ReT-ELBa: A Novel Algorithm for Efficient Load Balancing in Cloud Computing

ABSTRACT

This paper presents a novel load-balancing algorithm for balancing load on the cloud environment which we call Response Time Efficient Load Balancer (ReT-ELBa). Load balancing involves a dynamic and equal shifting of workloads among processors or Virtual Machines to achieve better resource utilisation. The study gives an insight into the design, implementation and evaluation of an enhanced load-balancing algorithm that allocates tasks across Virtual Machines on the cloud, therefore minimising Response Time and increasing resource utilisation. The ReT-ELBa distributes tasks considering their size and requirements for their executions. The cloud analyst simulation tool was employed to simulate or evaluate various cases for the ReT-ELBa in comparison with the Throttled and Round Robin algorithms. The study reveals that the ReT-ELBa outperforms the two algorithms with regard to Response Time. The ReT-ELBa also outperformed the two algorithms in terms of Data Centre Processing Time for all simulated cases except one (Case 3) which recorded a small difference of 0.07ms. Even though the Response Time was the Primary focus of this research, the Data Centre Processing Time of the algorithm was also analysed to present a more elaborate picture of the performances of the algorithms.

Keywords: IoT;Load Balancing;Cloud Computing;Algorithms.

1. INTRODUCTION

Cloud computing is a well-known technology that provides services (private and public) such as easy internet access to data, applications, and files. Users of this technology only pay according to their usage and it enhances businesses worldwide because it comes with the benefit of significantly reducing hardware cost among others. It enables a flexible allocation of resources and allows users to use services on demand [1].

In the cloud environment, the optimisation of resources (Virtual Machines) can be achieved through a method termed load balancing. This method involves a dynamic and equal shifting of workload and better resource allocation among processors or Virtual Machines to achieve enhanced resource utilisation. Load balancing reduces delays in the transmission of data and also prevents overloading of nodes in the cloud environment which affects the quality of service in the cloud Data Centres [2].

Cloud user requests are usually analysed and transferred to a particular Data Centre, which is done by considering the current resources of that Data Centre. The transmission mechanism is to ensure that the Virtual Machines are not underloaded or overloaded but to keep them balanced in their

handling of requests or loads. When a request is received, the distribution of this request to an appropriate Virtual Machine is achieved through load balancing; Cloud computing uses virtualization techniques to achieve this aim [3].

The traditional load-balancing techniques are categorised into three main types. These techniques are either implemented on the Data Centre end side, or, on the service broker policy (user end side) [4]. The first and second types are the static load balancing algorithms and the dynamic load balancing algorithms respectively. While the processes of the static algorithms demand prior knowledge of the system, its characteristics and capabilities like the processing power, the storage capacity and the memory, the dynamic algorithms perform better and are adaptable to dynamic environments. The dynamic algorithms even though complex, present flexibility benefits, taking into consideration the state of the previous system [5]. Nature-inspired algorithms constitute the third of the three types; they are intelligent algorithms that perform better for dynamic and complex systems, by mimicking how bees locate honey which is used as a search method in genetic algorithmic processes [6].

Response Time is one of the most important metrics by which the effectiveness of a load-balancing algorithm is measured. It is the time it takes the algorithm to send a response to a user's request. It considers the waiting, service and transmission time. For optimising the performance and effectiveness of a load-balancing algorithm, a minimal Response Time must be achieved.

.

2. RELATED WORKS

Nayak, Vania, and Robin proposed a modified Throttled Algorithm, which maintains an index table of the Virtual Machines with their respective states. It only selects the Virtual Machine at the first index and so on. Whenever there is a request from a client, it returns -1 when it is unable to find a Virtual Machine. Though this algorithm is slightly different from the traditional Throttled algorithm, the selection of Virtual Machines on the first index improves the utilisation of the Virtual Machines [7].

Ghosh and Banerjee also proposed a modified algorithm based on the Throttled algorithm which uses a switching queue to hold requests that have been temporarily removed from the Virtual Machine due to the arrival of a request with a higher priority. This algorithm improved the execution time when juxtaposed to the traditional Throttled algorithm but waiting requests are resumed only after the completion of requests with a higher priority which causes starvation for waiting requests and requests with a low priority [8].

Phi, Tin, Thu, and Hung proposed an improved algorithm that maintained two tables of Virtual Machines with the status *available* and *busy*. The proposed algorithm exhibited efficiency when the number of Virtual Machines increased. As the number of Virtual Machines increases, the Data Centre Processing Time also increases; the overall Response Time increases and the number of requests that queue for processing also becomes limited. This algorithm outperformed the existing traditional algorithms; however, their proposed algorithm was not subjected to diverse cases of configurations like increasing the number of Data Centres and sharing of loads at peak hours between the Data Centres to ascertain its extended efficiency [9].

Sachdeva and Kakkar also presented a hybrid algorithm which combines some advantages of both the Equally Spread Current Execution (ESCE) and the Throttled algorithm. It uses a HashMap to keep client requests, and it scans for a Virtual Machine with the least load to assign the request to it when there is a waiting request while the Virtual Machines are busy. This hybrid algorithm minimised the Response Time of the cloud environment [10].

Another hybrid load balancing algorithm as a result of the the combination of the ESCE and the Throttled algorithm was done to reduce the waiting time anytime when the number of tasks increased. The algorithm assigned a threshold limit for each Virtual Machine depending on the capacity of the individual Virtual Machines. However, the fault tolerance with this algorithm is minimal when one or more nodes fail [11].

The Spread spectrum technique and the Throttled algorithm were also integrated to distribute workload equally purposely for reducing cost and Response Time. The algorithm's aim was achieved by maintaining a threshold value as the priority for the individual Virtual Machines [12].

Kaurav and Yadav suggested an enhanced algorithm based on the Round Robin algorithm through the use of the genetic algorithm (an algorithm that works based on the evaluation of nature). The algorithm uses a HashMap to maintain the states (*available* or *busy*) of Virtual Machines. Upon receiving a request from the Data Centre, the enhanced algorithm scans through the HashMap from top to bottom for any Virtual Machine available to handle the request. For every request, the Virtual Machine chosen at random is compared with the previously routed Virtual Machine before the allocation is finally done. The comparison is done to distribute the request to the Virtual Machine with the shortest request queue length among the available ones [13].

3. METHODOLOGY

3.1 The Algorithm Design

Load balancers are usually designed to target CPU, Severs, Networks, Tasks, Virtual Machines, etc. The targeted domain of a balancer greatly dictates its key features and logic for decision-making as well as its requirements and constraints. In this study, the algorithmic efforts are targeted at balancing load on Virtual Machines, where requests are intended to be effectively distributed to servers in a manner that optimises the Response Time and the Data Centre Processing Time.

The ReT-ELBa algorithm was designed following a rule-based approach. This approach was chosen to establish novel defined rules for the balancer's intended efficiency. The algorithm employs task scheduling and resource allocation techniques, where tasks are assigned to computing resources, while the computing resources are monitored and maintained in three lists labelled as secondary, primary and engaged for the execution of the requests made. The level of activity in each Virtual Machine as well as the requirements of a request or task are also monitored. This monitoring effort feeds into the logic of the algorithm for the appropriate scheduling of tasks and allocation of resources.

3.2 The Key Parameters

Load balancers are usually not one-size-fits-all solutions for solving unbalanced load situations in Cloud computing. They are mostly heuristic in nature for establishing the most optimised use cases or scenarios, based on what key parameters are made the focus for improvement. In this research, the optimisation of the Response Time and the Data Centre Processing Time of the ReT-ELBa are the parameters targeted for assessing its efficiency. Two key things (among others) affect the response of load balancers; the load on a server and the requirement or complexity of requests. Based on these two effects, the ReT-ELBa considers the availability and load of Virtual Machines, and the requirements of a request for allocations. The goal of the algorithm which is to ensure that requests are being responded to quickly in a very efficient way, is achieved via the monitoring and optimisation of the Response Time.

3.3 Decision-making Logic and Work Load Distribution Technique

The decision-making logic of the ReT-ELBa is based on the optimisation of Response Time and resource utilisation load balancing policy, where thresholds are set for the request made to be executed on the Virtual Machines. A trigger condition is defined to offset the loads that are assigned to the Virtual Machines which are maintained and monitored in the primary and secondary lists. This analysis takes a real-time dynamic approach where new requests are assessed together with the current state of the Virtual Machines to make quick decisions to maintain a short Response Time.

The technique employed for the workload distribution is content-based, where the request is assessed for its content as well as the characteristics of the content. This establishes whether the request requires more or less resources to be executed; this forms a crucial layer for the proposed balancer to make its decisions for assigning loads to Virtual Machines.

4. THE ReT-ELBa

The ReT-ELBa maintains available Virtual Machines v in a primary list pl, where client requests r are maintained in a list rl scheduled for some v. The function $get(r)$ retrieves requests to be assigned to, and processed by some v via $v \rightarrow process(r)$; v is retrieved by the function $get(v)$. The pl must be a non-empty list for the assignment and process or r to be possible. The Virtual Machines in the pl that get engaged are maintained in a separate list λ, where their percentage is computed with the function $p(\lambda)$; this is used to check if the defined threshold t_v for engaged Virtual machines in pl is reached or otherwise. The engaged Virtual Machines are further assessed to establish if the requests they are engaged with are bigger requests $b(r)$ or smaller requests $s(r)$. When the Virtual Machines that are engaged with bigger requests exceed the defined threshold t_r, the non-engaged Virtual Machines will be maintained in a secondary list sl and allocated for processing smaller requests only until the engaged the Virtual Machines become less than the defined threshold.

Algorithm 1: The ReT-ELBa Algorithm

$1. pl = \{v: 1 \leq v \leq n\}, where\ v \in pl$
$2. sl = \{v: 0 \leq v \leq n\}, where\ v \in sl$
$3. rl = \{r: 0 \leq r \leq m\}, where\ r \in rl$
$4. tv = 0.7$
$5. tr = 0.5$
$6. get(v)$
$7. get(r)$
$8. while\ (|pl| > 1)$
$9. process(r) \rightarrow v, for\ v \in pl$
$10. end\ while$

11. $p(\lambda) = 100 - \left(\frac{sl}{pl}\ X\ 100\right)$
12. $sl\ =\ pl - \lambda$
13. $If\ p(\lambda)\ \geq\ (tv\ x\ pl)$
14. $If\ b(r)\ \geq\ (tr\ x\ \lambda)$
15. $sl\ =\ pl\ -\ \lambda$
16. $process(s(r))\ \rightarrow\ v, for\ v\ \in sl$
17. $end\ if$
18. end

In other words, when 70% of the total Virtual Machines on the primary list are used and 50% of the used Virtual Machines are as a result of bigger requests, the remaining number of Virtual Machines in the primary list are then used to create a secondary list of Virtual Machines for the processing of only smaller client request. Until the designated threshold is reached, the primary list of Virtual Machines will keep processing client requests.

5. EXPERIMENTAL RESULTS AND DISCUSSION

The experimental results obtained from the simulation of the ReT-ELBa were compared with the experimental results of the Throttled and Round Robin algorithms, where their corresponding average, minimum and maximum Response Times in milliseconds as well as Data Centre Processing Time were evaluated; this was done to assert the efficiency of ReT-ELBa.

5.1 Case 1: Using 50 Virtual Machines with one Data Centre

For all the simulations, the ReT-ELBa, the Round Robin algorithm and the Throttled algorithm were tested. In this case, 50 Virtual Machines from one Data Centre were tested and their corresponding Response Time were recorded. It turned out that the ReT-ELBa presents less Response Time than the other two algorithms with an average time of 249.25ms. The Response Time for the Throttled recorded an average of 249.26ms, and 271.51ms for the Round Robin algorithm (see Table 1).

Even though the ReT-ELBa performed better, it did not present a significant optimised Response Time compared to the Throttled algorithm. The Throttle does quite well in Response Time because it balances the workload by using two index tables of Virtual Machines, which enables it to utilise resources effectively. However, the Throttled algorithm always has to scan through the entire table for available or idle Virtual Machines to process a request. The downside of this is that the scanning may increase the Response Time in cases the idle Virtual Machine needed to process the request is at the bottom of the table.

Table 1: Response Time for Case 1

Algorithm	Average(ms)	Minimum(ms)	Maximum(ms)
Throttled	249.26	40.30	642.62
Round Robin	271.51	40.46	642.62
ReT-ELBa	249.25	40.29	642.62

The simulations were also done to assess the Data Centre processing time, which is the time it takes a Data Centre to process a request. The simulation results for Case 1 indicate that it takes 44.65ms on average to process a request by the Round Robin algorithm, whereas both the Throttled and the ReT-ELBa algorithmseach recored 22.57ms on average to do the same. Again, the ReT-ELBa appears to have a similar performance to the Throttled algorithm, while the Round Robin performs worst.

Table 2: Data Centre Processing Time for Case 1

Algorithm	Average(ms)	Minimum(ms)	Maximum(ms)
Throttled	22.57	0.12	102.27
Round Robin	44.65	0.12	117.55
ReT-ELBa	22.57	0.12	102.27

5.2 Case 2: 50 Virtual Machines for 2 Data Centres (25 each)

In this case, we simulate for a smaller number of Virtual Machines in two Date Centres. It is observed that (Table 3) the ReT-ELBa maintains a little better performance than the Throttled algorithm, while the Round Robin still lags in terms of Response Time. A similar performance is also recorded for the Data Centre Processing Time (Table 4).

Table 3: Response Time for Case 2

Algorithm	Average(ms)	Minimum(ms)	Maximum(ms)
Throttled	234.89	38.33	643.12
Round Robin	256.82	38.33	643.12
ReT-ELBa	234.86	38.33	643.12

Table 4: Data Centre Processing Time for Case 2

Algorithm	Average(ms)	Minimum(ms)	Maximum(ms)
Throttled	22.40	0.13	102.75
Round Robin	44.17	0.13	111.09
ReT-ELBa	22.38	0.13	102.75

5.3 Case 3: 100 Virtual Machines for 2 Data Centres (50 each)

The number of Virtual Machines were increased for two Data Centres, for this simulation. The ReT-ELBa continues to record the least Response Time, with an average of 234.86ms having to balance load not only within a Data Center but across other Data Centres as compared to the Throttled and the Round Robin, even though the Round Robin continues to be the worst performing algorithm among the three. The Throttled algorithm however outperforms the ReT-ELBa with difference of 0.07ms for the Data Centre Processing Time for the case.

Table 5: Response Time for Case 3

Algorithm	Average(ms)	Minimum(ms)	Maximum(ms)

Throttled	237.30	39.32	930.13
Round Robin	259.21	38.43	930.13
ReT-ELBa	237.24	39.32	930.13

Table 6: Data Centre Processing Time for Case 3

Algorithm	Average(ms)	Minimum(ms)	Maximum(ms)
Throttled	22.50	0.12	103.08
Round Robin	44.13	0.12	112.77
ReT-ELBa	22.57	0.12	103.88

5.4 Case 4: 150 Virtual Machines for 3 Data Centres (50 in each Data Centre)

The number of Virtual Machines were further increased for three Data Centres where we note the consistent performance differences among the three algorithms for Response Time and the Data Centre Processing Time as seen for Case 1 and Case 2. The ReT-ELBa outperforms the Throttled and the Round Robin algorithms in this case as well.

Table 7: Response Time for Case 4

Algorithm	Average(ms)	Minimum(ms)	Maximum(ms)
Throttled	136.66	37.85	390.25
Round Robin	163.34	38.21	388.62
ReT-ELBa	136.64	37.85	386.62

Table 8: Data Centre Processing Time for Case 3

Algorithm	Average(ms)	Minimum(ms)	Maximum(ms)
Throttled	27.14	0.12	103.18
Round Robin	53.55	0.12	112.78
ReT-ELBa	27.13	0.12	103.18

5.5 Performance Evaluation of ReT-ELBa

Generally, the ReT-ELBa shows similar performance with Throttled algorithm for Response Time with the ReT-ELBa performing better in all four cases. The ReT-ELBa in one instance (Case 3) performed a little less than the Throttled algorithm as indicated earlier. The three algorithms did not show any significant differences in the minimum and maximum recordings for both Response Time and Data Processing Time.

Figure 1 presents an interesting pattern for the Response Time in Case 4. It is expected that as the Data Centres increase the Response Time should also increase in direct proportionality. However, it is observed that, Case 4 records the lowest Response Time for the three algorithms. The feature accounting for this phenomenon is the fact that, the high number of Virtual Machines offers proximity and more availability for requests to be scheduled, it also lessens the number of engaged machines which makes it reach the threshold slowly. Availability translate into quick finding and assigning of requests and hence the pattern we see in the results.

As illustrated in Figure 2, as the number of Data Centres increases, the Data Centre Processing Time also increases from one case to the other. So, algorithms with better Data Centre processing times are expected to have a better Response Time and vice versa.

In Case 1, both the ReT-ELBa and the Throttled algorithms recorded the same Data Centre processing values of 22.57ms as compared to the Round Robin algorithm's 44.65ms. The Round Robin algorithm appear touses about twice as much time as used by the ReT-ELBa and the Throttled algorithm. A similar observation can be made from the experimental results for Case 2, Case 3 and Case 4.

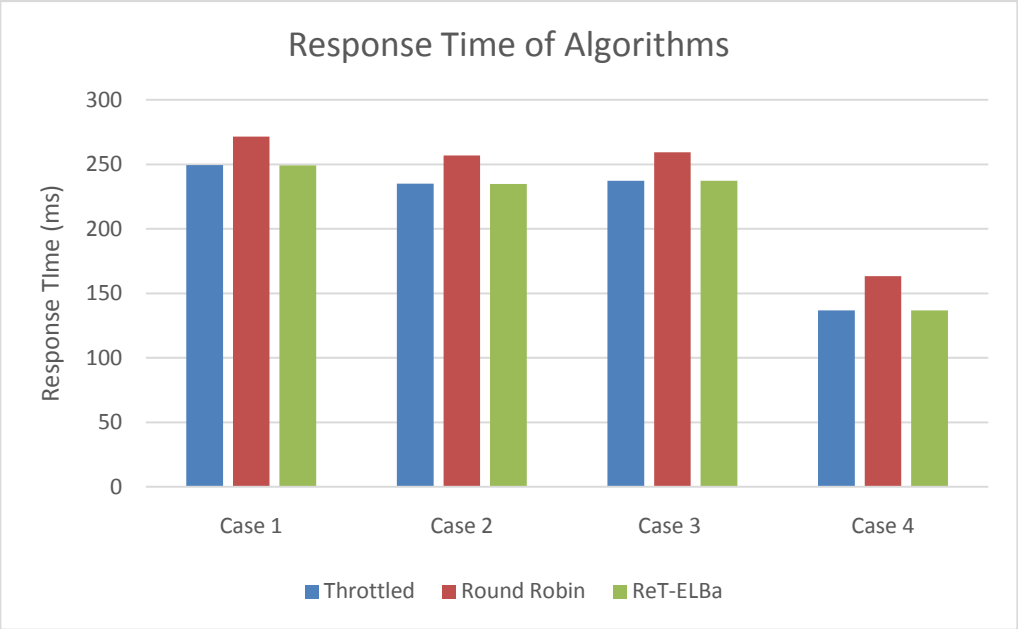

Figure 1: Performance Analysis of Response Time

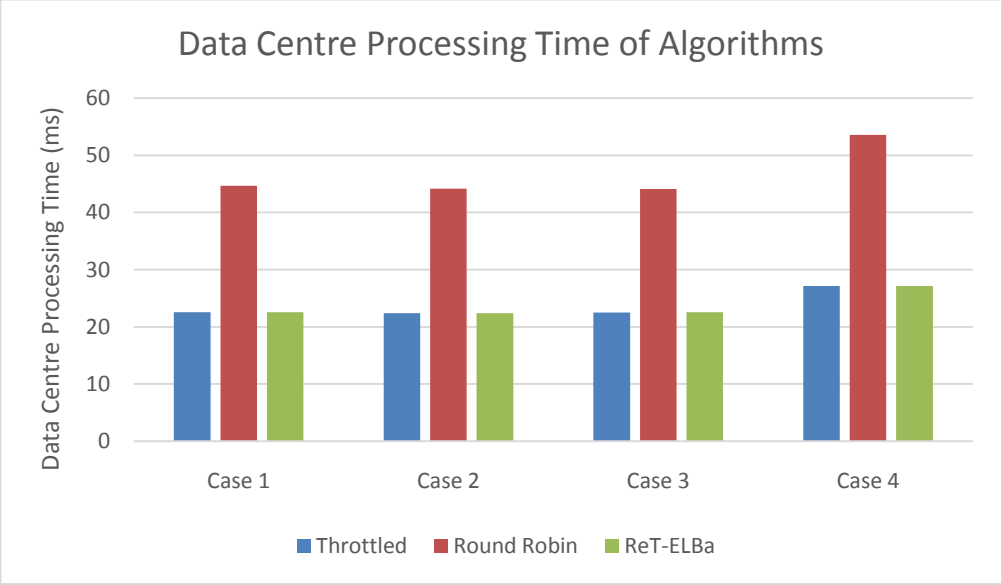

Figure 2: Performance Analysis of Data Centre Processing Time

6. CONCLUSIONS

This research presents a novel algorithm targeting optimising the Response and Data Centre Processing Time in comparison with the classical Throttled and Round Robin algorithms. The ReT-ELBa consistently outperforms the two algorithms with a not-too-significant difference from the performance of the Throttled algorithm. It however significantly outperforms the round Round Robin algorithm for both Response Time and Data Centre Processing Time.

Four simulation cases were completed to assess the performance of the algorithms for Virtual Machines in one Data Centre, as well as in multiple Data Centres with increasing number of Virtual machines, to assert if the performance of the algorithms would be affected by the variations. It turned out that such variations had no significant impact of how the algorithms performed in terms of their Response Time and Data Centre Processing Time.

Accounting for the ReT-ELBa's performance is its ability to track the requirements of requests, the current load and type of load on Virtual machines in its logics, in order to decide how to balance new load for existing Virtual Machines. The mechanism of maintaining list of primary machines engaged machines, and idle machines makes it easy for the ReT-ELBa to switch loading each category differently according to its state.

References

[1]. Hemalatha, M. (2012). An approach on semi-distributed load balancing algorithm for cloud computing system. International Journal of Computer Applications, 56(12):5-10.

[2]. Kaur, R. and Luthra, P. (2014). Load balancing in cloud system using max min and min min algorithm. International Journal of Computer Applications, 975:8887.

[3]. Saroj, H. D. K. (2011). Adaptive round robin scheduling using shortest burst approach based on smart time slice.

[4]. Choudhary, R. and Kothari, D. (2018). A novel technique for load balancing in cloud computing environment. International Journal of Software and Hardware Research in Engineering, 6(6):1-5

[5]. Alam, M. and Khan, Z. A. (2017). Issues and challenges of load balancing algorithm in cloud computing environment. Indian journal of science and Technology, 10(25):1-12.

[6]. Thakur, A. and Goraya, M. S. (2017). A taxonomic survey on load balancing in cloud. Journal of Network and Computer Applications, 98:43-57

[7]. Nayak, P., Vania, J., and Robin, R. (2015). Load balancing using modified throttled algorithm. International Journal of Science and Research Development, 3(3):3614-3616.

[8]. Ghosh, S. and Banerjee, C. (2016). Priority based modified throttled algorithm in cloud computing. In 2016 international conference on inventive computation technologies (ICICT), volume 3, pages 1-6. IEEE.

[9]. Phi, N. X., Tin, C. T., Thu, L. N. K., and Hung, T. C. (2018). Proposed load balancing algorithm to reduce Response Time and processing time on cloud computing. International Journal of Computer Networks and Communications, 10(3):87-98.

[10]. Sachdeva, R. and Kakkar, S. (2017). A novel approach in cloud computing for load balancing using composite algorithms. International Journal of Advanced Research in Computer Science and Software Engineering, 7(2):198.

[11]. Rathore, J., Keswani, B., and Rathore, V. (2018). An efficient load balancing algorithm for cloud environment. Journal of Inventions in Computer Science and Communication Technology, 4(1):37-41.

[12]. Aliyu, A. N. and Souley, P. (2019). Performance analysis of a hybrid approach to enhance load balancing in a heterogeneous cloud environment. International Journal of Advanced Science, Research and Engineering, 5(7):246-257.

[13]. Kaurav, N. S. and Yadav, P. (2019). A genetic algorithm-based load balancing approach for resource optimization for cloud computing environment. International Journal Information and Computer Science, 6(3):175-184.

[14]. Malhotra, M. (2017). A review, different improvised throttled load balancing algorithms in cloud computing environment. International Journal of Engineering, Technology Management and. Applied Sciences, 5(7):410-416.

[15].Villanueva, J. (2015). Comparing load balancing algorithms. Retrieved August, 10:2016.