BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **15**, No 4

Sofia • 2015

Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2015-0060

OLB: A Nature Inspired Approach for Load Balancing in Cloud Computing

B. Mallikarjuna[1], *P. Venkata Krishna*[2]

[1] *Bharathier University, Coimbatore, TamilNadu, India*
[2] *Dept of SCSE, VIT University, Vellore, TamilNadu, India*
Emails: mail2mallib@gmail.com dr.krishna@ieee.org

Abstract: *Load balancing is treated as one of the important mechanisms for efficient resource allocation in cloud computing. In future there will appear a necessity of fully autonomic distributed systems to address the load balancing issues. With reference to this, we proposed a load balancing mechanism called Osmosis Load Balancing* (*OLB*)*. OLB works on the principle of osmosis to reschedule the tasks in virtual machines. The solution is based on the Distributed Hash Table* (*DHT*) *with a chord overlay mechanism. The Chord overlay is used for managing bio inspired agents and status of the cloud. By simulation analysis, the proposed algorithm has shown better performance in different scenarios, both in heterogeneous and homogeneous clouds.*

Keywords: *Load balance, virtual machines, osmosis, cloud, relocation.*

1. Introduction

Cloud computing is treated as one of the emerging trends in distributed computing. It has changed the way of computing from small PCs to large data centers [1]. The cloud plays a major role in providing services to Internet operating remotely, and also provides a cost efficient computation of the requirements. The major services of cloud computing to IT industry is providing the infrastructure so it helps the

organizations to reduce the cost of installing software and hardware [2, 3]. Referable to the immense development in cloud computing, many industries are adapting this environment. Thus, it causes some complicated issues in migrating resources from industries to cloud computing environment. Load balancing, virtual machine migration and server consolidation are the traditional issues which are not fully solved up to today. Virtual machine migration handles the virtualization by balancing the load over the virtual machines and also allocates an efficient resource provisioning over the virtual machines [9-11]. Server consolidation deals with efficient utilization of resources by combining the VMs which are residing in the initialized servers into a single server. The load on VMs is efficiently managed by a load balancing mechanism.

Load balancing is treated as a very big issue in cloud computing. The even distribution of workload over VMs is carried out by load balancing in the cloud. It serves to attain to a high client fulfillment and asset usage proportion, henceforth enhancing the resource utility and general execution of the framework. This mechanism likewise guarantees that each computing resource is dispersed productively and decently. As a next step, it anticipates pitfalls of the framework which may happen because of imbalance of the load, when single or multiple components of a service come up short, load balancing allocates or discards the applications of the resources without any fail. In addition, it ensures that each computing resource is dispersed proficiently and reasonably. So as to backing on-demand provisioning, the infrastructure of the cloud is liable to end up extensive scale heterogeneous platforms with small providers coinciding with bigger ones [20]. This movement will naturally include the improvement of new systems for giving completely distributed task allocation. To handle this issue, in this paper we introduce a completely distributed VM load balancing that utilizes a bio-inspired method to give self-ruler and self-sorted out operation [19]. At the most, our solution is focused around ant inspired agents and the load balancing of tasks among VMs by taking over the rule of osmosis.

The rest of the paper is organized as follows: Section 2 deals with the study of load balancing approaches with respect to the nature inspired. Section 3 discusses the OLB model and related algorithms. Section 4 represents the simulation setup. Section 5 deals with discussion of the results and finally a conclusion is given in Section 6.

2. Literature review

Randles, Lamb and Taleb-Bendiab [4] introduced a nature inspired approach called honey bee based load balancing mechanism. The model concentrated on server activities for load balancing and the performance of the system is improved by an expanded framework. The throughput of the model does not impact the system size and it is well suited for the conditions where the service requests are more. Zhang and Zhang [5] concentrated on an ant colony mechanism which focused on network hypothesis in a distributed environment. It concentrated on small organizations to achieve load balancing [13]. This

mechanism is efficient in heterogeneous environments, adaptable to dynamic situations and it contains an efficient fault tolerance mechanism. Bhadani and Chaudhary [6] has developed a load balancing mechanism based on central policy which concentrates on the global state information for load balancing decisions. It works well in improving the performance, but is not concentrated on the fault tolerance. Stanojevic and Shorten [7] developed a combinational approach containing both load balancing and distributed rate limiting approach. It contains an efficient mechanism for cost minimization and allocation of resources [12]. Singh, Korupolu and Mohapatra [8] implemented VectorDot approach for data centers with storage virtualization and server integration.

3. Osmosis load balancing model

The Osmosis Load Balancing model (OLB) reallocates tasks among a set of virtual machines keeping in mind the end goal to adjust the load. The load balancing procedure is totally decentralized and is underpinned by the ant like agents that are executed by data centers on a Chord overlay. Every data center deals with a queue and can execute one or more tasks simultaneously with diverse execution qualities. We consider that the tasks are non-preemptive, individual and non-divisible. Our solution helps both homogeneous and heterogeneous cloud environments: in a completely homogeneous situation all tasks have the same prerequisites, and every data center is relied upon to have the capacity to execute any task (with the same run time execution) [15]. Alternately, in a completely heterogeneous lattice, each task has diverse necessities that can be satisfied just by some of the data centers. In the following sections we detail the operation of osmosis for cloud environments.

3.1. Distributed hash table with a chord overlay

The chord overlay uses the distributed hash table to organize the VMs. The overlay is organized with a ring like structure which consists of virtual machines, each VM is assigned a unique identifier. Based on their identifier, VMs are organized in order. The information of each identifier is maintained in the finger table with the addresses of each VM to quickly migrate the task from one VM to another VM for a load balancing process [14]. The chord overlay migrates the tasks in VMs with the time complexity $O(\log N)$.

3.2. Task execution on local and remote data centers

The problem of load balancing is defined as distribution of tasks over VMs in an equal manner. Here we consider that s data center with VMs has different performance [16]. So, the concentration of each VM was given in time to execute the whole queue. The concentration of VM is calculated by

$$\rho_{\mathrm{VM}} = \frac{\sum_{i \in T} i_{\mathrm{et}}}{\vartheta \times \mu}, \tag{1}$$

where T is the set of all allocated tasks, i_{et} is the estimated completion time of each task in i seconds, ϑ is the speed of the VM and μ is the maximum number of tasks that are currently executed.

3.3. Ant like agents

The goal of each VM is to minimize the load on its side and transform the load to its neighbors. To fulfil this, the osmosis procedure introduces three types of agents. These agents have access to the information like index values of VMs, predecessor and successor information of VMs, ability of the VMs to execute the tasks and load of the VMs.

3.3.1. Identification agent

The main role of the identification agent is to inform the performance characteristics and potential of VMs within the ring and data center [18]. The data center sends the identification agent to VMs periodically to the predecessors, successors, and to the randomly selected VMs. The addresses of predecessors and successors of VMs are identified by using the finger table. From Algorithm 1 (adopted from [21]), it is observed that the agent is assigned a target VM to migrate to (it is either a predecessor α, successor β, and the random one γ). The agent recognizes the local information about the VMs and performance information and subsequently migrates to the target VM, where this information is updated. The functionality of the identification agent is based on two factors: on one side, it is to identify the information about the load of the VMs and performance and on the other side it has to migrate the loads of VMs by identifying VMs based on load calculation.

Algorithm 1. Identification agent

Begin
Let: η be the current VM;
Let: τ be the target VM ;
Let: migrate(τ), function for migration of tasks to VM τ;

1. χ := η
2. ρ := ρη
3. δ := perfη
4. migrate(τ)
5. if χ = α then
6. ρα := ρ
7. perfα := δ
8. else
9. if χ = β then
10. ρβ := ρ
11. perfβ := δ
12. else

13. γ := ρ
14. ργ := ρ
15. perfγ := δ
16. end if
17. end if

End

3.3.2. Osmosis agent

The osmosis agent is responsible for migrating the tasks from VMs with higher load to VMs with lower load. Each VM identifies the osmosis pressure of other VMs from time to time by computing the differences between the load of VMs and that of each VM $n \in \{\alpha, \beta, \gamma\}$. The diffusion of the load is

(2) $$\rho_{\mathrm{VM}} - \frac{\rho_n}{\mathrm{per}f_n}.$$

VM χ is identified as the highest positive rate and it is chosen as a candidate for the load balancing process.

Algorithm 2. Osmosis agent

Begin

Input: Let T be the set of tasks to be migrated , ω be the direction to migrate (either α, β, γ), V be the maximum number of allowed VMs in the ring.

Let: η be the current VM;

Let: migrate(τ), function for migration of tasks to VM τ;

Let: next(ω), the following VM in the ω direction;

Let: execute(t), executes task t on the current VM;

1. migrate(next(ω))
2. if ω = γ then
3. if then
4. ω := α
5. else
6. ω := β
7. end if
8. end if
9. while $V > 0$ then
10. $V := V–1$
11. if then
12. migrate(next(ω))
13. else
14. break
15. end if
16. end while
17. For all tasks $\in T$ do
18. Execute (t)
19. End for

End

Algorithm 2 (adapted from [21]) explains an osmosis agent, in which the agent is assigned with a set of tasks and directions given. The agent identifies the target VM, it checks the local concentration. If the local concentration of the target VM is less than the successor VM, it drops the task and executes the task in the target VM. In this regard, the osmosis agent can migrate a number of steps in the ring and finally it leads to the load balancing process.

3.3.3. Relocation agent

In the data centers, the tasks may be submitted to inappropriate VMs whose profile is not suitable to execute the task. The role of the relocation agent is used to resubmit the tasks to the queue of another virtual machine in an appropriate group [17]. The relocation agent is explained in Algorithm 3 (adapted from [21]). With respect to the osmosis agent, the reallocation mechanism follows the key based routing method to directly allocate the VM in an appropriate group.

Algorithm 3. Relocation agent

Begin

Input: Let RA be the list of tasks to be relocated, t - the group of the tasks to be reallocated;

Let: η is the current VM;

Let: key(ρ), migrate to the first VM in group ρ with key based routing;

Let: Group (t) returns the tasks within group;

Let: execute(t), execute the task t on the current VM;

1. key(tgroup)
2. for all task $\in$ tasks do
3. execute(task)
4. End for

End

4. Simulation setup

In this experiment we set up 75 VMs on a chord overlay. The tasks are introduced when all the VMs are connected to the overlay. VMs transfer the concentration of the load to the remaining VMs in every 30 seconds, the osmosis is performed in every 60 seconds, the reallocation is performed in every 120 seconds on an average. A detailed analysis is yet to be carried out on the parameters and left for future work.

4.1. In homogeneous environmnet

In a homogeneous environment all VMs are grouped into one single group and have equal properties and the capability to execute any type of a task with at most one task at a time.

4.2. In heterogeneous environment

In heterogeneous environment the VMs and tasks have different types of characteristics. The executing performance of each VM is different from the other

one. Tasks execution time will be different when compared to homogeneous environments.

4.3. Parameters to evaluate the OLB

The load balancing measurement in the algorithms is calculated by using standard deviation. In heterogeneous environments, the individual VMs standard deviation is considered and maximum value of the standard deviation taken into account whereas in homogeneous environment the value is calculated for overall VMs. The algorithm performance is calculated by the number of tasks migrating from VMs that we commonly called task reallocation.

5. Result analysis

In this section we are going to deal with the result analysis of the proposed algorithm. All the values are concerned with the simulation environment. In Fig. 1, we can observe that the standard deviation rapidly converges below 80% in all scenarios. It is interesting to observe that the homogeneous one obtains better results when compared to heterogeneous environments in Fig. 2.

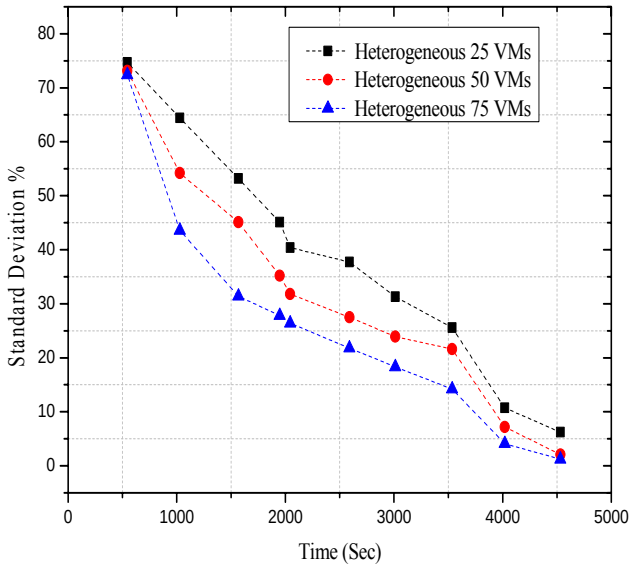

Fig. 1. Load balancing in heterogeneous environment

In both heterogeneous and homogeneous environments, the model is tested with 25, 50 and 75 VMs and their standard deviation is measured along with time. The obtained results are shown in Figs 1 and 2.

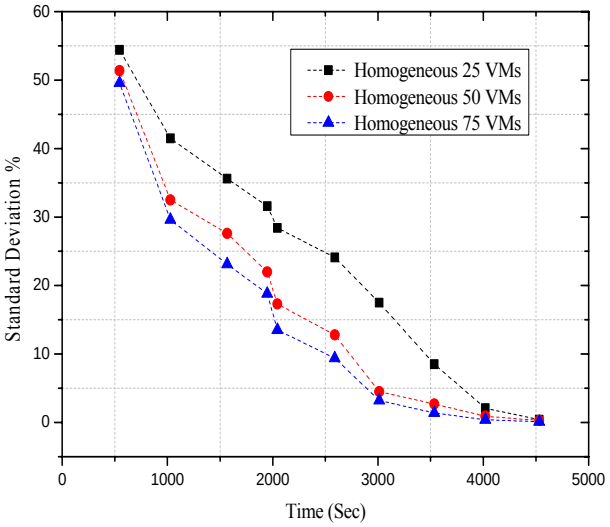

Fig. 2. Load balancing in homogeneous environment

Fig. 3 indicates the scalability of the model which is related to both homogeneous and heterogeneous models. Interestingly, the higher value of tasks improves the convergence of the standard deviation. In the homogeneous environment the migration of tasks is very low when compared with heterogeneous environments.

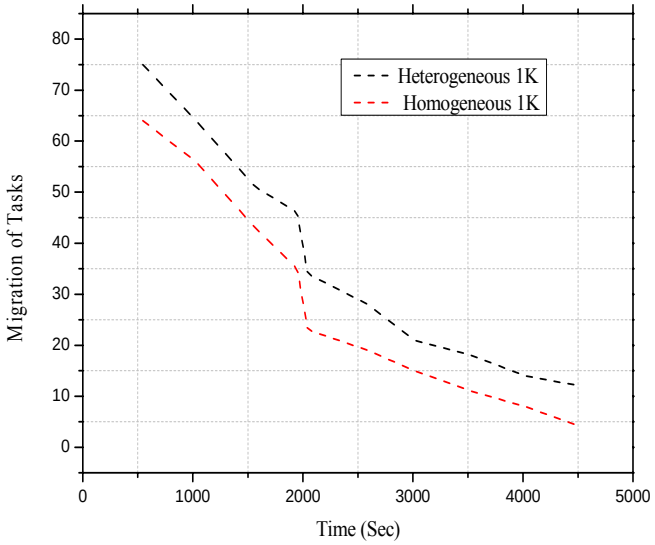

Fig. 3. Scalability measurement

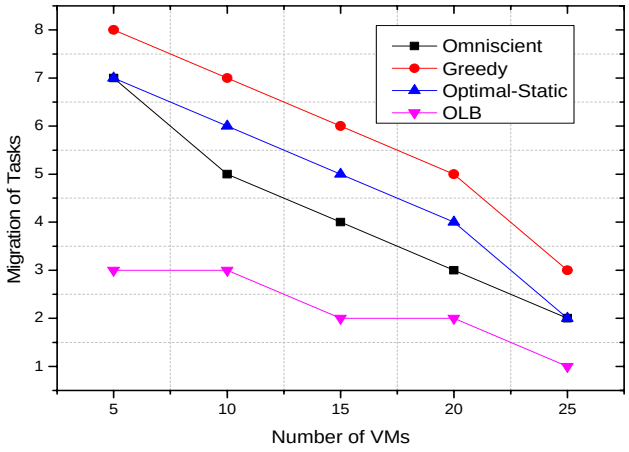

Fig. 4. Number of task migrations over VMs when the tasks are 20

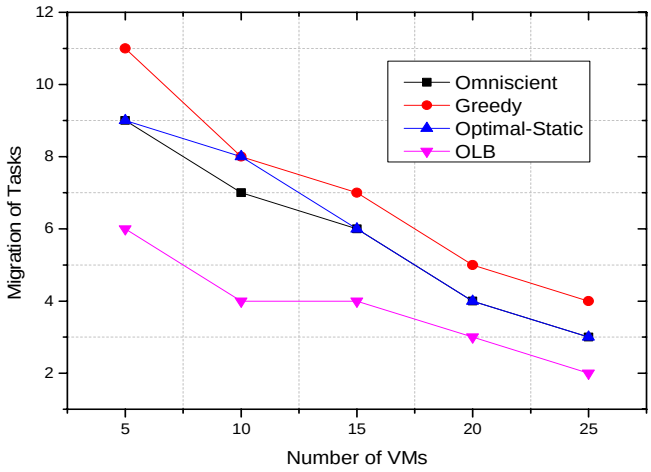

Fig. 5. Number of task migrations over VMs when the tasks are 40

6. Conclusion

In this paper we proposed an osmosis mechanism for load balancing of tasks among VMs in both homogeneous and heterogeneous environments. The proposed method, is helpful to reallocate the tasks among adjacent VMs with the help of a chord overlay.To distribute the information about the VMs, as well as to reallocate the tasks, bio-inspired agents are applied. The load balancing algorithm is evaluated in both homogeneous and heterogeneous environments. The performance of the algorithm is tested by different existing algorithms and it exhibits better results.

References

1. Armbrust, M., A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. EECS Department, University of California, Berkeley, Technical Report No UCB/EECS-2009-28, February 2009, pp. 1-23.
2. Lucky, R. W. Cloud computing. – IEEE Journal of Spectrum, Vol. **46**, May 2009, No 5, pp. 27-45.
3. Dikaiakos, M. D., G. Pallis, D. Katsa, P. Mehra, A. Vakali. Cloud Computing: Distributed Internet Computing for IT and Scientific Research. – IEEE Journal of Internet Computing, Vol. **13**, September/October 2009, No 5, pp. 10-13.
4. Randles, M., D. Lamb, A. Taleb-Bendiab. A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing. – In: Proc. of 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, Perth, Australia, April 2010, pp. 551-556.
5. Zhang, Z., X. Zhang. A Load Balancing Mechanism Based on Ant Colony and Complex Network Theory in Open Cloud Computing Federation. – In: Proc. of 2nd International Conference on Industrial Mechatronics and Automation (ICIMA), Wuhan, China, May 2010, pp. 240-243.
6. Bhadani, A., S. Chaudhary. Performance Evaluation of Web Servers Using Central Load Balancing Policy over Virtual Machines on Cloud. – In: Proc. of 3rd Annual ACM Bangalore Conference (COMPUTE), January 2010.
7. Stanojevic, R., R. Shorten. Load Balancing vs. Distributed Rate Limiting: A Unifying Framework for Cloud Control. – In: Proc. of IEEE ICC, Dresden, Germany, August 2009, pp. 1-6.
8. Singh, A., M. Korupolu, D. Mohapatra. Server-Storage Virtualization: Integration and Load Balancing in Data Centers. – In: Proc. of ACM/IEEE Conference on Upercomputing (SC), November 2008.
9. Reddy, T. S. K., P. V. Krishna, P. C. Reddy. Power Aware Framework for Scheduling Tasks in Grid Based Workflows. – Int. J. Communication Networks and Distributed Systems, Vol. **14**, 2015, No 1, Inderscience Publishers, pp. 74-88.
10. Kalaiselvan, K., P. Venkata Krishna. Performance Based QoS for Cloud's Infrastructure as a Service. – International Journal on Cloud Computing, Vol. **2**, 2013, No 4, Inderscience Publishers, pp. 325-339.
11. Tekriwal, N. Madhumita, P. Venkata Krishna. Integration of Safety and Smartness Using Cloud Services – An Insight to Future. Innovations and Advances in Computer, Information, Systems Sciences, and Engineering. – In: Lecture Notes in Electrical Engineering, Vol. **152**. Springer, 2013, pp. 293-303.
12. Varma, M. Krishna, Eumni Choi, P. Venkata Krishna. Ontology Framework with Semantic Information for Product Based Applications. – GESTS International Trans. on Computer Science and Engineering, Vol. **63**, February 2011, No 1, pp. 27-36.
13. Dhinesh, Babu L. D., P. Venkata Krishna, A. M. Zayan, V. Panda. An Analysis of Security Related Issues in Cloud Computing. – Contemporary Computing, Communications in Computer and Information Science, Vol. **168**, 2011, Part 2, pp. 180-190. DOI: 10.1007/978-3-642-22606, pp. 9-21.
14. Kalaiselvan, K., P. Venkata Krishna. Grid to Cloud (G2C) – A Infrastructure Based Transition. – CSI Communications, Vol. **33**, February 2010, Issue 11, pp. 22-25. ISSN 0970-647X.
15. Pandey, P., P. Venkata Krishna, B. Sarojini. QoS Aware Healthcare System on Mobile Clouds. – In: Proc. of 2014 World Congress on Computing and Communication Technologies (WCCCT), IEEE, 2014, pp. 154-157.
16. Babu, M. Rajasekhara, P. Venkata Krishna, M. Khalid. A Framework for Power Estimation and Reduction in Multi-Core Architectures Using Basic Block Approach. – Int. J. Communication Networks and Distributed Systems, Vol. **10**, 2013, No 1, Inderscience Publishers, Netherlands, pp. 40-51.

17. K r i s h n a, P. V., S. M i s r a, D. J o s h i, M. S. O b a i d a t. Learning Automata Based Sentiment Analysis for Recommender System on Cloud. – In: Proc. of 2013 IEEE International Conference on Computer, Information, and Telecommunication Systems, CITS'2013, 7-8 May 2013.
18. B a b u, M. R a j a s e k h a r a, P. V e n k a t a K r i s h n a, M. K h a l i d. Optimization Techniques and Performance Evaluation of a Multithreaded Multi-Core Architecture Using OpenMP, LNCS Series of Computer Communication and Information Science (CCIS) 190, 2011, Berlin Heidelberg, Springer-Verlag, 2011, pp. 182-191.
19. M i s r a, S., P. V. K r i s h n a, K. K a l a i s e l v a n, V. S a r i t h a, M. S. O b a i d a t. Learning Automata-Based QoS Framework for Cloud IaaS. – IEEE Transactions on Network and Service Management, Vol. **11**, 2014, Issue 1, pp. 15-24.
20. B a b u, L. D., P. V. K r i s h n a. An Execution Environment Oriented Approach for Scheduling Dependent Tasks of Cloud Computing Workflows. – International Journal of Cloud Computing, Vol. 3, 2014, No 2, pp. 209-224.
21. B r o c c o, A. Ozmos: Bio-Inspired Load Balancing in a Chord-Based P2P Grid. – In: Proc. of 3rd Workshop on Biologically Inspired Algorithms for Distributed Systems, 2011, pp. 9-16. ISBN: 978-1-4503-0733-8.