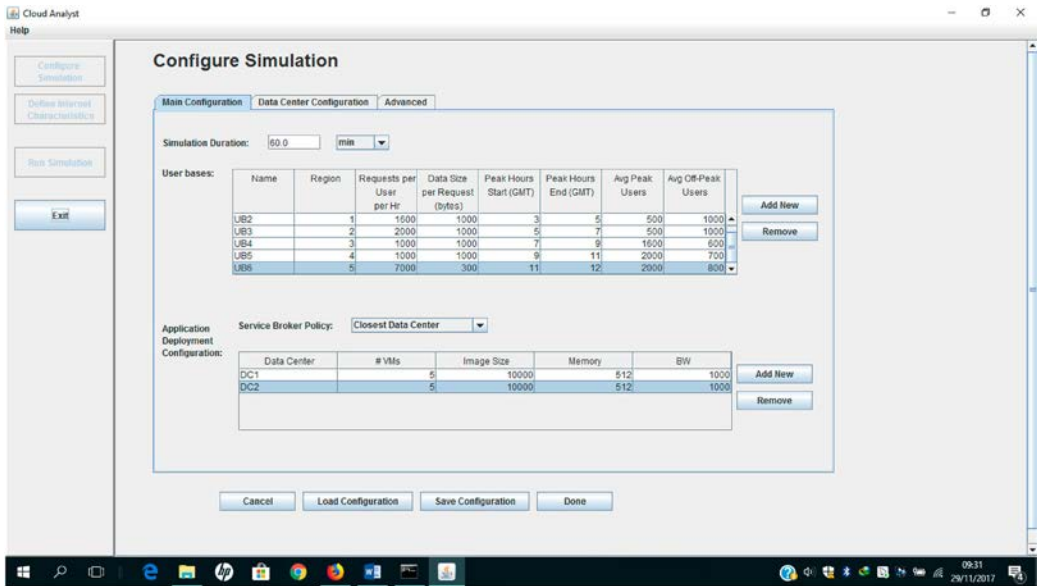


Figure 3. The cloud analyst configuration page.

Table 3. Extra simulation parameter.

Parameter	Value
User grouping factor in user base	1000
Request grouping factor	10
Executable instruction length/request	100
Load balancing policy	Round Robin, Max-Min, Equally Spread Current Execution, Throttle and MMRR
Simulation duration	60.0 min
VM image size	10,000
VM memory	512
VM bandwidth	1000
Data centre architecture	X86
Data centre OS	Linux
Data centre VMM	Xen

Figure 3 display the configuration page of the cloud analyst used. We define specific parameters in the main, data centre and advance configuration.

Table 3 shows the extra parameters set for this simulation.

Figure 4 is the page where we select a different algorithm to be tested.

Based on the simulation parameters, various ways of balancing load policies have been used for the simulation of computing in the cloud. Based on the simulations, numerous parameters were evaluated for performance assessment of the proposed approach.

4. Results and discussion

The outcomes gave the different algorithms of the different scenarios after simulation are displayed in Tables 4 and 5.

Table 4 shows the outcomes of the comparative analysis of the four algorithms show that the new algorithm MMRR performed well in terms of overall response time. The Throttle seems better at data centre processing time and the cost of data transfer from one place to the other, the new algorithm is cheaper to use than any of the different algorithms compared.

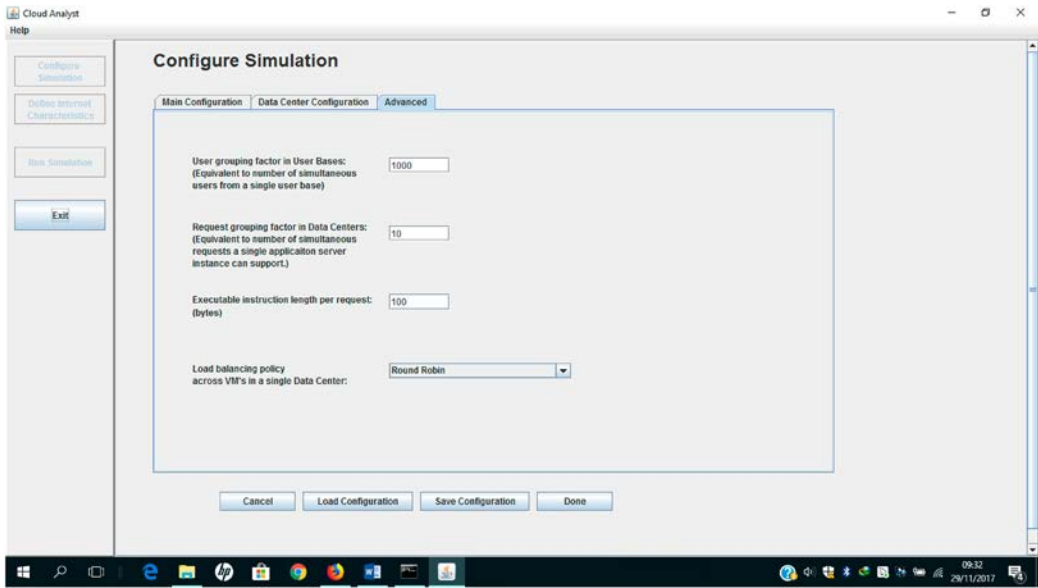


Figure 4. The advanced configuration page.

Table 4. Comparing all the algorithms.

Parameter	Algorithms				
	Round Robin	Max-Min	ESCE	Throttle	MMRR
Overall response time (ms)	228.29	228.20	228.12	227.28	227.26
Processing time of data centre (ms)	1.48	1.40	1.39	0.50	0.57
Cost of processing (\$)	630.20	630.20	630.20	630.20	222.32

Table 5. Comparing all the algorithms using optimize response time policy.

Parameter	Algorithms				
	Round Robin	Max-Min	ESCE	Throttle	MMRR
Total response time (ms)	228.29	228.20	228.12	227.28	227.26
Data centre processing time (ms)	1.48	1.43	1.39	0.50	0.57
DC1 request servicing time (ms)	1.45	1.40	1.36	0.49	0.56
DC2 request servicing time (ms)	1.76	1.68	1.61	0.61	0.63
DC1 transfer cost (\$)	562.36	562.36	562.36	562.36	196.56
DC2 transfer cost (\$)	67.05	67.05	67.05	67.05	23.53

Table 5 shows the result when using the optimized time policy to rerun the simulation; the result gotten is exciting. The new algorithm MMRR still performs better than any other algorithm. The throttle algorithm is much better at data centre processing time, the request servicing time for Data-centres 1 and 2 is fastest using the Throttle and followed by the new algorithm. When comparing the cost of data transfer, for both data centres, MMRR algorithm is far cheaper to use than any other algorithm compared.

The chart in Figure 5 shows the result of the five algorithms that were tested. The new algorithm performed slightly well than the Throttle procedure looking at the overall response time. It also displays that the new algorithm MMRR is better than ESCE, max-min and RR algorithm.

The chart in Figure 6 shows that concerning the processing time for data centre, the Throttle algorithm is better than the new algorithm just proposes slightly.

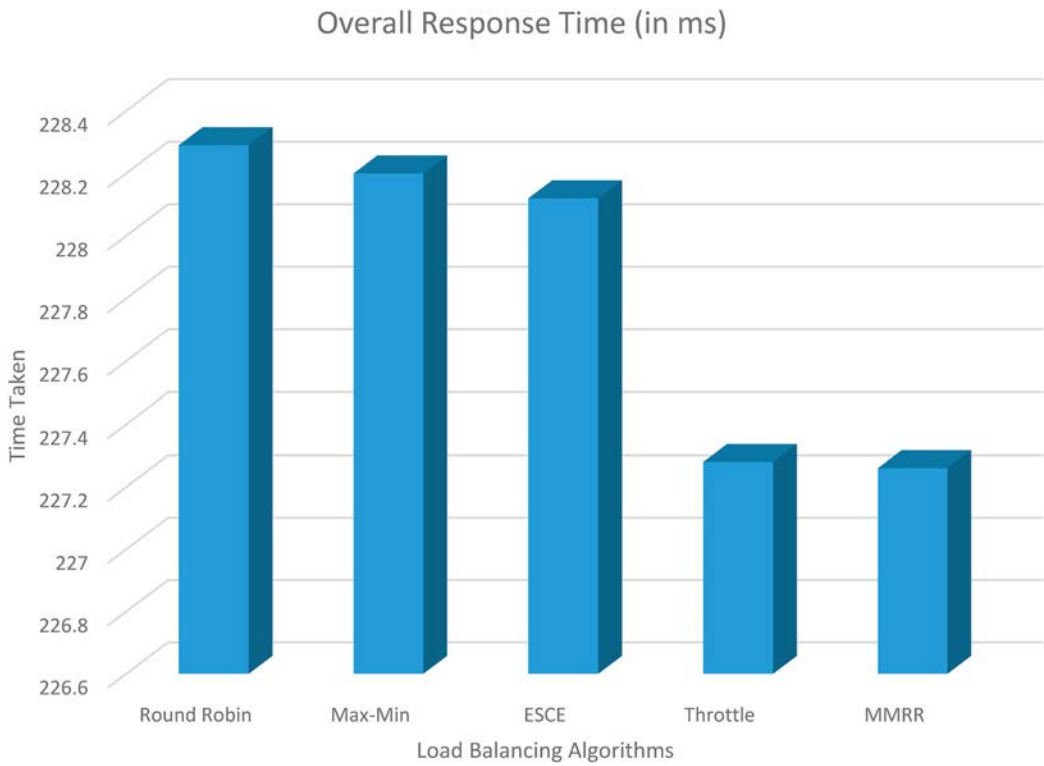


Figure 5. Overall response time of load balancing algorithms.

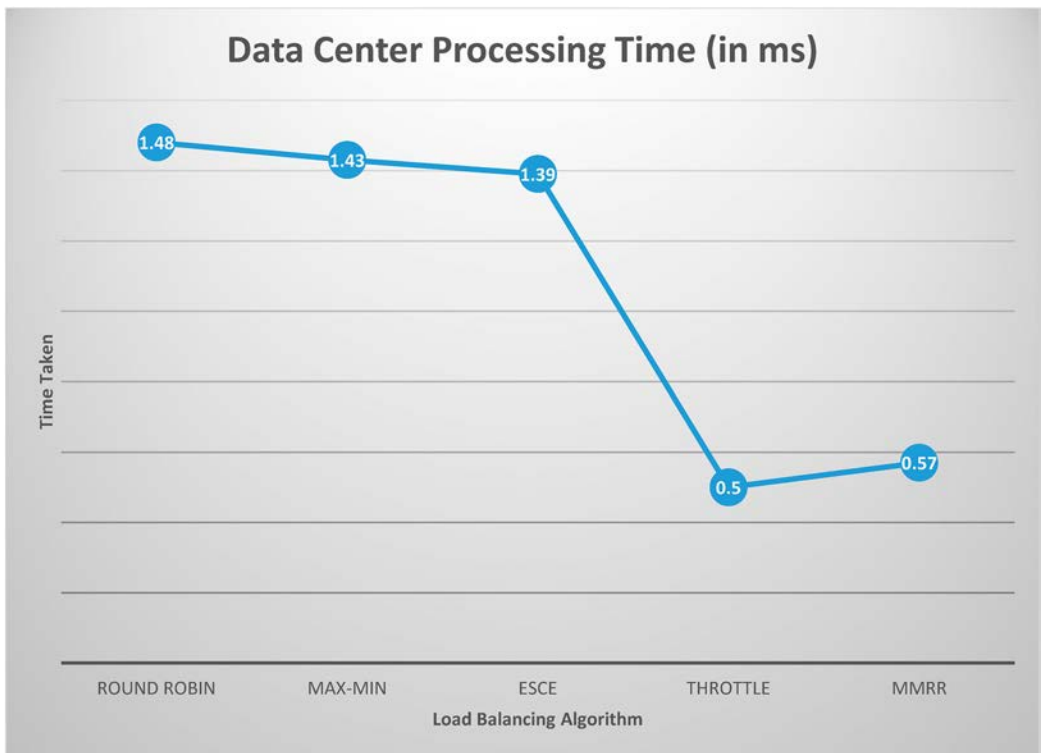


Figure 6. Load balancing algorithms processing time of data centre.

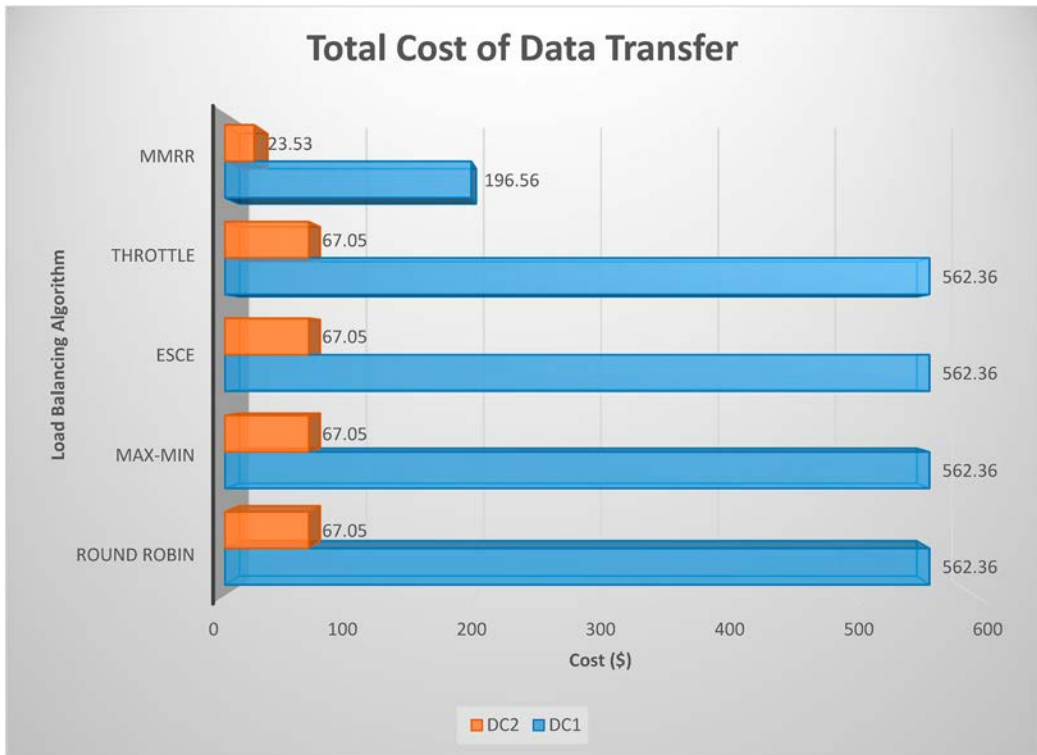


Figure 7. Total cost of data transfer of load balancing algorithms.

Figure 7 chart shows that in terms of the cost of data transfer, the new algorithm MMRR fared better than others.

The results show that the time of the new hybrid MMRR algorithm is better for two data centre and six user bases as compared to ESCE, Round-Robin, Max–Min and Throttled Algorithms. The time taken for processing by the data centre is worthy from both Throttled and MMRR, it was worst for Round Robin. The cost of data transfer was similar for ESCE, Throttled, Max–Min and Round Robin. Interestingly the improved MMRR recorded a low data transfer cost in both data centres, which makes it very cost-effective and cheaper to transfer data across the globe. The overall response time is 227.26 ms and 89.0% cost-effectiveness.

5. Conclusion and future work

In conclusion, the proposed hybrid approach of Round Robin with Maximum Minimum load balancing algorithm brought about significant improvement in the cloud services. The processing time taken by the data centre is good from both Throttled (0.50 ms) and MMRR (0.57 ms), but it was worst for Round Robin (1.48 ms). Out of the different algorithms that were evaluated, MMRR performed well in terms of an overall response time of 227.26 ms and 89% cost-effectiveness.

Thus, the study recommends that MMRR should be adopted in the cloud service to improve people satisfaction. The response time is encouraging; the processing time of the data centre is above average, and the cost of processing a job with the new algorithm is cheaper compare to others.

The role of various service factors produces noteworthy results when VMs are assigned to incoming tasks in a cloud environment. This study would be expanded using different quality of service constraints like cost, throughput and delay for various routing policies in future. More data centres, user base and more algorithms will be used to find out the best algorithm that will give optimum result in the cloud environment.

Disclosure statement

No potential conflict of interest was reported by the authors.

ORCID

Abiodun Kazeem Moses  <http://orcid.org/0000-0002-3049-1184>

Awotunde Joseph Bamidele  <http://orcid.org/0000-0002-1020-4432>

Ogundokun Roseline Oluwaseun  <http://orcid.org/0000-0002-2592-2824>

Sanjay Misra  <http://orcid.org/0000-0002-3556-9331>

Adeniyi Abidemi Emmanuel  <http://orcid.org/0000-0002-2728-0116>

References

- [1] R.L. Adelaar, *Cloud computing technology in manufacturing firms-Identifying adoption factors, challenges, and corresponding solutions*, Master's thesis, 2020.
- [2] T. Ahmed, and Y. Singh, *Analytic Study of Load Balancing Techniques Using Tool Cloud Analyst*, International Journal of Engineering Research and Applications, New Delhi, 2012. Vol. 2. ISSN: 2248.
- [3] D.F. Altayeb, and F.A. Mustafa, *Analysis of load balancing algorithms implementation on cloud computing environment*, Int. J. Innov. Res. Adv. Eng. 6(2) (2016), pp. 1–32.
- [4] V. Arulkumar, and N. Bhalaji, *Performance analysis of nature inspired load balancing algorithm in cloud environment*, J. Ambient. Intell. Humaniz. Comput. (2020), pp. 1–8.
- [5] M.G. Avram, *Advantages and challenges of adopting cloud computing from an enterprise perspective*, Procedia Technol. 12 (2014), pp. 529–534.
- [6] K. Bala. *A review of the load balancing techniques at cloud server*. 1, International Journal of Advances in Computer Science and Communication Engineering, Chandigarh, March 2014, International Journal of Advances in Computer Science and Communication Engineering, Vol. 2. ISSN 2347-6788.
- [7] R. Buyya, and K. Sukumar, *Platforms for building and deploying applications for cloud computing*. ArXiv Preprint arXiv 1104 (2011), pp. 4379.
- [8] S.G. Domanal, and G.R. Mohana Reddy, *Load balancing in cloud computing using modified throttled algorithm*, Cloud Computing in Emerging Markets (CCEM), 2013 IEEE International Conference on. IEEE, 2013.
- [9] K. Dubey, M. Kumar, and S.C. Sharma, *Modified HEFT algorithm for task scheduling in cloud environment*, Procedia Comput. Sci. 125 (2018), pp. 725–732.
- [10] S. Ghanbari, and M. Othman, *A priority based job scheduling algorithm in cloud computing*, Procedia Eng. 50 (2012), pp. 778–785.
- [11] H. Gibet Tani, C. El Amrani. *Smarter round robin scheduling algorithm for cloud computing and big data*, J. Data Min. Digital Humanities, Special Issue on Sci. Technol. Strategic Intell. jdmhd:3104 (2016).
- [12] T.C. Hung, L.N. Hieu, P.T. Hy, and N.X. Phi, *MMSIA: improved max-min scheduling algorithm for load balancing on cloud computing*, In Proceedings of the 3rd International Conference on Machine Learning and Soft Computing (pp. 60–64), 2019, January.
- [13] B.T. Khiet, N.T. Nguyet Que, H.D. Hung, P.T. Vu, and T.C. Hung, *A Fair VM Allocation for Cloud Computing based on Game Theory*, Proceedings of the 10th National Conference on Fundamental and Applied Information Technology Research (FAIR'10), Da Nang, Viet Nam, 2017.
- [14] T. Kokilavani, and G. Amalarethnam, *Load balanced Min-Min algorithm or static Meta-task scheduling in grid computing*, Int. J. Comput. Appl. 20(2) (2011), pp. 0975–8887.
- [15] X. Li, Y. Mao, X. Xiao, and Y. Zhuang, *An improved max-min taskscheduling algorithm for elastic cloud*, In: IEEE international symposium on computer, consumer and control, pp. 340–343, 2014.
- [16] D. MacVittie, *Intro to Load Balancing for Developers – The Algorithms*. DevCentral. [Online] 31 March 2009. <https://devcentral.f5.com/articles/intro-to-load-balancing-for-developers-ndash-the-algorithms#.U5k6Xnbm40E>.
- [17] K. Mahajan, A. Makroo, and D. Dahiya, *Round robin with server affinity: a VM load balancing algorithm for cloud based infrastructure*, J. Inf. Process. Sys. 9(3) (2013), pp. 379–339.
- [18] S.H. Mahalle, R.P. Kaveri, V. Chavan, *Load balancing on cloud datacenters*, Int. J. Adv. Res. Comput. Sci. Softw. Eng., 3 (January 2013), pp. 1–4.
- [19] S.K. Mishra, B. Sahoo, and P.P. Parida, *Load balancing in cloud computing: a big picture*, J. King Saud University-Comput. Inf. Sci. 32(2) (2020), pp. 149–158.
- [20] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, & R. Rastogi, *Load balancing of nodes in cloud using ant colony optimization*, Computer Modelling and Simulation (UKSim), 2012 UKSim 14th International Conference on. IEEE, 2012.

- [21] S. Patel, H. Patel, and N. Patel, *Dynamic load balancing techniques for improving performance in cloud computing*, Int. J. Comput. Appl. 138(3) (2016), pp. 1–5.
- [22] G.A.E.N.A. Said, *Nature inspired algorithms in cloud computing: A survey*, Int. J. Intell. Inf. Syst. 5(5) (2016), pp. 60–64.
- [23] P. Samal, and P. Mishra, *Analysis of variants in round robin algorithms for load balancing in cloud computing*, Int. J. Comput. Sci. Inf. Technol. 4(3) (2013), pp. 416–419.
- [24] B. Santhosh, and D. Manjiaiah, *An improved task scheduling algorithm based on max min for cloud computing*, 2014.
- [25] S. Sidana, N. Tiwari, A. Gupta, and I.S. Kushwaha, *NBST algorithm: A load balancing algorithm in cloud computing*, In 2016 International Conference on Computing, Communication and Automation (ICCCA), IEEE, 2016, April, pp. 1178–1181.
- [26] S.P. Singh, A. Sharma, and R. Kumar, *Analysis of load balancing algorithms using cloud analyst*, Int. J. Grid. Distrib. Comput. 9(9) (2016), pp. 11–24.
- [27] A.N. Singh, M. Hemalatha, *An approach on semi distributed load balancing algorithm for cloud computing systems*, Int. J. Comput. Appl. 56(12) (2012), pp. 5–10.
- [28] A. Thomas, G. Krishnalal, and V.J. Raj, *Credit based scheduling algorithm in cloud computing environment*, Procedia Comput. Sci. 46 (2015), pp. 913–920.
- [29] Y.F. Wen and C.L. Chang, 2014. *Load balancing job assignment for cluster-based cloud computing*, Sixth International Conference on Ubiquitous and Future Networks (ICUFN), Shanghai, pp. 199–204.
- [30] B. Wickremasinghe, R.N. Calheiros, and R. Buyya, *Cloudanalyst: A cloudsim-based visual modeller for analyzing cloud computing environments and applications*, In: IEEE international conference on advanced information networking and applications, pp. 446–452, 2010.
- [31] C. Xiong, L. Feng, and L. Chen, *A new task scheduling algorithm based on an improved genetic algorithm in a cloud computing environment*, Adv. Inf. Sci. Serv. Sci. 5(3) (2013), pp. 32.