An Advanced Algorithm for Load Balancing in Cloud Computing using MEMA Technique

Saher Manaseer, Metib Alzghoul, Mazen Mohmad

***Abstract*: Recently Cloud computing has become one of the most significant technologies for its impact in every aspect in networking and new technologies such as security, capacity, quality of service, cost, and accessibility etc. One of the major challenges in networks is Load Balancing. Many algorithms were proposed to solve the problem. Some considered static variables while others considered dynamic ones. In this paper static variables techniques are used with the new proposed algorithm "MEMA Technique". In the proposed algorithm few steps are added to the weighted round robin (WRR). Moreover, a comparison of performance between the (WRR) and MEMA is presented.**

***Index Terms*: Cloud Computing, Load Balancing, Static Algorithms, Weighted Round Robin (WRR) MEMA Technique.**

I. INTRODUCTION

Working in parallel on software and hardware, and interaction everywhere has made the cloud computing the world major trend recently [1][2]. Moreover, the features like on-demand services, utility based model, ease of use and accessible, pay per use model, etc. [3]. The service model of cloud computing is classified into three basic categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). In SaaS, the software vendor provides the software such as Google Driver, Gmail. Where as in PaaS, a platform provided for writing the code of the program, as Google app engine. However, in the IaaS, the hardware resources are recommended for user support to construct the framework, just like cloud server [4].

More Important, cloud computing provides services such as storing and accessing files which are located at different locations. As cloud users increases daily, data requests increase as well, which means more challenges such as hard NP problem (nondeterministic polynomial time) [1][5].

Another significant challenge that cloud computing faces is load balancing, which represents overloaded and under-loaded nodes in cloud networking. Since load control and management can achieve fairness for network and better service efficient algorithms are needed for this issue [1][21].

In static Load balancing algorithms, the load is calculated at the compilation time and once the load is given, modifications cannot be done. In dynamic load balancing algorithms, the determined load is calculated at the runtime with no previous information required. And because of huge number of requests, dynamic load balancing algorithms are suitable for many cloud computing environments because it focuses on response time minimization and throughput of the overall system, while the static load balancing algorithms focuses on minimizing the response time without focusing on overall system throughput [5]. This research paper mainly focuses on challenges in cloud computing through static load balancing algorithm and suggested an efficient static algorithm. It uses the idea of weighted round robin algorithm (WRR) with MEMA technique. This approach is expected to reduce the response time, minimize the amount of wasting data, and maximize the fairness concept as well as it gives better performance.

II. RELATED WORKS

A. Weighted Round Robin in Honeybee Inspired load balancing approach

In [5], a hybrid algorithm of weight round robin (WRR) and honeybee inspired load balancing approach was introduced. The algorithm finds and examines the overloaded and under loaded virtual servers' capacity. Weights are entered based on its capacity, if the load of the server is less than the capacity entered no more checks are done, otherwise check the overloaded servers using the honeybee inspired load balancing algorithm to allocate high priority requests to the under-loaded servers (virtual machines) respectively. As a result, this algorithm presents a better response time as well as data center processing time. Figure 2 shows the flow diagram of this hybrid algorithm.

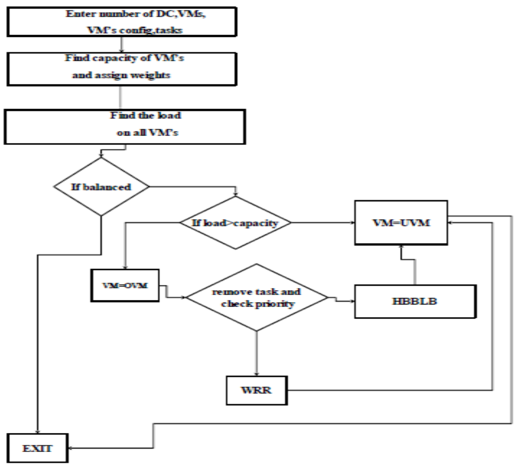

Enter number of DC,VMs, VM's config,tasks
Find capacity of VM's and assign weights
Find the load on all VM's
If balanced
If load>capacity
VM=UVM
VM=OVM
remove task and check priority
HBBLB
WRR
EXIT

Figure 1: The Flow Diagram of the Hybrid Algorithm

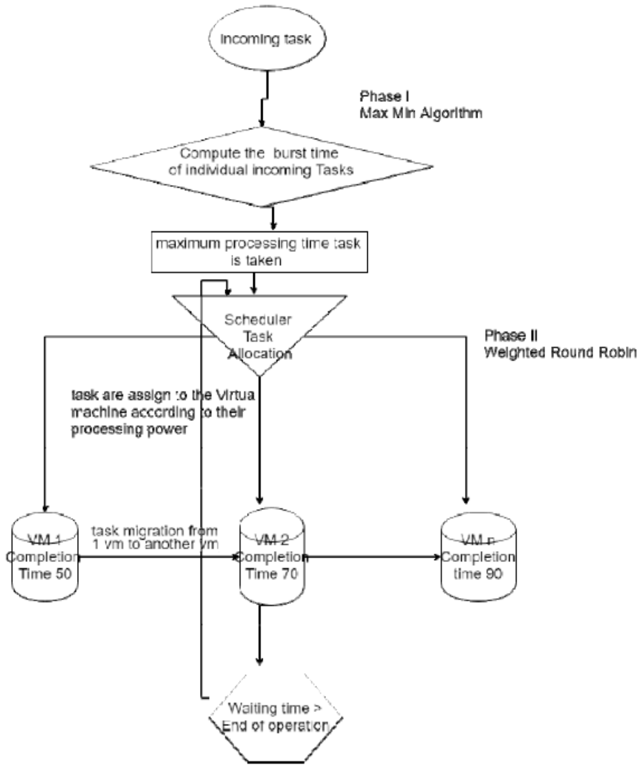
Manuscript published on 30 January 2019.
***Correspondence Author(s)**
Dr. Saher Manaseer, King Abdullah II school for Information Technology, Computer Science Department, The University of Jordan, Amman, 11942, Queen Rania Street, Jordan.
Metib Ali Ahmad Alzghoul, King Abdullah II school for Information Technology, Computer Science Department, The University of Jordan, Amman, 11942, Queen Rania Street, Jordan.
Mazen Jameel Shawkat Mohmad, King Abdullah II school for Information Technology, Computer Science Department, The University of Jordan, Amman, 11942, Queen Rania Street, Jordan.

In [6] some of load balancing in cloud computing algorithms were studied such as round robin approach, stochastic hill climbing algorithm, join idle queue algorithm, ant colony optimization, genetic algorithm.

B. WMaxMin Algorithm

Two algorithms were combined in [7]; max min and weighted round robin (WRR) algorithm. First, maximum completion time, burst time and max processing time are sorted, based on the list those values are allocated to the servers of high available weight. where the processing capabilities of the servers are counted by the minimum time needed to execute the task.

Figure 3 shows how process in this algorithm is divided into two phases, the first one calculates the burst time of the process using max min algorithm, followed by discovering the maximum completion time and burst time for requests. In phase two, the results of phase one are merged and entered in the scheduler that allocates the requests to the servers which are sorted in decreasingly order of the maximum completion and burst time. As a result, this technique is appropriate for all environments except dynamic and hieratical environment because it cannot interact with different environment.



Incoming task
Phase I
Max Min Algorithm
Compute the burst time of individual incoming Tasks
maximum processing time task is taken
Scheduler Task Allocation
Phase II
Weighted Round Robin
task are assign to the Virtual machine according to their processing power
VM 1 Completion Time 50
task migration from 1 vm to another vm
VM 2 Completion Time 70
VM n Completion time 90
Waiting time > End of operation

Figure 2: Phases of the Process in WMaxMin algorithm

C. Fuzzy Round Robin (FRR)

A new load balancing algorithm based on fuzzy technique was proposed by [8], which represents a combination between round robin (RR) algorithm with fuzzy good judgment technique to improve response time and processing time. Rules of balancing are assigned before the server starts processing. Also, requests are organized depending on processor speed and the amount of load per server. This technique can perform smooth to apprehend, flexible, to approximate capabilities etc.

The pseudocode of the proposed algorithm as shown below in figure3.

```
  Algorithm
Begin
   Connect_to_resource()
   L1
   If (useful resource found)
          Begin
          Calculate connection_string()
          Select fuzzy_connection()
          Return resource to requester End
   Else if
          Begin If (Anymore useful resource to be had)
          Choose_next_resource()
           Go to L1
   Else
          Exit
    End
  End
```

Figure 3: Fuzzy Round Robin (FRR)

III. METHODOLOGY

As discussed later, various algorithms attempts to balance the load in cloud computing. This paper takes into consideration argent messages in parallel with fairness in distributions as much as possible.

Depending on the idea of weighted round robin (WRR) that have been discussed in previous section, the authors of this paper introduces a technique which is divided into two parts, the first one is determined by priority messages where the original balancer is divided into normal request balancer and priority request balancer as shown in figure 5. On the other hand, distribute requests among servers in which maximum weight obtains maximum number of requests. This technique reduces wasted requests, minimize response time, distribute requests more fairly through servers and then better performance is expected to be obtained.

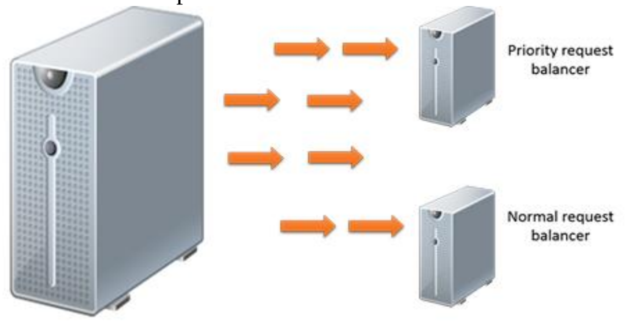


Figure 4: Balancer Parts

In cloud computing infrastructure, as shown in figure 6, clients send their requests to the server (balancer). In the server (balancer), the received requests categorized into argent requests and normal requests in which argent requests are sent first, normal requests are sent in the second place, knowing that all argent requests are scheduled in the same priority level.

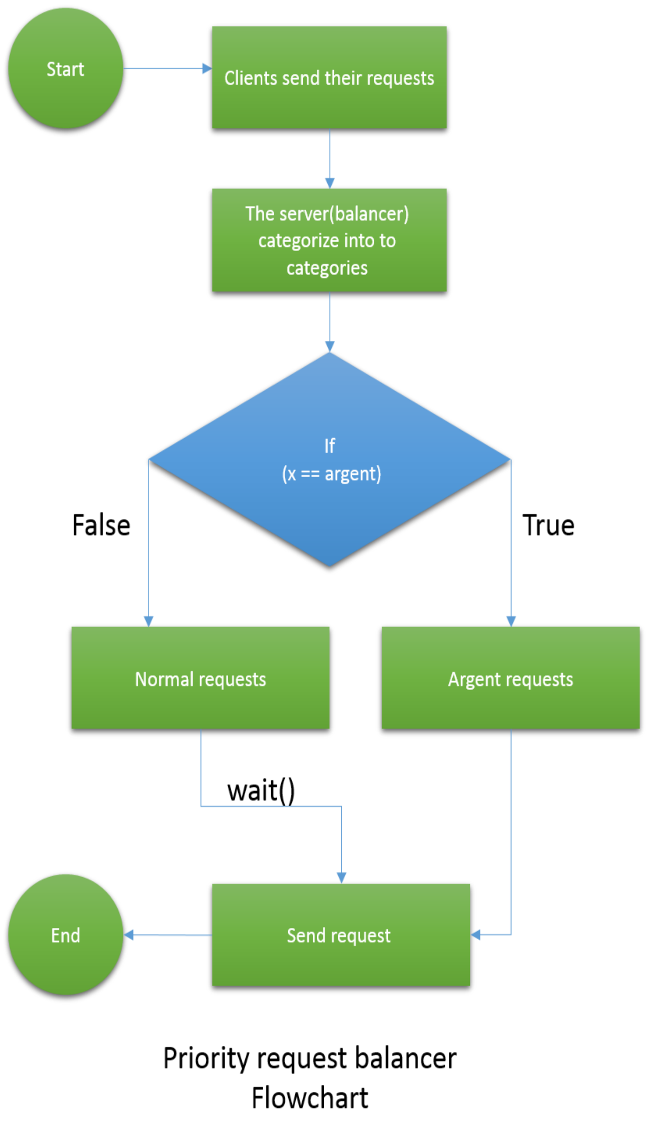

Start
Clients send their requests
The server(balancer) categorize into to categories
If (x == argent)
False
True
Normal requests
Argent requests
wait()
Send request
End
Priority request balancer Flowchart

Figure 5: Priority Request Balancer

On the other hand, as shown in figure 7, using the idea of weighted round robin (WRR). First, each node sends its weight (capacity) to the server (balancer) that will be used to determine the number of request for each. Second, the server (balancer) starts sending requests to the nodes provided that each odd step (1, 3, 5 and so on) sends the maximum weight (capacity) of requests that the node successfully accepts as shown in formula number 1, moreover, each even step (2, 4, 6 and so on) sends the maximum weight (capacity) of requests divided by 2 (ceiling function) as shown in formula number 2. This technique called "MEMA technique" that could improve server (node) performance by reducing the number of sending request per server (node), minimizing the response time, minimizing the bandwidth overload, also it minimizes the number of discarded requests in parallel as well as improving the power of efficiency in request distribution among servers (nodes). Furthermore, it maximizes the probability of resource utilization among servers (nodes).

$$\text{New weight (odd step)} = \text{full weight (capacity)}$$

$$New\ weight(even\ step) = \lceil Old\ weight(capacity)/2 \rceil$$

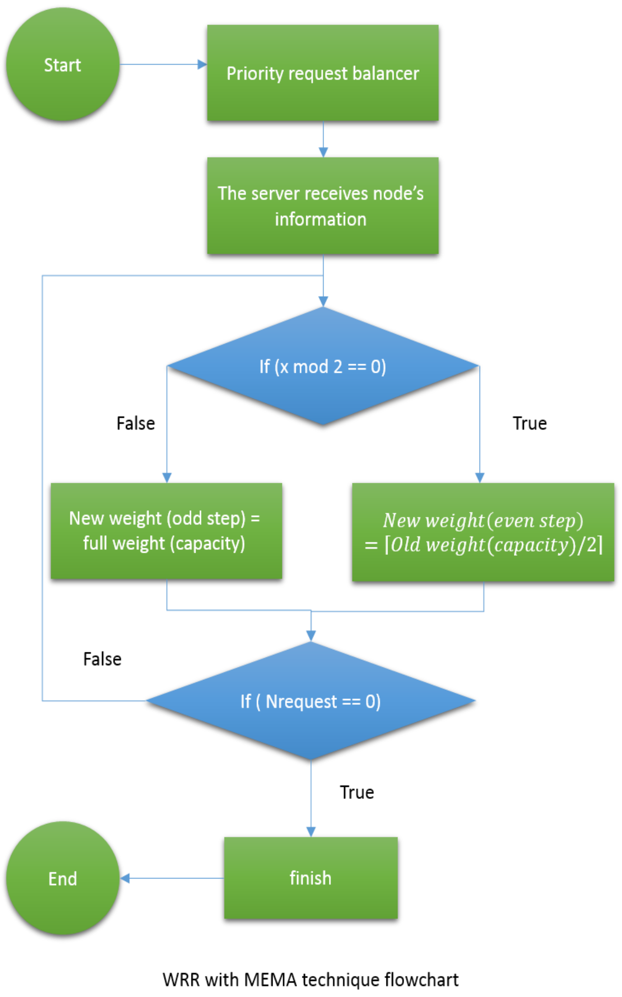
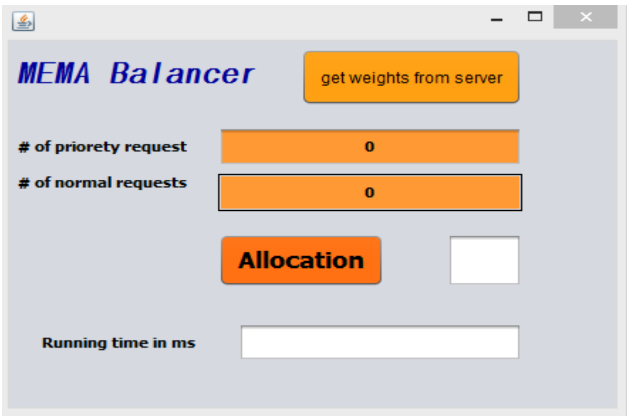

Start
Priority request balancer
The server receives node's information
If (x mod 2 == 0)
False
True
New weight (odd step) = full weight (capacity)
New weight(even step) = [Old weight(capacity)/2]
False
If (Nrequest == 0)
True
finish
End
WRR with MEMA technique flowchart

Figure 6: WRR with MEMA

IV. RESULTS

The proposed technique that previously discussed implemented in core i3, 2.4 GHz with 6 GB RAM and windows 8.1 OS laptop. Furthermore, Java programming language used on NetBeans IDE 7.4 version. This experiment worked by using two servers (nodes) shown in figure 8, 9, and 10. Also, the authors of this paper used socket library on Java in which to establish a small network. By this, client server model has been accomplished.


MEMA Balancer

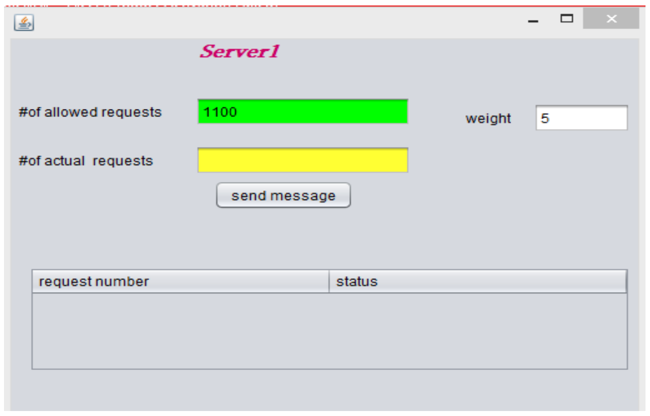
get weights from server

# of priorety request	0
# of normal requests	0

Allocation

Running time in ms

Figure 7: Experiment Balancer

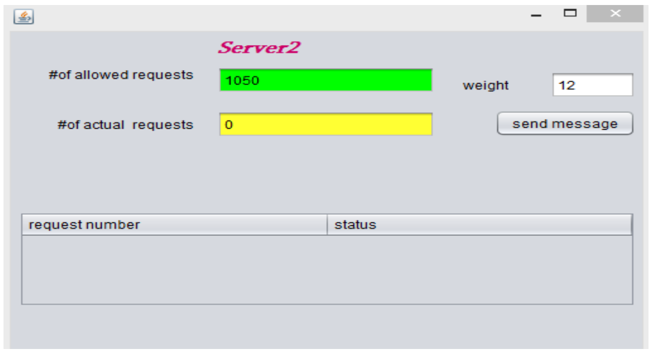

Server1

#of allowed requests	1100	weight	5
#of actual requests			

send message

request number	status

Figure 8: Experiment Server 1


Server2

#of allowed requests	1050	weight	12
#of actual requests	0	send message	

request number	status

Figure 9: Experiment Server 2

Pseudo code of the proposed technique is written as follows:

```
➢ Start

➢ Receive all requests from different clients
➢       n= number of requests which received from clients
➢           Distribute Balancer into two virtual balancer  priority_balancer,
  normal_balancer

➢                   priority_length = length of priority_balancer
➢                    normal_length=length of normal_balancer

➢               for i = 0 to n
➢                   IF (request[i] is priority) then
➢                       insert it into priority_balancer
➢                   else
➢                       insert it into normal_balancer
➢                   end if
➢               end for I

➢           specify weight for each server into servers_weights array and
  server_address into servers_names array
➢              sort server_names based on its weights descending

➢          j=0
➢          temp1=0
➢          while(j<priority_length) do
➢              if (temp==0) then
➢               for m=0 to length of servers_names array
➢                   for L=0 to length of servers_weights
➢                           send request[j] from priority_balancer to server_names[m]
➢                            j=j+1
➢                   end for L
➢               end for m
➢                temp1=1
➢              else
➢             for m=0 to length of servers_names array
➢                   for L=0 to length of servers_weights/2
➢                           send request[j] from priority_balancer to server_names[m]
➢                            j=j+1
➢                   end for L
➢                end for m
➢                temp1=0
➢             End if
➢          end while
➢       a=0
➢        temp2=0
➢         while(a<normal_length) do
➢             if(temp2==0) then
➢               for w=0 to length of servers_names array
➢                   for r=0 to length of servers_weights
➢                           send request[a] from normal_balancer to server_names[w]
➢                            a=a+1
➢                    end for r
➢               end for w
➢            temp2=1
➢             else
➢             for w=0 to length of servers_names array
➢                   for r=0 to length of servers_weights/2
➢                           send request[a] from normal_balancer to server_names[w]
➢                            a=a+1
➢                    end for r
➢               end for w
➢            temp2=0
➢             End if
➢         end while

➢ End
```

V. DISCUSSION

Results shows that MEMA technique improves the ability to balance the load among servers (nodes) with more fairness requirements achieved, although response time for argent requests have been reduced but it may increase its running time.

As described in table 1, a comparison between weighted round robin (WRR) without MEMA technique and weighted round robin (WRR) with MEMA technique introduced. This comparison shows the differences in running time through its number of requests. In case of small number of requests, both algorithms perform well. However, as shown in data chart comparison 1.1, weighted round robin without MEMA technique performs better in case number of requests increased. But in case of huge number of requests, both algorithms approximately perform the same. Moreover, as shown in data chart comparison 1.2, the authors observed that in extra huge number of requests, the weighted round robin (WRR) with MEMA technique will perform better.

As known, weighted round robin (WRR) works immediately after the servers (nodes) send their weight (capacity). By this way, the probability of discarded requests will be high. In contrast, by using weighted round robin (WRR) based on MEMA technique, the probability of discarded requests will be decreased as well as the reliability will be increased which leads to maximize the running time for it.

Although weighted round robin (WRR) without MEMA technique performs better than weighted round robin with MEMA technique in case of running time. But it does not serve argent requests first. Also MEMA technique reduces discarded requests in case of overloaded servers (nodes) as well as maintains the fairness requirements in case of overloaded and under-loaded servers (nodes). As a conclusion, MEMA technique guides the weighted round robin (WRR) to be more organized and efficiently working.

Table1: A comparison between WRR without MEMA VS WRR using MEMA

Number of requests	Weighted round robin (WRR) without MEMA technique/ ms	Weighted round robin (WRR) with MEMA technique/ ms
25	2	12
50	3	16
100	7	29
200	13	61
400	25	104
800	47	147
1600	124	309
3200	234	493
6400	455	1027
12800	967	1391
25600	1715	2658
100000	5891	9385
500000	31063	46039
1000000	60741	94339

Number of Requests	Weighted round robin (WRR) without MEMA technique/ ms	Weighted round robin (WRR) with MEMA technique/ ms
25	2	12
50	3	16
100	7	29
200	13	61
400	25	104
800	47	147
1600	124	309
3200	234	493
6400	455	1027
12800	967	1391
25600	1715	2658
100000	5891	9385
500000	31063	46039
1000000	60741	94339

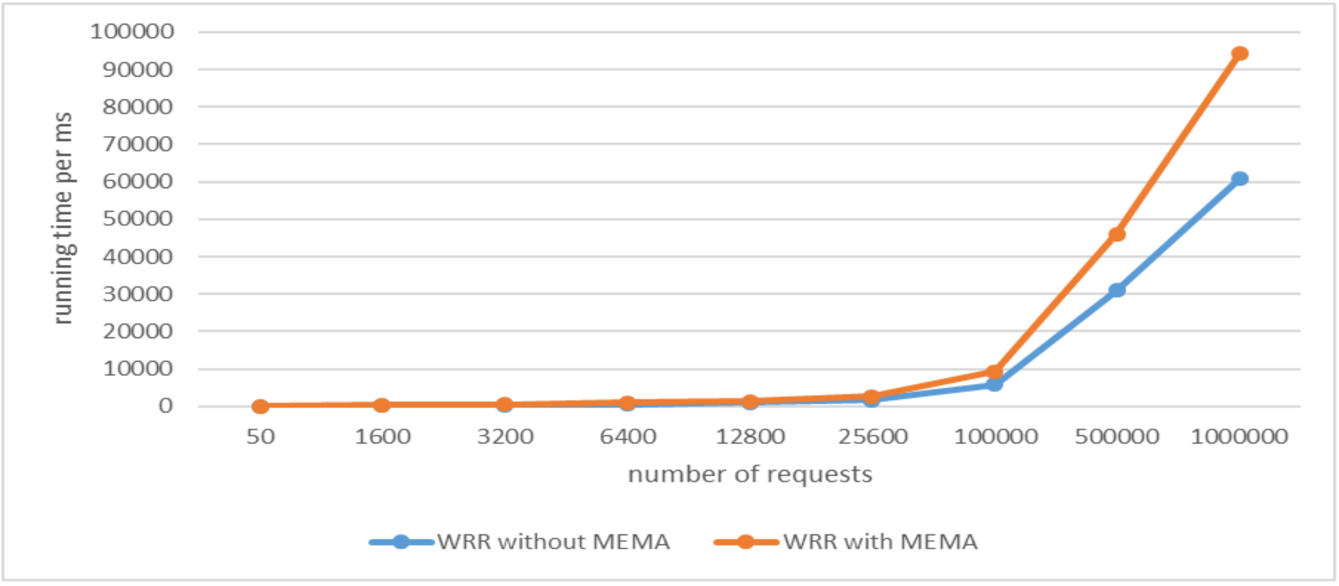


Figure 10: A comparison between WRR without MEMA VS WRR using MEMA

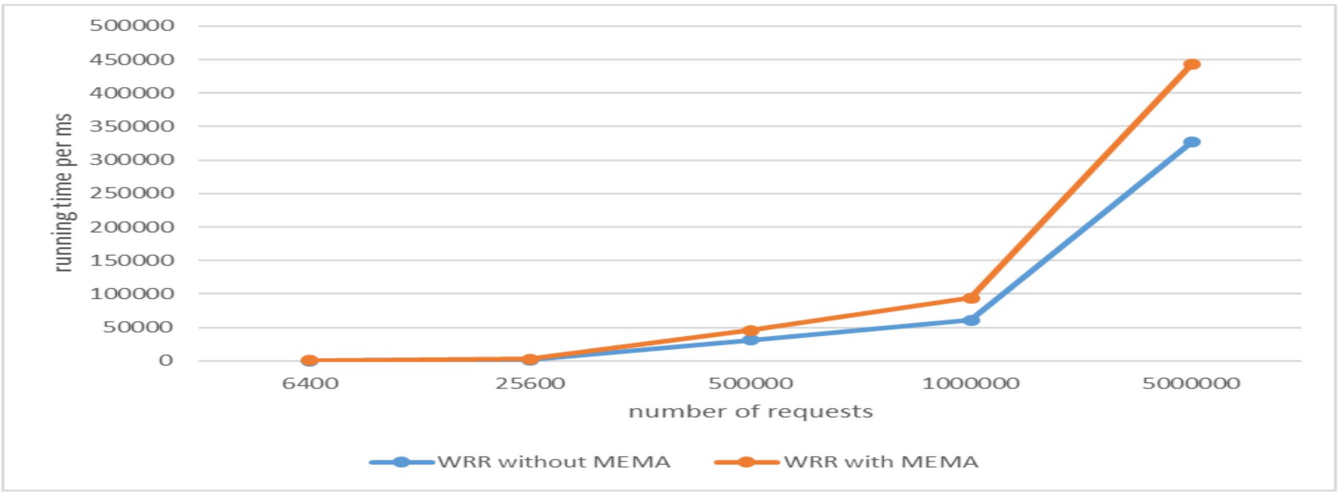


Figure 11: A Comparison Between WRR without MEMA VS WRR using MEMA using Larger Number of Request

VI. FUTURE WORK

Cloud computing technology growing up rapidly. Which guides the technology to face many challenges such as load balancing, cost, security, complexity of building a private cloud etc. load balancing, as its one of the challenges that faces the technology, will negatively affects the performance. And so on, various algorithms with different techniques are needed.

As a future work, the proposed formula works through two sides in which the odd step sends the weight (capacity) as it is, and the second side, in the even step, it sends the weight (capacity) divided by 2. Hence, a third side will be proposed working in parallel with threads to minimize discarded request that could occur when the server (node) receives more than its capability (weight). As well as, it maximizes the reliability with less response time for argent requests.

VII. CONCLUSION

As discussed, various algorithms attempt to interact with cloud computing challenges. In this paper, diverse static algorithms are highlighted and discussed to balance the load among servers (nodes). Based on the idea of weighted round robin (WRR), a new technique is introduced with an implementation called “MEMA technique”. Basically, this technique aims to improve the ability of efficient distribution through servers (nodes), minimal response time for urgent requests, and the probability of discarded requests have been minimized, which leads to increase the reliability with better performance.

REFERENCES

1. P. M. B. M.Padmavathi, "Dynamic And Elasticity ACO Load Balancing Algorithm for Cloud Computing," International Conference on Intelligent Computing and Control Systems ICICCS 2017, 2017.
2. P. U. T. Prof. Satvik Khara, "A Novel Approach for Enhancing Selection of Load Balancing Algorithms Dynamically in Cloud Computing," International Conference on Computer, Communications and Electronics (Comptelix) Manipal University Jaipur, Malaviya National Institute of Technology Jaipur & IRISWORLD, 2017.
3. M. R. D. D. R. S. Khusboo K. Patel, "Dynamic Priority Based Load Balancing Technique For VM Placement In Cloud Computing," Proceedings of the IEEE 2017 International Conference on Computing Methodologies and Communication (ICCMC), 2017.
4. H.-C. H. W.-C. T. M.-C. K. Mao-Lun Chiang, "An Improved Task Scheduling and Load Balancing Algorithm under the Heterogeneous Cloud Computing Network," IEEE 8th International Conference on Awareness Science and Technology (iCAST 2017), 2017.
5. M. S. G. a. J. P. Nithin Das K.C, "Incorporating Weighted Round Robin in Honeybee Algorithm for Enhanced Load Balancing in Cloud Environment," International Conference on Communication and Signal Processing, April 6-8, 2017, India, 2017.
6. K. R. K. D. Kripa Sekaran, "SIQ Algorithm for Efficient Load Balancing In Cloud," International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), 2017.
7. P. B. Bharat Khatavkar, "Efficient WMaxMin Static Algorithm For Load Balancing In Cloud Computation," International Conference on Innovations in Power and Advanced Computing Technologies [i-PACT2017], 2017.
8. B. R. Venkateshwarlu Velde, "An Advanced Algorithm for Load Balancing in Cloud Computing using Fuzzy Technique," International Conference on Intelligent Computing and Control Systems ICICCS 2017, 2017.
9. G. P. R.Kanniga Devi, "A Graph-based Mathematical Model for an Efficient Load Balancing and Fault tolerance in Cloud Computing," Second International Conference on Recent Trends and Challenges in Computational Models, 2017.
10. M. S. H. Tahira Islam, "A Performance Comparison of Load Balancing Algorithms for Cloud Computing," 978-1-5386-3148-5/17/$31.00 © 2017 IEEE, 2017.
11. D. V. R. U. Sridevi S, "A Survey of Soft Computing Techniques Applied in Cloud Load Balancing," IEEE Eighth International Conference on Advanced Computing (ICoAC), 2016.
12. S. M. S. N. I. P. Jananta Permata Putra1, "Live Migration Based on Cloud Computing to Increase Load Balancing," International Seminar on Intelligent Technology and Its Application, 2017.
13. R. G. Ashish Gupta, "Load Balancing Based Task Scheduling with ACO in Cloud Computing," international conference on computer and applications (ICCA), 2017.
14. O. H. O. H. Sofiane Mounine Hemam, "Load Balancing Between Nodes in a Volunteer Cloud Computing by Taking Into Consideration the Number of Cloud Services Replicas," IEEE, 2017.
15. D. S. P. D. N. J. D. K. K. D. D. Jaimeel M Shah, "Load Balancing in cloud computing: Methodological Survey on different types of algorithm," International Conference on Trends in Electronics and Informatics ICEI 2017, 2017.
16. N. B. MONIKA LAGWAL, "Load balancing in Cloud Computing using Genetic Algorithm," International Conference on Intelligent Computing and Control Systems ICICCS 2017, 2017.
17. S. P. N. A. A. Mrs. Shruti Tripathi, "MODIFIED OPTIMAL ALGORITHM FOR LOAD BALANCING IN CLOUD COMPUTING," International Conference on Computing, Communication and Automation (ICCCA2017), 2017.
18. E. A. R. Muhammad Sohaib Shakir1, "Performance Comparison of Load Balancing Algorithms using Cloud Analyst in Cloud Computing.," IEEE, 2017.
19. P. C. Umadevi.K.S, "Predictive Load Balancing Algorithm for Cloud Computing," IEEE, 2017.
20. M. A. K. B. S. D. P. a. M. S. O. F. o. I. a. K. H. Sambit Kumar Mishra, "Time Efficient Dynamic Threshold-based load balancing technique for cloud computing," IEEE, 2017.
21. Manaseer, S. and , Alawneh, A. “Fairness based on backoff algorithms for enhancing MANETs performance: a comprehensive review”, Journal of Mathematical and Computational Science, Vol 8, No.3, Pp: 394-420.

Dr. Manaseer has a PhD in Computer Science from the Department of Computing Science at the University of Glasgow. His main area of research is Computer Networks and Embedded Systems. Currently, Dr. Manaseer is an active researcher in the field of Mobile Ad Hoc Networks. More specifically, his research on MANETs is focused on MAC layer protocols. Before obtaining his PhD, He got his Masters in in Software Engineering from the University of Jordan.