A Survey on Load Balancing Techniques in Cloud Computing Environment

J.K. Verma
School of Computer & Systems Sciences,
Jawaharlal Nehru University, New Delhi, Delhi, India

C.P. Katti
School of Computer & Systems Sciences,
Jawaharlal Nehru University, New Delhi, Delhi, India

Abstract- Cloud computing is a new paradigm for hosting applications on virtualized resources. Services of cloud computing are delivered by millions of servers either located at geographically apart locations or located in a large scale data center. But the load among the servers is not in a justified manner. Load balancing techniques are used to solve the problem of distributing the load among the servers for fulfilling the various goals of load balancing. This paper presents the survey on five selected load balancing techniques based on different parameters.

Keywords – Load balancing, cloud computing, federation.

I. INTRODUCTION

Load Balancing (LB) is the method of assignment of work to processors. LB in clouds is a mechanism to distribute excess dynamic local workload evenly across all the nodes of the system. The amount of workload is assigned to each processor is balanced so that some processor do not sit idle while others are executing the task. LB techniques are used to achieve a high user satisfaction and resource utilization ratio so that no single node is overwhelmed due to excess workload, hence, improves the overall performance of the underlying system. Proper load balancing technique can help in utilizing the available resources in optimum way, thereby minimizing the resource wastage. It also helps in coping up with fail-over, enabling scalability, avoiding bottlenecks and over-provisioning, reducing response time etc. [1].

Energy efficiency is an important aspect of cloud computing. J.K. Verma et al. [2], [3] focused upon energy efficient (Green computing) techniques for cloud computing environment which is also a form of load balancing. Green computing or energy efficient computing is the practice of implementing techniques and procedures in such a way that improve the efficiency of computing resources/components and reduce the energy consumption that result into lower environmental impact of their utilization. Apart of the factors aforementioned, load balancing helps in avoiding overheating or creation of hotspot by balancing the workload among all the nodes of a cloud system, and cut downs the air conditioner cooling cost of components deployed. Energy consumption and carbon emission go hand in hand. The more is the energy consumption, higher will be the carbon footprint. As the energy consumption is reduced with the help of load balancing, so is the carbon emission that helps in achieving the goal of Green computing.

The rest of the paper is organized as follows. Section II classifies LB techniques and Section III presents summary of existing LB techniques followed by performance evaluation metrics in Section IV. Section V concludes the paper.

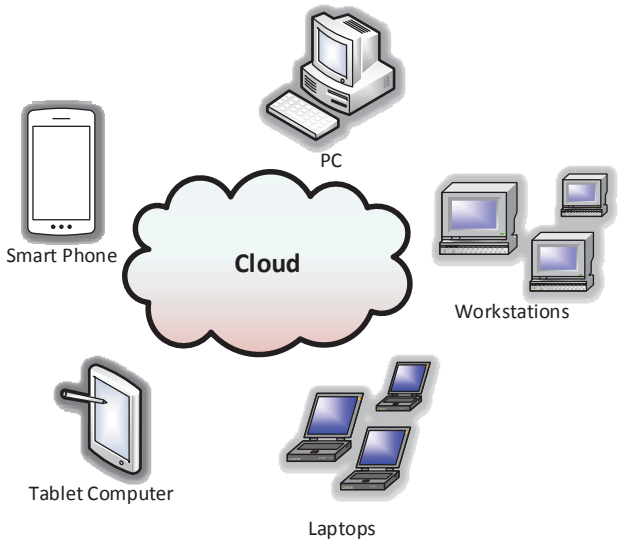

Figure 1: Schematic Representation of Cloud Computing

II. Classification of Load Balancing Techniques

A. Static LB Techniques –

Static LB techniques are used to minimize the overall execution time of concurrent programs while minimizing the communication delays. Krueger and Livny [4] has shown that load balancing are able to reduce the mean and standard deviation of task's response times. Static LB techniques distribute the workload among the servers prior to the execution of an algorithm. Servers participating in cloud computing framework has a certain specified capability to serve the requests directed to get services. Static LB techniques allocate tasks to these servers/hosts based only on the ability to serve the new requests which is already known prior to the redirecting the requests. The capability of the hosts is defined in terms of processing power, memory, and storage. Although knowledge of host capability is already known, the static techniques are not able to accommodate any dynamic changes in the attributes during the runtime. On the other hand, change in workload cannot be adopted by these techniques during the runtime. In applications with the constant workload, static LB can be used as per-processor to the computation. The notable examples of static LB techniques are Round Robin Algorithm, Randomized Algorithms, Recursive Bisection, Simulated Annealing, and Genetic Algorithm.

B. Dynamic LB Techniques -

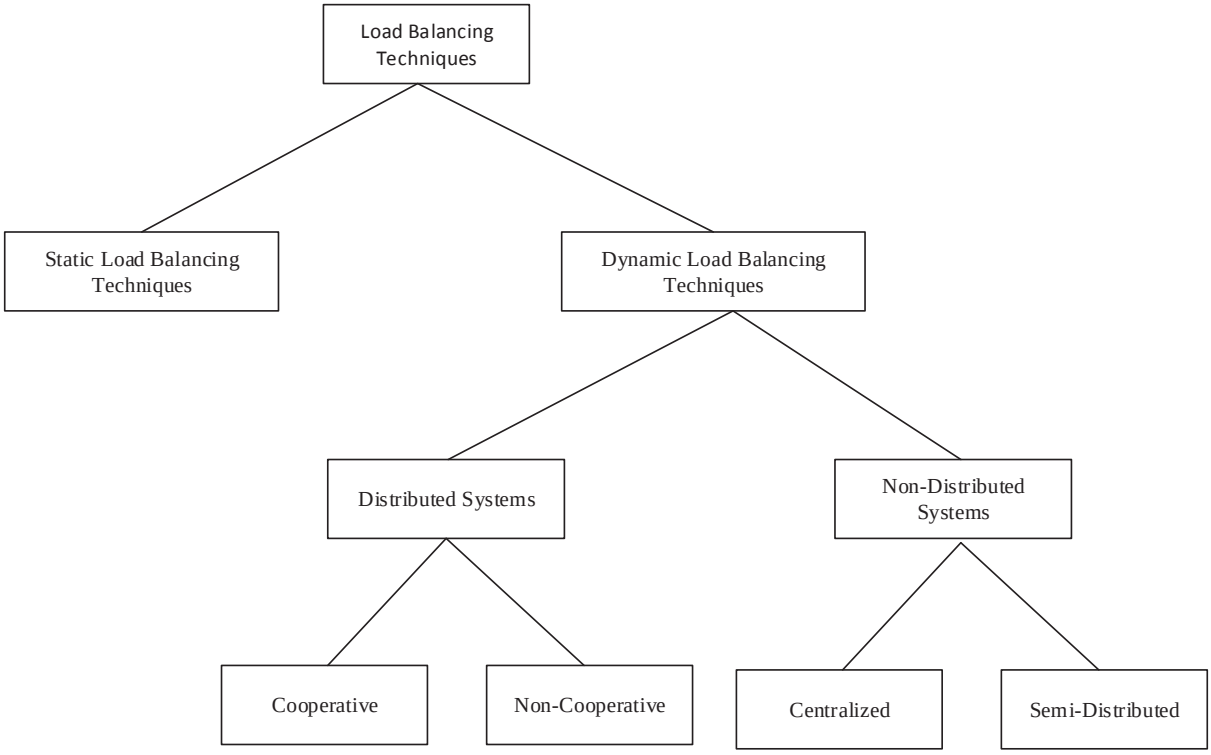

Figure 2: Classification of Load Balancing Techniques

Applications with adaptive finite element methods which have unpredictable and dynamic workload that changes during the computation require dynamic load balancers. Dynamic load balancers adjust the decomposition as the computation proceeds forward. Dynamic LB techniques distribute the workload among the servers during the runtime of the algorithm. Dynamic LB techniques assign and reassigns the tasks on hosts dynamically based on the information of attributes gathered and calculated. These techniques require constant monitoring and they are harder to implement. Dynamic LB techniques usually results into efficient load balancing and can be achieved through following two approaches:

a. Distributed systems.

A distributed system is a collection of computing and communication resources shared by the active users. Distributed systems do not have centralized control over other computing resources. Therefore, the load balancing algorithm executes on all the hosts/servers present in the underlying system. For the execution of the workload allocated, these computing resources interact with each other. The interaction among them can be of two types; (i) Cooperative, and (ii) Non- Cooperative.

In cooperative interaction, hosts work side-by-side to achieve the common objective, for instance, improving overall response times. On the other hand, hosts work independently towards the accomplishment of goal local to the host in non-cooperative interaction, for instance, minimizing the response time for the task local to the host. Dynamic load balancing techniques are more useful in a system that consists a network of the workstation where primary performance goal is to maximize utilization of the processing power instead of minimizing execution time of the application [5].

The advantage of distributed load balancing techniques is that in case of failure of any node the whole system does not go down. However, distributed techniques generate a lot of messages for interaction among computing resource.

b. Non-distributed systems.

In non-distributed systems, the task of load balancing is achieved through either one node or a group of nodes. Non-distributed dynamic load balancing algorithms take two forms: (i) Centralized, and (ii) Semi-Distributed. In the first scheme, LB algorithm is executed by only a single node of the system i.e. in Central Mode. The centralized node is solely responsible for load balancing of whole the system and other nodes interact with the central node only to exchange the messages. In the semi-distributed scheme, nodes of the underlying system grouped into cluster where load balancing takes place in each cluster in the form of centralized scheme. The centralized node is elected by appropriate election technique in each cluster to take care of load balancing in each cluster. However, being centralized on or another level causes exchanges of a lot of messages that causes overhead to the system. On the hand, if the centralized node crashes outs then the whole system goes down. Therefore, this scheme is most suitable for smaller networks.

III. Selected Load Balancing Techniques for Cloud Computing Environment

a. *Power-aware load balancing algorithm* (*PALBA*).

J.M. Galloway et al. proposed PALBA in [6] for energy efficient load balancing. The PALBA constitutes three basic sections:

(i) *Balancing section.* This section keeps monitoring the utilization percentage of all the active nodes and take decisions regarding instantiation of new VM on the compute node. If utilization of all compute nodes are above 75%, then PALBA instantiates a new VM on the compute node with the lowest utilization number. Utilization percentage of 75% for all the compute node is treated as all the nodes are in operation, otherwise, new VM is booted on the compute node which is having highest utilization if it can accommodate the VM. The threshold of 75% utilization is kept because still 25% of the resources of the node are free and at least one more VM can be accommodated using three out of five available configurations of Amazon EC2 specification. [11].

(ii) *Upscale section.* If all the currently active nodes have utilization greater than 75 % then the upscale section of the algorithm is invoked. It is used to extend the capability by powering on the available additional compute nodes.

(iii) *Downscale section.* The downscale section is invoked to power down the idle compute nodes. If the compute node utilizes less than 25% of its resources, the shutdown command is sent to that node.

b. *Join-idle-queue algorithm* (JIQA).

Y. Lu et al. proposed Join-Idle-Queue LB algorithm in [7] for dynamically scalable web services with distributed dispatcher to decouple discovery of lightly loaded servers from job assignment. This algorithm considers homogeneous processors and assumes Poisson arrival of requests. It involves informing to dispatchers, using a data-structure *I-Queue*, about the idle processors for their idleness without interfering job arrivals. Informing to large number of dispatchers causes the increase in the large number of jobs on idle processors and runs the risk of allocating too many jobs on the same processor, henceforth, results into large queuing overhead. This algorithm has two subparts:

(i) Primary load balancing.

Primary load balancing gathers information of idle servers present in the *I-Queue* that avoid the communication overhead from probing server loads. On arriving a job to the dispatcher, the dispatcher looks at the *I-queue* and if it is non-empty then dispatcher remove the first idle processor from the *I-Queue* and direct the job to this idle processor and, otherwise, choose the processor randomly for assignment of job. Each of the idle processors joins only one *I-Queue* to reduce extra overhead in withdrawing it during the assignment of the job.

(ii) Secondary load balancing.

For assignment of an idle processor to *I-Queue*, this algorithm works on two algorithms: *JIQ-Random* and *SQ*(*d*). With *JIQ-Random*, the processor chooses an *I-Queue* uniformly at random, and with *JIQ-SQ*(*d*), an idle processor selects '*d*' *I-Queues* and joins the *I-Queue* with smallest length.

c. *Honey bee foraging behavior algorithm* (*HBFBA*).

Dhinesh Babu L.D. et al. [8] proposed a LB technique for cloud computing environment based on Honey Bee Behavior inspired phenomena.

After finding the workload and standard deviation, it is decided that whether load balancing is required or not. So there are basically two situations: (i) Finding whether the system is balanced, and (ii) Finding whether the whole system is saturated or not (i.e. overloaded or not). In case of the overloaded condition, load balancing is meaningless. This algorithm follows steps given below:

1. *Finding the state of VM group.*
   If the standard deviation of the VM load is lower or equal to the threshold condition set then the system is said to be as Balanced. An imbalance system is either overloaded or under-loaded.

2. *Finding the overloaded group.*
   If the current workload of VM group exceeds the maximum capacity of the group then the group is declared to be overloaded and hence load balancing is not possible for such group.

3. *VM grouping.*
   The VM grouping takes place based on three criterion: (i) Overloaded VMs, (ii) Under-loaded VMs, and (iii) Balanced VMs. Tasks assigned to the overloaded VMs are placed on low loaded VMs or under-loaded VMs.

   The migrating task is considered as a honey bee and low loaded VMs are considered as destination of honey bees. Load balanced VMs are not used in switching of tasks. After switching of tasks, VMs become balanced and those balanced VMs are included into the balanced VMs group. The process continues until all the VMs are not included in the balanced VMs group and then load balancing is declared as successful.

4. *Task Transfer.*
   During the load balancing, following information's are required: (i) Overloaded VMs, (ii) demand i.e. load required, (iii) low-load VMs, and (iv) supply or available load. Then tasks are migrated from overloaded VMs which are referred as Scout bee. New VMs are selected based on task priority and the migrated tasks are assigned to low-loaded VMs i.e. Forager bee. Forager bee becomes Scout bee for next tasks. The process continues until load balancing is successful.

5. *VM Selection of different prioritized task.*
   The priorities of tasks are categorized into three type i.e. high, mid, and low. In case of overloaded VMs, tasks executing over current VM is migrated to the under-loaded machine. On under-loaded VMs, high priority tasks are considered as this task is already submitted. This method ensures that the high priority tasks will find the VM which has number of high priority of tasks.

d. *Biased random sampling algorithm* (*BRSA*).

O.A. Ramesh et al. proposed in biased random sampling scheme in [9] to distribute the load in the dynamically changing network as efficiently as possible.

This algorithm works on the principle of the degree distribution of nodes as stochastic system where number of nodes and number of edges are fixed. The in-degree refers to free resources of the node. Therefore, when a job is

assigned to the node, it removes one of its edges to reduce its in-degree by one. On completion of the job, it will add an edge back to the node to increase the in-degree by one. The increment and decrement of in-degree of the node are performed through Biased Random Sampling where nodes in the network are chosen randomly with equal probability.

The sampling starts at some fixed node and moves to the neighboring node of the current node in each step. This algorithm is encoded into the network itself and does not require any monitoring mechanism.

e. *Ant colony and complex network theory* (*ACCNT*).

Z. Zhang et al. proposed the Ant Colony and Complex Network (ACCN) theory in [10] to realize load balancing in distributed systems which take into consideration the complex network with small-world and scale-free characteristics. ACCN has the following functional steps:

1. Underloaded Load Balancing Method
   An ant is sent out by underloaded node periodically to notify that node is underloaded and initiate the load balancing on whole open cloud computing federation. To keep the vitality of complex network, ants update pheromone on each node and finds the transition probability $P_{ij}(t)$ to choose a neighbor node as its next step.

   During the trip, ant remembers the node with maximum and minimum workload, and if the difference between maximum and minimum workload crosses the certain threshold 'T' or total number of move steps are greater than previously appointed number 'm', then both of the nodes are informed by the ant to balance the load between them.

2. Overloaded Load Balancing Method
   If the workload on the node crosses the threshold 'W' of the node itself then an ant is sent out of the node and rest of the process of load balancing is same as for underloaded nodes.

3. The Update of Pheromone
   On performing load balancing on nodes with maximum and minimum workload, the ant needs to backtrack the path which it traversed to update the pheromone value on the nodes which is maintained through pheromone table in each node to link its neighbor node.

4. The Evolution of the Complex Network
   For implementing this algorithm on a complex network, two characteristics of complex network j should be satisfied: (i) Small-world, and (ii) Scale-free. Therefore, the addition of new edge between nodes with maximum and minimum workload is considered if there is no such pre-existing edge. On the other hand, deletion of existing edges is considered if it is not in operation for a very long duration. Nodes with a large number of degree may expedite the process of load balancing as the ant can move more quickly towards the resources for the execution of a task.

IV. Performance Evaluation Metrics

a. *Throughput.* In a literal sense, throughput is the rate of work done. With reference to LB, throughput is used to calculate the number tasks which has finished its execution. For high performance of the system, throughput should be high enough.

b. *Resource Utilization.* It is used to find the degree of resource utilization. Resource utilization is the measure of optimum performance of the underlying system.

c. *Overhead associated.* Execution of tasks and application in cloud environment involves a lot of workload movement, inter-process communication, and inter-process communication through message passing

mechanism or shared memory technique. Such mechanism causes the lot of overhead to the system. For the efficient execution of LB techniques, the overhead of the system should be minimized.

d. *Response time.* It is the amount of time taken by a LB algorithm to respond to the system. For high performance, this parameter should be minimized.

e. *Migration time.* Jobs and tasks executing on one node need to be migrated on other nodes depending on the algorithm. It takes a certain amount of time to transfer them to other nodes, therefore, it should be minimized for better performance.

f. *Fault tolerance.* Server systems and network links are not fail-safe in nature. To perform uniform LB alternative approaches are adopted due to potential failure situations. LB techniques should be fault-tolerant to make the system robust.

g. *Scalability.* Cloud computing framework promise to deliver services of a scalable system as per the user's requirement. Therefore, the LB algorithm should be capable to perform across the finite number of hosts, theoretically.

h. *Performance.* The performance of the system is calculated in terms of other parameters like response time, migration time, resource utilization and throughput etc. It determines the efficiency of the underlying system with reference to LB technique deployed.

i. *Energy consumption.* This parameter calculates the energy consumed by the various components or resources deployed in the underlying system. High energy consumption causes overheating and creation of hotspots. An efficient LB technique helps to avoid creation of hotspots across the underlying system.

j. *Carbon emission.* Energy consumption by the resources of the underlying system and carbon emission go hand in hand. The more is the energy consumption, the higher is the carbon emission. Therefore, LB techniques focus upon energy efficient solutions too.

Table I: Comparison of Load Balancing Techniques

Techniques	PALBA	JIQA	HBFBA	BRSA	ACCNT
Environment	IaaS Cloud Architectures	Cloud Data Centers	Large-Scale Cloud Systems	Large-Scale Cloud Systems	Open Cloud Computing Federation
Performance	✓	✓	✓	✓	✓
Response	×	✓	×	×	×
Scalability	✓	×	✓	✓	×
Overhead	✓	✓	×	×	×
Throughput	✓	×	✓	✓	×
Resource Utilization	✓	×	×	×	×
Fault Tolerance	×	×	×	×	✓
Migration Time	×	×	×	×	×
Energy Consumption	✓	×	×	×	×
Carbon Emission	✓	×	×	×	×

V.CONCLUSION

This paper surveyed five selected load balancing techniques for cloud computing environment and presented those techniques in summarized way followed by comparative analysis among them in Table I. Each of the technique was developed for the specific predesignated objective only such as Join-Idle Queue algorithm has efficient performance and response but not scalable and offers more overhead.

REFERENCES

[1] J. K. Verma, and C. P. Katti, "Study of Cloud Computing and its Issues: A Review", Smart Computing Review, Vol. 4, no. 5 pp. 389-411, Oct. 2014.

[2] J. K. Verma, C. P. Katti, and P. C. Saxena, "MADLVF: An Energy Efficient Resource Utilization Approach for Cloud Computing", *I.J. Information Technology and Computer Science*, Vol. 6, no. 7, pp. 56-64, Jun. 2014.

[3] J. K. Verma, and C. P. Katti, "A Comparative Study into Energy Efficient Techniques for Cloud Computing," *IEEE Proc. 2nd International Conference on Computing for Sustainable Global Development* (*INDIACom*), New Delhi, Mar 11-13, 2015, pp. 2062 – 2067.
[4] P. Krueger and M. Livny, "The Diverse Objectives of Distributed Scheduling Policies," *Proc. 7th Int'l Conf. Distributed Computing Systems*, IEEE CS Press, Los Alamitos, Calif., Order No. 801 (microfiche only), 1987, pp. 242-249.
[5] B.A. Shirazi, A.R. Hurson and K.M. Kavi, "Scheduling and Load Balancing in Parallel and Distributed Systems," ISBN: 0-8186-6587-4, April 1995, Wiley-IEEE Computer Society Press.
[6] J.M. Galloway, K.L. Smith and S.S. Vrbsky, "Power Aware Load Balancing for Cloud Computing," *Proceedings of the World Congress on Engineering and Computer Science*, San Francisco, USA, Oct 19-21, 2011, vol. 1, pp. 19-21. 2011.
[7] Y. LU, Q. XIE, G. KLIOT, A. GELLER, J.R. LARUS, AND A. GREENBERG, "JOIN-IDLE-QUEUE: A NOVEL LOAD BALANCING ALGORITHM FOR DYNAMICALLY SCALABLE WEB SERVICES," *PERFORMANCE EVALUATION*, VOL. 68, NO. 11, PP. 1056–1071, NOV. 2011.
[8] Dhinesh Babu L.D. and P.V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing*, Vol. 13, no. 5, pp. 2292–2303, May 2013.
[9] O. A. Rahmeh, P. Johnson, and A. Taleb-Bendiab, "A dynamic biased random sampling scheme for scalable and reliable grid networks," *INFOCOMP Journal of Computer Science*, Vol. 7, no. 4, pp. 1-10, Dec 2008.
[10] Z. Zhang and X. Zhang, "A Load Balancing Mechanism based on Ant Colony and Complex Network Theory in Open Cloud Computing Federation," Proc. Of 2nd International Conference on Industrial Mechatronics and Automation (ICIMA), Wuhan, China, 30-31 May, 2010, pp. 240-243.
[11] Amazon Web Service, Amazon EC2 Instances, Accessed Jan 1, 2013. http://aws.amazon.com/ec2/ instance-types/