Least Afflicted Load Balancing Algorithm (LALBA) for Performance Improvement of Multi-Scale Applications in Cloud Environment

Vijaya Kumbhar¹, Jyoti Yadav², Ajay H. Kumar³

¹Asst. Prof., Indira College of Commerce and Science, Pune.
Research Scholar, Savitribai Phule Pune University, Pune/India
²Assistant Professor, Savitribai Phule Pune University, Pune, India
Research Guide, Savitribai Phule Pune University, Pune, India.
³Director, JSPM’s Jaywant Technical Campus, Pune, India
Research Guide, Savitribai Phule Pune University, Pune, India

Article History: Received: 10 November 2020; Revised: 12 January 2021; Accepted: 27 January 2021; Published online: 05 April 2021

Abstract: Cloud computing is a new image of Information Technology and the way applications are available to end-users. Also, the applications available are growing popularity as being able to demonstrate the flexibility of use of resources as in Multi-Scale Cloud-Based Applications. These applications on the cloud are hosted on remote virtual machines with multi-tenancy architecture. Multi-tenant cloud computing architecture helps cloud users and customers to share computing resources and still the individual tenant's data is stored separately so as to make it invisible to other tenants. With this multi-tenancy architecture, the application is being used by multiple people at the same time. Also, the application resides in various copies across various servers situated at different geographical locations. Virtual Machines that are housed in multiple Data Centers are coupled by way of load balancers. These load balancers are responsible for the optimal distribution of load/demand/tenant requests and the size of the data held within the request. So in short, two areas need to be focused on when architecting such systems. Firstly, the load must be balanced optimally across data centers and virtual machines for seamless operation at the back end. Secondly, at the front end, it is imperative that every tenant enjoys optimal service speeds without any degradation of service request speed. In case the back-end virtual machines are unbalanced in their load then there is bound to be degradation of services. For this very purpose, various researchers/academicians, and practitioners in the cloud space have been working tirelessly towards developing optimal algorithms to get the best load balancing architecture to ensure that the objective of seamless back end and front end quality of service in terms of average response times. Multi-Scale applications consist of variable payload for time and often unpredictable i.e. load can be increased/decreased suddenly or gradually. Hence to balance these loads equally at any time, a new algorithm is proposed. This algorithm considers the characteristics of multiscale applications in terms of data size and runtime characteristics of VM in terms of capacity and availability. Finally, the performance of this method is analyzed qualitatively as well as quantitatively by performing simulation experiments of multiscale applications.

Keywords: Load Balancing, Least Afflicted VM LBA, Multiscale Applications, Performance Analysis, Virtual Machine, multi-tenancy

1. Introduction

Cloud Computing once an emerging technology, has now occupied the world and redefined the way the “applications are accessed and made available to end-users” (Jyoti, A. et.al., 2015) across the world. Almost all recent applications are now implemented, deployed, and executed on a cloud. Open applications are hosted on public clouds and applications for a closed group of users of a community are hosted on private or community clouds respectively. These clouds may have a link to the public clouds also and so are termed as hybrid clouds. These clouds provide the preliminary services of software, platform and infrastructure to the users on-the-go (Manikandan, N., 2019). As many applications are moving to cloud, many practical problems arise while using cloud applications (Kumar, 2014 and Vijaya K.et al, 2019). Multiscale applications are few of them, which comprise of mutable data sizes with respect to varying data types. The data involved in such applications is accessible to its users anytime and anywhere, hence such cloud applications are becoming more and more popular (Gupta, P., 2013). According to Gartner’s statistical report, from 2013 to 2018, users of cloud computing services have increased from 30% to 50% (Statista Report, 2019) It can be predicted that in subsequent five years it may reach up to 75%.

When these applications are utilised more and more, the number of concurrent requests generated and sent to the cloud systems exponentially (Jyoti, A., 2020). This creates huge load on cloud system which is incurred due to number of users, data size per requests, types of requests (Jyoti, A., 2020). Hence to manage and balance this load for consistent and reliable performance, there is a need to have a sustainable method. This job is done using load balancing algorithms (Jyoti, A., 2020). This is known as Load Balancing Problem in Cloud Computing. In
case of business applications and social networking applications, the performance is measured in the form of time (Jyoti, A., 2020). to service the request and it is a key differentiator between cloud based multi scale applications even to the extent of its wide spread acceptance and success. Many multi scale applications have lost the race towards success due to the lagging of speeds in terms of time taken to process the requests. In general terms this can be referred to as a light weight and fast applications versus those applications which take a lot of time to process requests (Vijaya K., 2019). Infrastructure is costly in terms of hardware and the physical location where the hardware data centers are lodged. If the infrastructure is to be kept fixed to manage the costs, load balancing is the best mechanism to enhance the performance of the applications and data hosted in cloud systems with respect to waiting time, response time and throughput.

Load balancing is the procedure for distributing the amount of workload into various available servers in such a way that all servers will have near equal load. Well balanced servers tend to show better performance and the end user can optimally enjoy the high speeds offered by these clouds based multi scale applications. This load balancing is achieved by way of a software program which manages the distribution of the requests across servers. Thus, Load Balancing algorithm is the mechanism of balancing such load programmatically. Figure 1 depicts the pictorial representation of load balancing concept.

![Figure 1. Conceptual Framework of Load Balancer](image)

2. Research Problem

Multiscale applications are the applications that involve varying data sizes and data types. Consequently, when such applications are executed in cloud environment, it can lead to situations where the small sized requests get delayed due to large size request already been queued to be executed on the VMs. Effectively, the waiting time and the response time increase resulting in degradation of performance.

Hence, the objectives of the proposed research are summarized as follows:

- To study fundamental load balancing algorithms in the cloud environment.
- To improve the performance of multiscale applications for better waiting time and response time.

3. Related Work
Many researchers have proposed different algorithms for performance improvement of cloud-based applications, which are basically categorized as static or dynamic. A broad class of algorithms are based on design strategies like divide and conquer, greedy method, round-robin method, and evolutionary algorithms like honeybee, Particle swarm algorithm. As such, some of these algorithms are implemented for workflow applications, some are for scientific applications, some of them are for performance improvement while the remaining are used for optimization purposes. Every algorithm has its own strengths and pitfalls too. Depending on the problem to be solved, the appropriate algorithm should be considered for implementation. Wickremasinghe et. al (2009) proposed the three algorithms which are Round Robin, Active Monitoring and Throttled. In 2010, Zhang et.al (2010) implemented new and novel load balancing mechanism using the strategy of ant colony and complex network. Tian et. al (2011) suggested dynamic and integrated algorithm. Fayoumi et. al (2011) suggested Pareto traffic algorithm for distributing burden of VMs. Elzeki et. al (2012) coined new algorithm namely Improved max-min and Maguluri et. al (2012) designed stochastic models. Trust and reliability based algorithm was given by Gupta et. al (2013). Dynamic compare and balance algorithm was suggested by Sahu et. al (2013). ‘Zero proof authentication’ was suggested by Sran et. al (2013)


Round Robin was again proposed by Nishad et. al (2016). Parekh et. al (2016) proposed new algorithm based on Distance, energy and storage. Mousavi et. al (2016) merged HBA and ACO and proposed new algorithm based on these two algorithms. Tiwari et. al (2016) gave an algorithm which was hybrid algorithm based on DAG and Honey Bee. Babu et. al (2016) enriched Bee Colony algorithm by giving the name to new algorithm as “Enhanced Bee Colony”. Zhou et. al (2016) pooled two algorithms analytic hierarchy process and hybrid hierarchical genetic algorithms and developed new one. This was happened in 2016.


The algorithms developed by Buyya, R. (2009) and Kumar et. al (2014) are based on independent load of VMs, computed as number of requests allotted to it. But many a times, single request itself is big enough as compared to number of smaller requests. Many algorithms focus on the number of incoming requests, but fail to consider the data size of the allocated request resulting in an unbalance of load on various VMs. Thus, some VMs are overloaded and some VMs are under loaded. Hence there is a need to establish a balance between number of requests with number of virtual machines by considering data size as a crucial parameter and capacity of the VM.

4. Proposed Work

The proposed algorithm is based on “Active Monitoring” algorithm ( Buyya R., 2009 and Kumar V.,2014 ). The algorithm is based on a load of the virtual machine which can be calculated in a better way so that the correct load is distributed amongst the VMs, which will ultimately improve utilization of each VM and thus the performance of the entire system. Instead of just counting the number of requests as the load of requests, genuine
burden of each VM is calculated like this and then VM of minimum load is searched for the new request assignment of VM.

Therefore, the cumulative load L of each virtual machine is formulated as:

\[ \text{Total Load (L)} = D(R_1) + D(R_2) + D(R_3) \ldots D(R_n) \]  \hspace{1cm} (1)

Where \( D(R_1), D(R_2), D(R_3) \ldots D(R_n) \) represents the data size of individual request.

4.1. Mathematical Model of the proposed method

As the goal of new algorithm is for better performance of all requests, the objective of this research is to improve the response time of ‘n’ requests. Minimal response time refers to better performance; therefore the objective of this research is to minimize the response time of all requests. Response time of each requests is denoted as \( T(R_i) \). It is computed on the basis of Network Latency Time of respective request, data size of individual request and Bandwidth received for each request. The data size of individual requests is denoted by \( D(R_i) \). The requests are assigned to most appropriate Virtual Machine. This most appropriate VM is determined by \( \text{min()} \) function, which discovers the VM of least load. The Load of each VM is calculated using cumulative addition of all assigned requests of it. This load of each VM is represented as \( L(VM_j) \). VMj represents the most appropriate VM which is assigned to processing request \( R_i \).

Hence, this process mathematically modelled as:

The objective function is:

\[ \text{Minimize } \sum_{1 \leq i \leq n} T(R_i) \]  \hspace{1cm} (2)

\[ \text{Such that } T(R_i) = \text{Network Latency Time (R)} + \frac{D(R_i)}{\text{Bandwidth}(R_i)} \]  \hspace{1cm} (3)

\[ , L(VM_j) = \min (L(VM_1), L(VM_2), L(VM_3) \ldots L(VM_m)) \]  \hspace{1cm} (4)

and Bandwidth\((R_j) = \frac{\text{Total Network Bandwidth}}{\text{Total Number of Concurrent Users}} \)  \hspace{1cm} (5)

where,

\( R_i \rightarrow i^{th}\) request with configuration as (Application ID, Region ID, datasize) \\
\( D(R_i) \rightarrow \text{Datasize of } i^{th} \text{ request} \) \\
\( \text{Bandwidth}(R_j) \rightarrow \text{Bandwidth of } j^{th} \text{ user} \) \\
\( T(R_i) \rightarrow \text{Time required to complete } i^{th} \text{ request} \) \\
\( VM_j \rightarrow j^{th} \text{ Virtual Machine allocated for processing current request} \) \\
\( L(VM_j) \rightarrow \text{Cumulative load of } j^{th} \text{ Virtual Machine} \) \\
\( n \rightarrow \text{Total number of incoming requests.} \)

4.2. Pseudocode of Proposed Least Afflicted Load Balancing Algorithm (LALBA)

Algorithm-1. Least Afflicted Load balancing Algorithms searches for the virtual machine (VM) having minimal load amongst all available VMs. This VM id is then sent to data centre controller(DCC). This procedure is shown in Algorithm-2. DCC assigns the respective request to the VM. It then increases load count of each VM. After the processing of the request, the load gets reduced. This process is continued in Algorithm-1 as continuation of first procedure.

Hence, the Pseudocode of the LALBA algorithm is given Algorithm-1 and Algorithm-2.

Algorithm-1: Data Center Controller Algorithm (DCC)

01: \textbf{Input}: Data Sizes of all incoming requests : \{\( D(R_1), D(R_2), \ldots D(R_n) \) \},

02: \hspace{1cm} Allocation Table of Requests to VM : Allocation (VMj, Rj),

03: \hspace{1cm} Total Bandwidth,

04: \hspace{1cm} Total users,

05: \hspace{1cm} Load of all VMs : \{Load(VM_1), Load(VM_2), \ldots, Load(VM_m)\},

06: \hspace{1cm} Network Latency Times : \{NLT(R_1), NLT(R_2), \ldots, NLT(R_N)\},

07: \textbf{Output}: Response Time of all requests : \{T(R_1), T(R_2), \ldots, T(R_N)\},

08: \hspace{1cm} Overall Response Time : sum.

09: \hspace{1cm} sum ← 0;
Least Afflicted Load Balancing Algorithm (LALBA) for Performance Improvement of Multi-Scale Applications in Cloud Environment

Algorithm-2: Least Afflicted Load Balancing Algorithm (LALBA)

01: \textbf{Input} : Load of all VMs : \{Load(VM_1), Load(VM_2), ..., Load(VM_m)\}
02: \textbf{Maximum Integer Value} : Max\_Int.
03: \textbf{Output} : ID of VM having minimal Load : VMid.
04: \(i \leftarrow 0\);
05: \(VMid \leftarrow -1\);
06: \(min \leftarrow \text{Max\_Int}\);
07: \textbf{for} \(i : 1 \text{ to } m \text{ do}\)
08: \quad \textbf{if} \(min < \text{Load (VMi)} \text{ then}\)
09: \quad \quad \(min = \text{Load (VMi)}\);
10: \quad \(VMid = i\);
11: \quad \textbf{end if}\;
12: \textbf{next} \(i\);
13: \textbf{return} \(VMid\);

4.3. Flowchart for Proposed Least Afflicted Load Balancing Algorithm (LALBA)

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{flowchart.png}
\caption{Flowchart of the proposed LALBA algorithm}
\end{figure}
Figure 3. Flowchart of the Data Center Controller (DCC)

Figure 2 and Figure 3 depicts the procedure of the proposed algorithm. There is large number of requests received at respective cloudlet, to be processed at any time. All such requests are maintained in a request queue. Such requests are processed one by one using First Come First Served (FCFS) mechanism. “Whenever a cloudlet request is passed to data center controller, data centre controller (DCC) forwards that request to virtual machine (VM) load balancer” (Kumar, V., 2014). Load balancer finds which VM is free or least load value which has lowest load by computing load using data size incurred in requests and this VM id is used by data center controller for allocation and computation of load value of each VM. DCC assigns the respective request to the VM. The VM allocation table is used for storage of VMid and assigned requests (Kumar, V., 2014). It then increases load count of each VM. After the processing of the request, the load gets reduced.

4.4. Complexity Analysis of the proposed LALBA Algorithm

After introducing the algorithms for balancing the load of Virtual machines, the intricacy of the algorithm analysed using space and time complexity and then by performing the experiments based on selected scenarios.

4.4.1. Time Complexity:

It is clear that the LALBA algorithms searches all virtual machines and checks the load values of each machine. It can receive the ID of the respective machine by scanning loads of all machines. Let’s assume that there are ‘n’ virtual machines, therefore the algorithms take O(n) time to execute. Even if the first Virtual machine has least load, the algorithm scans load values of all virtual machines. Hence in Best case, Average case and worst case, the algorithm takes the same time i.e. O(n).

Hence, \( Space \ (LALBA) = O(n \times k) \) ............................................(6)

where \( n \in \text{number of virtual machines and } k \text{ is constant} \)

4.4.2. Space Complexity:
The space complexity in this algorithm is the extra memory space required to hold the Load Value of each Virtual Machine. Assume that each VM has ‘m’ requests assigned it. But it requires fixed unit of memory such as ‘k’ as it computes the load as addition of data sizes of all requests. Therefore the LALBA requires \( O(n \times k) \) as extra memory space for its execution.

\[
\text{Space (LALBA)} = O(n \times k) \tag{7}
\]

where \( n \) ∈ number of virtual machines and \( k \) is constant

5. Experimental Work

CloudAnalyst v1.0 (Buyya, R., 2009) simulation tool is used for the implementation and testing of the proposed LALBA algorithm. “The CloudAnalyst is extending functionality of CloudSim and built on top of CloudSim tool kit. It is GUI-based simulation tool that also gives simulation result in graphical form” (Kumar, V., 2014).

5.1. Simulation Setup and parameters:

In order to execute the experiments and testing of proposed algorithm, the basic and fixed configuration is set in the simulation tool. These configuration user’s setup, described in Table 1, parameters for data centers and virtual machines, described in Table 2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of concurrent users</td>
<td>10</td>
</tr>
<tr>
<td>Number of concurrent requests</td>
<td>10</td>
</tr>
<tr>
<td>Executable instruction length per request</td>
<td>100 bytes</td>
</tr>
<tr>
<td>Number of users</td>
<td>1 to 10^{10}</td>
</tr>
<tr>
<td>Existing &amp; Proposed Load Balancing Algorithms</td>
<td>Round Robin (RR LBA), Active Monitoring (AM LBA) and Throttled (TH LBA) Least Afflicted (LA LBA)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of hardware units(server)</td>
<td>02</td>
</tr>
<tr>
<td>No. of VM(Virtual Machine(VMs))</td>
<td>02</td>
</tr>
<tr>
<td>Operating System of servers</td>
<td>Linux OS</td>
</tr>
<tr>
<td>Processor Architecture</td>
<td>x86</td>
</tr>
<tr>
<td>Virtual Machine Monitor(VMM)</td>
<td>Xen</td>
</tr>
<tr>
<td>Memory</td>
<td>204800 Mb</td>
</tr>
<tr>
<td>Storage</td>
<td>100000000 Mb</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>1000 Kbps</td>
</tr>
<tr>
<td>No. of processors</td>
<td>4</td>
</tr>
<tr>
<td>Processor Speed</td>
<td>10000 Mhz</td>
</tr>
</tbody>
</table>

5.2. Performance Metrics for Evaluation:

From the set of numerous quantitative and qualitative metrics defined by researchers and system evaluators in the study of performance of lead bearing algorithms, the most valuable quantitative metrics are Waiting Time, Average Response Time and Throughput (Buyya, R. 2009, Jyoti, A. et. al 2020). For this research work, response time is considered for the comparative analysis of existing and proposed algorithm. The metric response time is the time that requires to response the requests of applications i.e. the time interval from submission of request till response received (Manikandan, N., & Pavin, A., 2019). It is computed as mean of response times of all users. It is formulated in Equation (Elzeki et.al., 2012).

\[
\text{Average Response Time} = \frac{\sum \text{Response times of all requests}}{\text{number of requests per user} \times \text{number of users}} \tag{8}
\]

5.3. Simulation Results and Analysis
For simulation purpose, two scenarios are considered viz. users expandability and the number of requests respectively. The hardware configuration is kept constant as described in Table 1, Table 2 and Table 3 as the focus of this research is to evaluate the testing and comparing existing and proposed algorithm.

**Scenario-1: With Users Expandability**

In this case the numbers of users are increased exponentially keeping in mind that the application is accesses by millions of users from worldwide. The results obtained by the simulation experiment of existing and proposed algorithm are shown in following Figure 3 and Table 4.

<table>
<thead>
<tr>
<th>No. of users</th>
<th>Avg. Response Time [Ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RR LBA</td>
</tr>
<tr>
<td>1</td>
<td>8253.11</td>
</tr>
<tr>
<td>10^1</td>
<td>5885.82</td>
</tr>
<tr>
<td>10^2</td>
<td>6425.07</td>
</tr>
<tr>
<td>10^3</td>
<td>5906.9</td>
</tr>
<tr>
<td>10^4</td>
<td>5580.94</td>
</tr>
<tr>
<td>10^5</td>
<td>9709.06</td>
</tr>
<tr>
<td>10^6</td>
<td>8392.42</td>
</tr>
<tr>
<td>10^7</td>
<td>8197.44</td>
</tr>
<tr>
<td>10^8</td>
<td>4192.72</td>
</tr>
<tr>
<td>10^9</td>
<td>9671.17</td>
</tr>
<tr>
<td>10^10</td>
<td>9571.18</td>
</tr>
</tbody>
</table>

**Observation**

From the Figure 3 and Table 4, it is observed that, with expansion of number of concurrent users, the overall response time after 10000 users till billion users has been reduced by the proposed algorithm. The graph also shows that the variation (or fluctuation) of the proposed algorithms is comparatively less than existing algorithms.

**Scenario-II: With Requests Expandability**
In this case the numbers of requests per users are increased exponentially keeping in mind that each user can give multiple requests at the same time for single application. The results obtained by the executing existing and proposed algorithm are shown in following Figure 4 and Table 5.

**Table 4. Tabular Representation of Average Response Times for LAVMLBA**

<table>
<thead>
<tr>
<th>Number of Requests Per Users</th>
<th>Avg. Response Time[ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RR LBA</td>
</tr>
<tr>
<td>100</td>
<td>7442.03</td>
</tr>
<tr>
<td>101</td>
<td>13566.97</td>
</tr>
<tr>
<td>102</td>
<td>90971.88</td>
</tr>
<tr>
<td>103</td>
<td>68626.81</td>
</tr>
<tr>
<td>104</td>
<td>83434.77</td>
</tr>
</tbody>
</table>

**Figure 4. Graphical Representation of Variation of Response Times for LAVMLBA**

**Observation**

Figure 4 and Table 5, depicts that observed that, when each user is sending large number of requests at a given interval of time, the response time after 100 users till 1000 requests has been reduced (with least significant that previous scenario) by the proposed algorithm. The graph in Figure-4 depicts that the variation (or fluctuation) of the proposed algorithms is comparatively less than existing algorithms.

**5.4. Discussion**

The above results interpret that certainly LALBA algorithm shows better performance with respect to the results of average response time and throughput as compared to Round Robin, Active Monitoring and Throttled algorithm. The traditional and standard algorithm Round Robin has less performance than proposed Least Afflicted algorithm.

**6. Conclusion & Future Work**

Study of load bearing of virtual machines is critical to the success and failure of a multi scale application on the cloud due to the optimal access speeds enjoyed by the user. Researchers have in numerous independent studies stressed on this fact and academicians / researchers and practitioners in this area have untiringly worked towards proposing numerous algorithms to provide optimal and seamless access to the cloud based applications. In this paper, the proposed method, Least Afflicted VM load balancing Algorithm emphasized on effective utilization of VMs by computing the load of the virtual machine on the basis of data size of assigned requests.
assigned to it. The algorithm is verified using CloudAnalyst Simulator and result shows improvement over existing algorithms. The proposed method can be helpful for the cloud practitioners working in application scalability and its performance optimization domain. Hence it can be concluded that, for multiscale applications, appropriate load balancing based on attributes of input data affects the performance of application.

References


