CLBS-3: A Three-Tier Load Balancer for ensuring Fault-Tolerance of Software running in Open-Source Cloud

Conference Paper · September 2015

DOI: 10.1145/2832987.2833068

7 authors, including:

Asif Imran

University of Dhaka

18 PUBLICATIONS **26** CITATIONS

SEE PROFILE

Shadi Aljawarneh

Jordan University of Science and Technology

61 PUBLICATIONS **65** CITATIONS

SEE PROFILE

Kazi Sakib

University of Dhaka

40 PUBLICATIONS **52** CITATIONS

SEE PROFILE

Kazi Salatuzzaman

National Academy for Planning And Develo…

1 PUBLICATION **0** CITATIONS

SEE PROFILE

CLBS-3: A Three-Tier Load Balancer for ensuring Fault-Tolerance of Software running in Open-Source Cloud

Md. Shariful Alam
bit0433@iit.du.ac.bd
IIT, University of Dhaka
Ramna, Dhaka 1000,
Bangladesh

Syed Ali Asif
bit0432@iit.du.ac.bd
IIT, University of Dhaka
Ramna, Dhaka 1000,
Bangladesh

Md. Shamsuddoha
bit0309@iit.du.ac.bd
IIT, University of Dhaka
Ramna, Dhaka 1000,
Bangladesh

Dr. Shadi Aljawarneh
shadi.jawarneh@yahoo.com
Software Engineering
Department
Jordan University of Science
and Technology
Irbid, Jordan

Kazi Sakib
sakib@iit.du.ac.bd
IIT, University of Dhaka
Ramna, Dhaka 1000,
Bangladesh

Asif Imran*
asifimran@du.ac.bd
IIT, University of Dhaka
Ramna, Dhaka 1000,
Bangladesh

ABSTRACT

High availability and low service expenditure has enabled cloud computing to become popular to be used in providing large number of software services. However, supporting a large collection of software together with data storage present new threats to the cloud in terms load imbalance and service failures. This paper proposes CLBS-3, a three-tier Load balancing Schema (LBS) for the cloud preserving low communication overhead and fault tolerance. Next, it provides a mechanism to recover from a failed condition and excessive load within short time and low network traffic. Performance analysis reveal desirable service delivery speed of 11.40 to 84.43 seconds. Also the less time taken to recover from a failure ensures that the service is consistent and reliable. The experimentation of the proposed framework on cloud platform with web enabled software services yield desirable performance.

Keywords

Fault Tolerance; Load Balancing in Cloud; Software Reliability.

1. INTRODUCTION

Cloud computing has become a widely popular mechanism for providing software services through it mainly due to its high availability and ease of access using the World Wide Web (WWW). Due to an increasing number of software service being provided by the cloud, ensuring its reliability through effective load-balancing and failure resiliency measures have become a high priority. In this regard, solving the problem of load balancing of cloud is a major challenge due to the huge amount of load imbalance cases that occur in distributed environments. Also, the traditional mechanisms of using many levels of machines to solve the load imbalance issue present newer complications of communication overhead since a lot of messages need to be transmitted across those machines. Hence, an effective load balancing mechanism with minimal tiers need to be proposed for the cloud to ensure proper load distribution at reduced communication overhead. This will ensure an improved load balanced and fault tolerant cloud that will increase its popularity amongst the software vendors, resulting in an increased number of software being provided through it to the users.

There is a need for conducting research on whether it is possible to provide a framework of load balancing in the cloud preserving fault tolerance and low communication overhead. Multi-tier architecture based on Compute Ratio (CR) to ensure load balancing for the cloud without partitioning the data files of software running in it needs to be provided. More precisely, the following research question needs to be addressed:

- *How to solve load imbalance complications for software running in the cloud with minimum communication delay?*

Geographic partitioning of data centers and assignment of tasks to the partition based on job location to minimize load imbalances has been proposed in [1]. The authors stated that the best data center for the task will be chosen based on the geographic location of task initiation. However, the internal architecture changes that needs to be implemented when a specific region has a large number of tasks were not considered by the authors. Security of cloud computing has been ensured through journal based detection of malwares in distributed virtual machine (vm) instances [2]. The authors stated that the analysis of cloud journal files is a desirable procedure for detecting malicious entities. Moreover, the

*Corresponding author.

ICEMIS '15 Istanbul, Turkey

increased load on the cloud for handling the large array of journal files and the necessity to balance it has been considered to a limited extent by the authors. The need for a dynamic load partitioning and balancing mechanism for the cloud environment is stated in [3]. It is suggested that effective load balancing is needed for wide acceptability of cloud services, emphasizing the importance of resolving load imbalance issues in cloud. However, despite identifying the need, the authors did not propose a mechanism for solving the problem of service failures due to excessive load exerted on cloud vm-instances.

This paper proposes an effective 3-tier Load Balancing Schema (LBS) called CLBS-3 for software services running in cloud computing environments. The framework presents incoming jobs to vm-instances based on the calculation of CR as proposed in this research. The proposed algorithm considers throughput, response time, fault tolerance and service migration time as pre-specified benchmarks for determining the effectiveness of the load balancer [2]. Increased computing resources are allocated to the vm-instances conducting jobs with high resource demand as determined by the CR. Hence, a load balancing model for the cloud is provided with respect to the demand of workflows and data communication of the services being executed in the vm's. Additionally, the framework includes the amalgamation of the job fragments from different vm's to obtain the final service output. Due to three layered LBS, the communication across nodes and vm's are limited to only three tiers, resulting in reduced overhead and less network traffic.

Empirical analysis of the experimental results show the proposed LBS outperforms existing models in terms of fault tolerance and network latency. The algorithm is implemented in Openstack Juno cloud running in Ubuntu 14.04 LTS servers with 3 TeraByte (TB) storage. Software services namely online book-keeping and web enabled inventory management services are implemented in the cloud vm-instances for provisioning services to users. The cases of excessive loads on those are mitigated and balanced using the proposed scheme. Failure cases were detected for those and CR was used to assign supplementary vm-instances. The speed of recovery with fault tolerance was found to be 11.40 seconds at times of less load and 84.43 seconds during high load on the cloud for a job of 100 MB file processing for online inventory management system stated previously, yielding desirable results of less than 100 seconds for failure recovery time.

2. RELATED WORK

Xu *et al*. developed a load balancing model for the public cloud based on the cloud partitioning concept with a switch mechanism to choose different strategies for different situations [1]. In their proposed model, a controller is used that partitions a large public cloud based on geographical locations and assign jobs to the suitable cloud partitions. Load balancer chooses the best load balancing strategy among partition based on status information of every machine and then chooses the right strategy to distribute the jobs [4]. However, the authors do not show any performance analysis of their proposed model. Also, the proposed framework may result in the low utilization of resources inside the data centers due to resource stranding and fragmentation.

Provenance based malware detection mechanism for the cloud has been proposed in [2]. The authors provided a journal based analysis of cloud provenance to determine the presence of malicious entities such as viruses and worms in open source cloud computing environments. After detection, the recovering mechanism for the detection entity has been identified using a malware recovering dictionary. However, errors and faults caused in the dynamic cloud environment in the physical machines have been addressed to a limited extent by the authors.

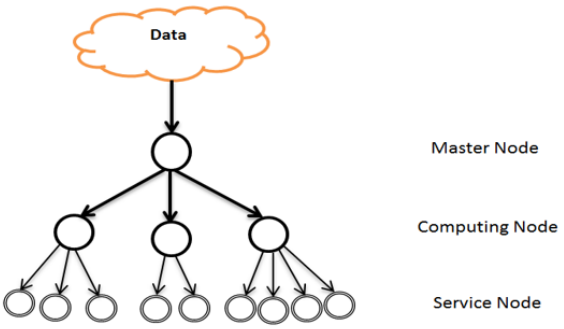

Figure 1: Node responsibility of 3-layered LBS for cloud

Run-time adaptation mechanism using redundant array of vm's for load-balancing and recovering failed nodes in a dynamic web based software running in cloud is provided in [5]. The authors highlighted the increasing need for ensuring effective load balancing for the cloud based Software as a Service (SaaS). However, the importance of addressing the increasing network overhead when implementing many-tier architecture for load balancing is considered to a limited extent. The Load Balancing Nearest Neighbor algorithm is stated for addressing load imbalance in cloud environments [6]. The authors proposed 6-tier framework of LBS to address the increasing load on cloud computing vm's instead of dividing the load into chunks. As stated before, the issue of network overhead for a large number of machine tiers to address the load imbalance has not been considered by the authors.

3. PROPOSED FRAMEWORK FOR LOAD-BALANCING IN CLOUD

The purpose of this research is to develop a load balancing model for the cloud environment which keeps track the growing volume of information, nature and workflows. Besides controlling large amount of data in the cloud, the challenge of migrating the data from overloaded processes to under loaded nodes transparently is equally important and to solve this issue a vm load balancing algorithm has been proposed here. After getting requests from clients the algorithm calculates each request's execution time on each node and then compares this time to resource usages threshold [2]. Memory wastage and saving costs are important issue of cloud computing [4]. The research aims to minimize execution time on parallel computing job and balance load among service node based on the nodes CPU usages, hard disk and CR.

As discussed in the previous section, developing a load balancing mechanism for the cloud is highly important in regard to the existing scenario. The multi-level hierarchical network topology to achieve load distribution can significantly decrease the data storage cost. However, too many

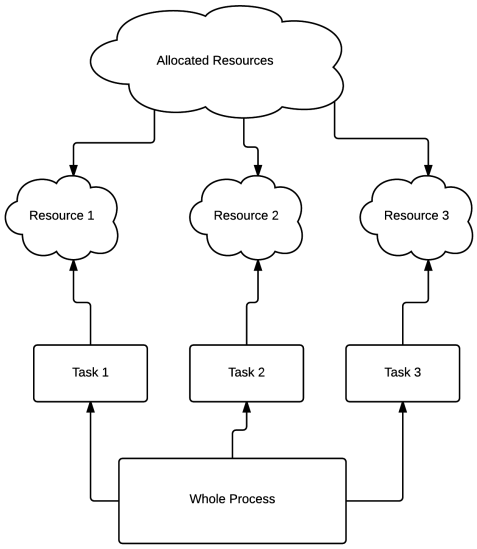

Figure 2: Cloud vm assignment based on the resource demands for the different jobs preserving fault tolerance

levels of load balancers will increase the network management cost and increase communication overhead. Therefore, in this paper, a three level load balancing schema (LBS) has been proposed that effectively balances load in cloud environments at low communication delay. Figure 1 provides the overview of the proposed LBS where user tasks are executed among the service nodes spread across the vm-instances of cloud. The three layers consist of master node, computing nodes and services nodes as described in the following:

1. *Computing Nodes: Those nodes are used to distribute the job across the service nodes.*

2. *Master Node: The master node is used to allocate the task to a compatible computing node. The master node is also responsible for the task of the cloud controller as it manages the physicals machines and uses virtualization technology to launch vm-instances.*

3. *Service Nodes: Those refer to the collection of vm-instances used to run the software in the cloud environments. Those are directly provided to the users and the load balancing framework leverages the power of huge array of vm's in cloud to ensure fault tolerance and integrity of the services.*

When a new user's job appears in LBS, fragments of the job are delivered to a service vm-instance. Therefore, the aim is to identify how data and jobs can be effectively allocated to suitable service nodes. Since all the nodes do not have the same computing capacity, an effective algorithm is proposed to identify the most applicable physical machine to assign an incoming job, with respect to server utilization and minimal communication overhead as presented in Algorithm 1. Such homogeneity assumption can decrease the performance of LBS. This paper is motivated to promote a data placement technique that can locate job to the most

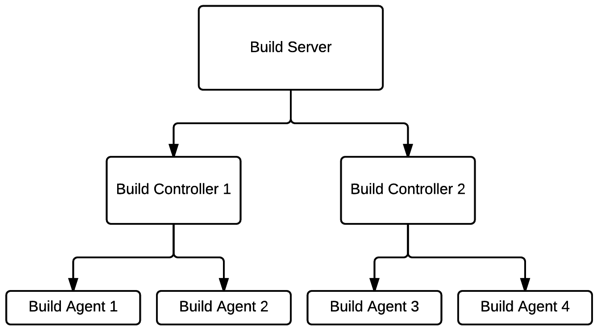

Figure 3: Framework of the proposed solution

competent service node and elevate the performance of the LBS [3].

In LBS, the CPU usage, Disk space, Processing Speed can very weightily for each of the service nodes. A high processing speed node can finish processing data faster than a low processing speed node in the cloud. Hence the transfer of data within nodes via network as proposed in traditional research will require huge amount of data migration for the cloud [2, 3]. When the amount of migrated data is huge, the overhead of demising unprocessed data from fast nodes to slow ones becomes a troublesome issue and negatively affects the performance of LBS.

The proposed framework boosts up the performance of the LBS through minimizing the data movement between slow and fast nodes. Migrating data is reduced through ensuring that the placement of data-files to a node is proportional to processing speed of the cloud vm-instance. Hence the CR of each node is determined and based on that the heterogeneity problem of service nodes was successfully solved as shown here.

The build process in Software Development Lifecycle basically concerns with the proper compilation of the code and making the project ready for deployment as highlighted in Figure 2. For conducting build process, a defined hierarchy needs to be followed for the cloud. The hierarchy for build process is illustrated in above illustration. In the top level there is build server. There could be single or multiple build servers for the build purpose. Each build server consists of one or more than one build controllers which defines the attribute of build agents. Build agents generally keep the whole process going. The application and implication of build process is carried out by build agents. Generally there are more numbers of build agents present in the whole process than build controllers.

Cloud computing assures the dynamic storage of data and resources. Build servers are generally physical servers with known vulnerabilities such as single point of failure, constant data loss, costly maintenance and unwanted queuing in case of resource unavailability. If cloud computing is used to host the data and resources necessary for usage of build server, physical build server can be eliminated from the whole build process. This will result into faster performance, efficient data storage and cost reduction. This will also eradicate the single point failure problem from the whole build process.

In this research, build process is first divided into several tasks. Those tasks are prioritized based on the importance and contribution in the process flow. The resources needed

Algorithm 1 Algorithm for 3-layered load balancer in cloud

procedure 3-TIER LBS(VM_{Info}, $Data_{fine}$,$count_job$)
  $VM_{Info} \leftarrow VM_{ID}, Service_{node}$
  $CR \leftarrow T_{end} - Tstart$
  $CR_t \leftarrow CR/count_jobs$
  while $CR_t <= Load(job)$ **do**
    Assign T_i *to* VM_{ID}
    $count = count + 1$
  end while
  If $len.Message < len.M_{pr}$, then
  Return VM assigned for $Load_i$
end procedure

to perform those tasks will be hosted in the cloud which will act as a build server for the process. Required build controllers and build agents to carry out the build process will be available in Cloud. Tasks will be assigned to the required resources based on complex method. After performing the activity, resources will be freed instantly for other usage. This kind of approach will annihilate redundancy problem of physical build processes.

4. ANALYSIS OF OBTAINED RESULTS

CR symbolizes how much time each service node incurs to complete a unit task. The performance measure is identified through processing speed of node depicted as Speed in Table I. To fairly calculate processing speed, firstly equal amount of data to each node of LBS was assigned. Hence, each node will process same amount of data. Secondly, after accomplishing the task processing, response times of each service node are recorded. Thirdly, the shortest response time is determined using the measurement scale and the data are allocated to nodes based on that.

$$CR(t) = \int_{a=0}^{100} \left\{ \frac{x_{end} - x_{begin}}{t_i} \right\} \quad (1)$$

The CR for each of the nodes is calculated using the equation 1. Hence, based on the CR, the nodes are selected and data files together with the tasks and subtasks are run on specific vm-instances. Finally, the computing nodes deliver the fragments of the task to service nodes so that all service nodes complete their assigned actions within almost the same time without incurring faults, thereby achieving effective load-balancing. For experimentation, the following case is considered. Here, four service nodes were assumed to be under the computing nodes in LBS namely Node_A, Node_B, Node_C and Node_D respectively and each are assigned to process a total of 1 Giga Byte (GB) data. The goal is to distribute 1GB task within service nodes so that all nodes complete processing of the respective tasks in almost the same amount of time.

After processing task into service nodes, it has been found that the response time of service nodes A, B, C, D are found to be 10, 30, 40, and 20 seconds respectively. Here, response time of node A is smallest and node C is largest. Therefore, the CR of node A is 1 which is used to reference the CR of other nodes. So that, CR of B, C, and D service nodes are 3, 4 and 2 respectively. So it is seen that CR of service nodes A, B, C and D are quite stable.

Additionally there is no need to calculate the CR of each

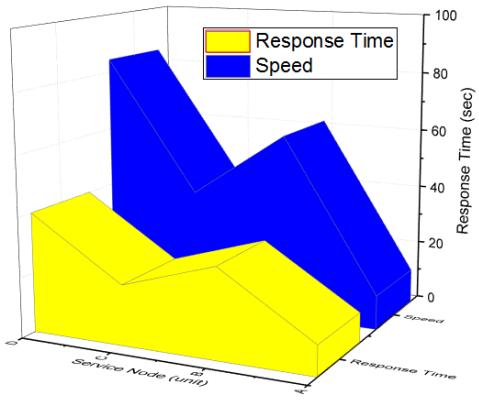

Figure 4: Comparison between Response Time and Speed of task completion for the four test nodes in the cloud

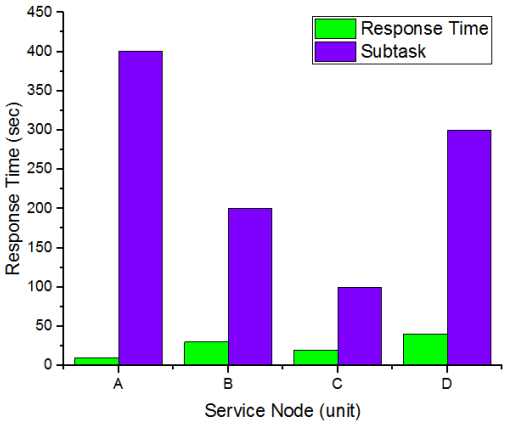

Figure 5: Analysis of recovery from failure using the proposed algorithm for the 4 test nodes

service node again as this extra action can be prevented by keeping the record of each service node into an array. When another task is being executed, again there is no need to calculate the CR. However, based on the previously recorded CR subtasks are assigned into service nodes. If the CR is calculated accurately then almost all of the service nodes below the computing nodes processes task at almost same amount of time as found in experimentation. Table 1 shows the *Response Time*, CR and *Subtask Allocated* to those vm-instances. Here, it can be seen that the fastest node processes 400MB subtask and slowest node processes 100MB subtask out of 1GB task.

The relationship between the *Response Time* and *Speed* of task completion by the Service nodes of the cloud environment are depicted in Figure 1. As shown in the figure, the Response Time is proportional to the Speed of task completion, emphasizing the assumption of using the compute ratio for effective node selection during task assignment. As seen in the figure Node_a consists of the best Speed of task completion and can respond to a greater number of tasks within a given time compared to the other nodes due to its higher physical resource allocation for its vm-instances. The

Table 1: Tabulation of experimental outcome for the proposed load balancing scheme

Service Node	*Response_Time*	*Computing_Ration*	*Allocated − Subtask*	*Speed*
$Node_a$	10	1	400 MB	11.40
$Node_b$	30	3	200 MB	40.00
$Node_c$	40	4	100 MB	62.09
$Node_d$	20	2	300 MB	84.43

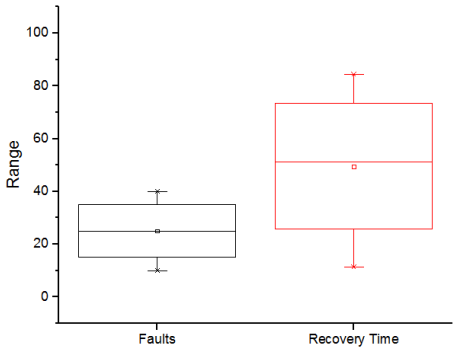

Figure 6: Identification of failure cases and recovery time

number of tasks completed and the time taken to complete each of those if shown in Figure 5. Similar to Figure 1, here it is seen that $Node_a$ has the best performing out of all the nodes due to the implementation of the node selection algorithm proposed here and due to implementation of 3-tier LBS as compared to many-tiers of previous research efforts.

Figure 6 highlights the number of cases of faults that occurred during the period of 3 months that required load balancing and the average time taken for solving each of those. As seen in Figure 6, 86% of faults were detected and load for those were balanced in the cloud within 40 - 60 seconds upon detection, with few exceptions ranging up to 83 seconds that is still below the desirable benchmark of 100 seconds. Hence the desirable performance of the proposed framework can be determined from the graphical representation of experimental results.

5. CONCLUSION

The paper presented an effective 3-tier load absorbing framework named CLBS-3 for mitigating load imbalance of cloud computing. The aim was to identify the services nodes that are available for executing a new task and determine the one with the least node amongst the available vm-instances. Next, the goal was to ensure minimum network and communications overhead for the cloud environment whilst ensuring dynamic load balancing.

It was exhibited via experimentation that the proposed mechanism of load-balancing effectively reduces network overhead and increases the capability of the vm-instances to complete an increased number of task within a specific time. Also the 3-tier LBS is easy to implement and ensure user-friendliness in open source cloud environment. Analysis of the experimental results show that the node of the cloud where the proposed mechanism is implemented outperforms the other nodes in terms of speed for service delivery ranging to about 11.40 seconds as compared to 40.00, 62.09 and 84.43 seconds respectively for traditional chunk-data oriented load balancers. Traditional load optimizers aim to separate data files into chunks and add header information to each chunk that is used to amalgamate the data file once load imbalance is mitigated. However, such mechanisms yield higher communication delays as discussed in the earlier sections of this paper.

The proposed framework achieves load-balancing of cloud vm-instances and ensures fault tolerant functioning of the software running in the cloud environment through implementation of CLBS-3. Improving the proposed methodology through incorporation of load balancing technique for multiple databases residing in cloud opens a scope for future research.

6. ACKNOWLEDGMENTS

This research has been supported by the University Grants Commission, Bangladesh under the Grant No-Reg/Admn-3/2015/48743.

7. ADDITIONAL AUTHORS

Kazi Salatuzzaman (IIT, University of Dhaka,email: kmsalatuzzaman@gmail.com)

8. REFERENCES

[1] Gaochao Xu, Junjie Pang, and Xiaodong Fu. A load balancing model based on cloud partitioning for the public cloud. *Tsinghua Science and Technology*, 18(1):34–39, 2013.

[2] Asif Imran, Alim Ul Gias, Rayhanur Rahman, and Kazi Sakib. Provintsec: a provenance cognition blueprint ensuring integrity and security for real life open source cloud. *International Journal of Information Privacy, Security and Integrity*, 1(4):360–380, 2013.

[3] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, 2008.

[4] Rogério De Lemos, Holger Giese, Hausi A Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M Villegas, Thomas Vogel, et al. Software engineering for self-adaptive systems: A second research roadmap. pages 1–32, 2013.

[5] Asif Imran, Alim Ul Gias, and Kazi Sakib. An empirical investigation of cost-resource optimization for running real-life applications in open source cloud. In *High Performance Computing and Simulation (HPCS), 2012 International Conference on*, pages 718–723. IEEE, 2012.

[6] Ms Radha G Dobale and RP Sonar. Load balancing in cloud. *International Journal of Engineering Research and General Science*, 3(3):160–167, 2015.