C. Stabiliity

Stability can be characterized in terms of the delays in the transfer of information between processors and the gains in the load balancing algorithm by obtaining faster performance by a specified amount of time.

D. Overload Rejection

If Load balancing is not possible additional overload, failure can cause total failure in load balancing.

H. Migration Time

It is the time to migrate the jobs or resources from one node to other. It should be minimized in order to enhance the performance of the system.

7. Distributed Load Balancing for the Clouds

In complex and large systems, there is a tremendous need for load balancing. For simplifying load balancing globally (e.g. in a cloud), one thing which can be done is, employing techniques would act at the components of the clouds in such a way that the load of the whole cloud is balanced.

In a distributed system, dynamic load balancing can be done in two different ways: distributed and non-distributed. In the

There's no other way to assume increased load other than adding new instances and distributing that load with software or hardware. Similarly, when the additional instances of that application are deprovisioned, the changes to the network configuration need to be reversed, but software and hardware load balance is easy to scale up or scale down.

Obviously a manual process would be time consuming and inefficient, effectively erasing the benefits gained by introducing a cloud based architecture in the first place. The below is an example of how dynamic load balancing can be implemented [20].

- Let's assume that cloud management console, or a custom developed application or cloud tool kit, triggers an event that indicates a new instance is required to maintain availability. How it determines capacity limitations may be based on VM (Virtual Machines) status via VMware APIs (VMware is a virtualization and cloud computing software provider for x86compatible computers.) or data received from the load balancer, or a combination both.
- A new instance is launched in the cloud environment for same application. This is accomplished via the cloud management console or cloud tool kit.
- The cloud management console or tool kit grabs the IP address of the newly launched instance and instructs the load balancer to add it to the configuration as new resources for same application. This is accomplished by the standards based API which presents the configuration and management control plane of the load balancer to external consumers as services.
- The load balancer adds the new application instance to the appropriate configuration and as soon as it has confirmation that the instance is available and responding to requests, begins to direct traffic to that new instance without disturbing existing instances.

This process should be easily reversed upon termination of an instance, load balancer should be able to release termination instance IP. Note that, there may be other infrastructure components that are involved in this process that must also be considered on launch and decommission, but for this discussion we're just looking at the load balancing piece as it's critical to the concept of auto-scaling.

9. Load Balancing Algorithms for Distributed Systems

Here we will discuss three types of load balancing algorithms which can be applied to a distributed system [21]: honeybee foraging algorithm, a biased random sampling on a random walk procedure and active clustering.

A. Honeybee Foraging Algorithm

Randles et al. [22] investigated a decentralized honeybee-based load balancing technique that is a nature inspired algorithm for self- organization. It achieves global load balancing through local server actions. Performance of the system is enhanced with increased system diversity but throughput is not increased with an increase in system size. It is best suited for the conditions where the diverse population of service types is required.

This algorithm is derived from the behaviour of honey bees for finding and reaping food. There is a class of bees called the forager bees which forage for food sources, upon finding one, they come back to the beehive to advertise this using a dance called waggle dance. The display of this dance, gives the idea of the quality or quantity of food and also its distance from the beehive. Scout bees then follow the foragers to the location of food and then began to reap it. They then return to the beehive and do a waggle dance, which gives an idea of how much food, is left and hence results in more exploitation or abandonment of the food source [23].

The considered honeybee based load balancing technique uses a collection of servers arranged into virtual servers, each serving a virtual service queue of requests. Each server processing a request from its queue calculates a profit or reward, which is analogous to the quality that the bees show in their waggle dance. One measure of this reward can be the amount of time that the CPU spends on the processing of a request. The dance floor in case of honey bees is analogous to an advert board here. This board is also used to advertise the profit of the entire colony.

In load balancing operation [22], each server takes a particular bee role with probabilities px or pr. These values are used to mimic the honeybee colony whereby a certain number of bees are retained as foragers (to explore (px); rather than as harvesters) to exploit existing sources. A server successfully fulfilling a request will post on the advert board with probability pr. A server may randomly choose a virtual server's queue with probability px (exploring), otherwise checking for an advert (watching a waggle dance). In summary, idle servers (waiting bees) follow one of two behaviour patterns: a server that reads the advert board will follow the chosen advert, and then serve the request; thus mimicking harvest behaviour. A server not reading the advert board reverts to forage behaviour; servicing a random virtual server's queue request. An executing server will complete the request and calculate the profitability of the just-serviced virtual server.

B. Biased Random Sampling

Randles et al. [22] investigated a distributed and scalable load balancing approach that uses random sampling of the system domain to achieve self-organization thus balancing the load across all nodes of the system. The performance of the system is improved with high and similar population of resources thus resulting in an increased throughput by effectively utilizing the increased system resources. It is degraded with an increase in population diversity.

In this second load balancing approach, the load on a server is represented by its connectivity in a virtual graph. A full analysis of this mechanism is found in [24], with this section providing a brief overview. Here a virtual graph is constructed, with the connectivity of each node (a server is treated as a node) representing the load on the server. Each node is represented as a vertex in a directed graph and each in-degree represents free resources of that node.

Whenever a client sends a request to the load balancer, the load balancer allocates the job to the node which has at least one in-degree. Once a job is allocated to the node, the in-degree of that node is decremented by one. After the job is completed, the node creates an incoming edge and increments the in-degree by one. The addition and deletion of processes is done by the process of random sampling.

Each process is characterized by a parameter know as threshold value, which indicates the maximum walk length. A walk is defined as the traversal from one node to another until the destination is found. At each step on the walk, the neighbour node of current node is selected as the next node.

In this algorithm, upon receiving the request by the load balancer, it would select a node randomly and compares the current walk length with the threshold value. If the current walk length is equal to or greater than the threshold value, the job is executed at that node. Else, the walk length of the job is incremented and another neighbour node is selected randomly. The performance is degraded as the number of servers increase due to additional overhead for computing the walk length.

C. Active Clustering

Randles et al. [22] investigated a self-aggregation load balancing technique that is a self-aggregation algorithm to optimize job assignments by connecting similar services using local re-wiring. The performance of the system is enhanced with high resources thereby increasing the throughput by using these resources effectively. It is degraded with an increase in system diversity. Active clustering works on the principle of grouping similar nodes together and working on these groups. The process involved is:

- A node initiates the process and selects another node called the matchmaker node from its neighbours satisfying the criteria that it should be of a different type than the former one.
- The so called matchmaker node then forms a connection between a neighbour of it which is of the same type as the initial node.
- The matchmaker node then detaches the connection between itself and the initial node.

The above set of processes is followed iteratively.

10. Additional Algorithms

Here we describe some other load balancing algorithms used in distributed systems.

A. Compare and Balance

This algorithm [25] uses the concept of compare and balance to reach an equilibrium condition and manage unbalanced system's load on the basis of probability (number of virtual machine running on the current host and whole cloud system). The current node selects randomly a node and compares the load with itself.

B. ACCLB (Load Balancing mechanism based on Ant Colony and Complex network theory)

Zhang and Zhang [26] proposed a load balancing mechanism based on ant colony and complex network theory in an open cloud computing federation. It uses small-world and scale-free characteristics of a complex network to achieve better load balancing. This technique overcomes heterogeneity, is adaptive to dynamic environments, is excellent in fault tolerance and has good scalability hence helps in improving the performance of the system.

C. Join--Idle Queue

Join-idle queue [27] is basically used for large-scale systems. It uses distributed dispatchers by first load balancing the idle processors across dispatchers and then assigning jobs to processors to reduce average queue length at each processor. The disadvantage of this algorithm is that it is not scalable. Lua et al. [28] proposed this load balancing algorithm for dynamically scalable web services. It effectively reduces the system load, incurs no communication overhead at job arrivals and does not increase actual response time. It can perform close to optimal when used for web services. However, it cannot be used for today's dynamic-content web services due to the scalability and reliability.

11. Comparison of Existing Load Balancing Techniques Based on Various Metrics

Based on metrics discussed in section VI, the existing load balancing techniques have been compared in following table:

Metrics	*Honeybee Foraging*	*Biased Random Sampling*	*Active Clustering*	*Compare and Balance*	*ACCLB*	*Join-Idle Queue*
Performance	Yes	Yes	Yes	No	Yes	Yes
Response Time	No	No	No	No	No	Yes
Scalability	Yes	Yes	Yes	No	Yes	No
Overload	No	No	No	Yes	No	Yes
Throughput	Yes	Yes	Yes	No	No	No
Resource Utilization	No	No	No	Yes	Yes	No
Fault Tolerance	No	No	No	No	Yes	No
Migration Time	No	No	No	Yes	No	No

12. Conclusion

Load balancing is one of the main challenges in cloud computing. It is required to distribute the dynamic local workload evenly across all the nodes to achieve a high user satisfaction and resource utilization ratio by making sure that every computing resource is distributed efficiently and fairly. With proper load balancing, resource consumption can be kept to a minimum which will further reduce energy consumption and carbon emission rate which is a desire need of cloud computing.

This study explains the concept of load balancing, general idea about static and dynamic load balancing algorithms and explains various comparative parameters of different load balancing algorithms. This study also gives an overall description of various distributed load balancing algorithms that can be used in case of clouds. We have also compared these distributed load balancing algorithms and concluded that we can use a particular algorithm according to our requirement. But as we know that the cloud computing covers a very vast area, it is applicable to both small and large scale area but as we have concluded that none of the above algorithms satisfies the criteria. So there is a need to develop an adaptive algorithm which is suitable for heterogeneous environment and should also reduce the cost.

References

[1] Sidhu, A. K. and Kinger, S., "Analysis of Load Balancing Techniques in Cloud Computing", International Journal of Computers & Technology, 4, 2, March – April 2013.
[2] Kunamneni, V., "Dynamic Load Balancing for the Cloud", International Journal of Computer Science and Electrical Engineering (IJCSEE), 1, 1, 2012.
[3] Laha, J., Satpathy, R. and Dev, K., "Load Balancing Techniques : Major Challenges in Cloud Computing - A Systematic Review", International Journal of Computer Science and Network, 3, 1, 1-8, February, 2014.
[4] Kansal, N. J. and Chana, I., "Cloud Load Balancing Techniques: A Step Towards Green Computing", International Journal of Computer Science Issues (IJCSI), 9, 1, 238-246, January, 2012.
[5] Banerjee, S., Mandal, D., Adhikari, M. and Biswas, U., "Service Delivery Improvement for the Cloud Service Providers and Customers", International Journal of Computer Applications, 51, 5, 20-23, August, 2012.
[6] Kashyap, S. and Sharma, A. K., "Load Balancing Techniques in Cloud Computing Environment", International Conference on Electrical, Electronics & Computer Science Engineering, 94-97, 26th May, 2013.
[7] Chhabra, A., Singh, G., Waraich, S. S., Sidhu, B. and Kumar, G., "Qualitative Parametric Comparison of Load Balancing Algorithms in Parallel and Distributed Computing Environment", Proceedings of World Academy of Science, Engineering and Technology (PWASET), 16, November, 2006.
[8] Chopde, N. R. and Tijare, P. A., "Dynamic Load Balancing in Parallel Computing Based on Execution Time", International Journal of Advanced Research in Computer Science and Software Engineering, 2, 10, 372-375, October, 2012.
[9] Sharma, S., Singh, S. and Sharma, M., "Performance Analysis of Load Balancing Algorithms", World Academy of Science, Engineering and Technology, 38, 2008.
[10] Chaczko, Z., Mahadevan, V., Aslanzadeh, S. and Mcdermid, C., "Availability and Load Balancing in Cloud Computing", International Conference on Computer and Software Modeling, IACSIT Press, Singapore, 14, 134-140, 2011.
[11] Enslow P. H., "What is a "Distributed" Data Processing System ?", Computer, 11, 1, 13-21, January, 1978.
[12] Grosu, D. and Chronopoulos, A. T., "Noncooperative load balancing in distributed systems", Elsevier, Journal of Parallel and Distributed Computing, 65, 1022-1034, 2005.
[13] Nikravan, M. and Kashani, M. H., "A Genetic Algorithm for Process Scheduling in Distributed Operating Systems Considering Load Balancing", Proceedings 21st European Conference on Modelling and Simulation (ECMS), 2007.
[14] Madhu, K., M. and Shah, S., M., "Study on Dynamic Load Balancing in Distributed System", International Journal of Engineering Research & Technology (IJERT), 1, 9, November, 2012.
[15] Livny, M. and Melman, M., "Load balancing in homogeneous broadcast distributed systems", Proceedings of the Computer Network Performance Symposium, College Park, Maryland, United States, 47-55, 13th – 14th April, 1982,
[16] Rajguru, A. A. and Apte, S., S., "A Comparative Performance Analysis of Load Balancing Algorithms in Distributed System using Qualitative Parameters", International Journal of Recent Technology and Engineering (IJRTE), 1, 3, 175-179, August, 2012.
[17] Malik, S., "Dynamic Load Balancing in a Network of Workstation", 95.515 Research Report, 19th November, 2000.
[18] Wang Y. and Morris, R., "Load balancing in distributed systems", IEEE Trans. Computing, C-34, 3, 204-217, March, 1985.
[19] Alakeel, A. M., "A Guide to Dynamic Load Balancing in Distributed Computer Systems", IJCSNS International Journal of Computer Science and Network Security, 10, 6, 153-160, June, 2010.
[20] Kunamneni, V., "Dynamic Load Balancing for the Cloud", International Journal of Computer Science and Electrical Engineering (IJCSEE), 1, 1, 2012.
[21] Randles, M., Odat, E., Lamb, D., Abu-Rahmeh, O. and Taleb-Bendiab, A., "A Comparative Experiment in Distributed Load Balancing", Second International Conference on Developments in eSystems Engineering, 2009.
[22] Randles, M., Lamb, D. and Taleb-Bendiab, A., "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing", 24th International Conference on Advanced Information Networking and Applications Workshops, 551-556, 2010.

[23] Padhy, R. P., and Rao, P G. P., thesis entitled "Load balancing in cloud computing system", Department of Computer Science and Engineering, National Institute of Technology, Rourkela, Orissa, India, May, 2011.
[24] Foster, I., Yong, Z., Raicu, I. and Lu, S., "Cloud Computing and Grid Computing 360-Degree Compared", Grid Computing Environments Workshop, November, 2008.
[25] Zhao, Y. and Huang, W., "Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud" Fifth International Joint Conference on INC, IMS and IDC, 6-9, 2009.
[26] Zhang, Z. and Zhang, X., "A Load Balancing Mechanism Based on Ant Colony and Complex Network Theory in Open Cloud Computing Federation", Proceedings of 2nd International Conference on Industrial Mechatronics and Automation (ICIMA), 240-243, May, 2010.
[27] Sran, N. and Kaur, N., "Comparative Analysis of Existing Load balancing techniques in cloud computing", International Journal of Engineering Science Invention, 2, 1, 2013.
[28] Lua Y., Xiea Q., Kliotb G., Gellerb A., Larusb J. R. and Greenber A., "Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services", accepted in International Journal on Performance Evaluation, in press, 3rd August, 2011.

Authors Profile

Sheetanshu Rajoriya is the youngest author in the field of Computer Science and Applications. He was authored several books on this field. This name is synonym of mathematics as in his higher secondary examination, he scored cent percent marks in mathematics, for which he was awarded by many famous personalities. He then obtained his bachelor's degree in Computer Science from Dr. Hari Singh Gour Central University, Sagar, Madhya Pradesh in 2007 and attaining highest marks in the entire university. After that he completed MCA from the same university of repute in 2010 and again attaining highest marks in the entire university. He also obtained masters degree in mathematics. Now he is pursuing the PhD in Computer Applications from SunRise University, Alwar, Rajasthan. He has rich practical experience of 8 years in the field of Hardware & Networking and has been teaching to the students at various levels like diploma, PGDCA, BCA, MCA, etc.