Load Balancing in Cloud—A Systematic Review

Veenita Kunwar, Neha Agarwal, Ajay Rana and J.P. Pandey

Abstract Cloud computing is an upcoming technology, which has been recently introduced in the field of IT for delivering services that are hosted over the Internet. It is an amalgamation of Grid computing, Utility computing, Autonomic computing, and utilizes the concept of virtualization. It provides on demand service to the users for accessing resources, information, and software as per their needs. With increased popularity, there has been a tremendous increase in the demands of services by the users, which can be fulfilled by effective load balancing techniques. Load balancing allows even distribution of workload across various nodes in the cloud and aims to provide efficient utilization of resources, improving the system performance, minimizing the resource consumption resulting in low energy usage. In this paper, load balancing techniques proposed by researchers have been discussed and studied and a comparative analysis is being provided based on certain parameters.

Keywords Cloud computing · Load balancing · Virtualization

V. Kunwar (✉) · N. Agarwal · A. Rana
Amity University Uttar Pradesh, Noida, Uttar Pradesh, India
e-mail: veenitakunwar@gmail.com

N. Agarwal
e-mail: agarwalnehajain@gmail.com

A. Rana
e-mail: ajay_rana@amity.edu

J.P. Pandey
KNIT, Sultanpur, India
e-mail: tojppandey@rediffmail.com

1 Introduction

In the modern era, the Internet technology is developing at a faster rate, which has led to increase in the number of user requests for various services, which needs to be fulfilled in minimum possible time. For this, faster processing of servers is required in order to respond to various client requests. Thus, cloud computing comes into the picture.

Cloud computing is an evolutionary outgrowth of prevailing technologies that provides hosting and storage services on the Internet. It is an on demand, virtualized, location independent, pay peruse pricing model, which aims to achieve optimal resource utilization and higher throughput. But there are also certain issues involved like security, privacy, load balancing, fault tolerance, server consolidation. This paper addresses the load balancing issues.

Load Balancing is one of the prime challenges in the cloud, which distributes the tasks among multiple nodes evenly to provide proper resource utilization improving the overall system performance. It also provides low energy usage and less rate of carbon emission, which helps to achieve green computing.

In this paper, we have discussed and compared various load balancing algorithms developed in the cloud. The rest of the paper is organized as follows: Sect. 2 gives the overview of the cloud. Section 3 describes load balancing and its types. In Sect. 4, we discuss various load balancing algorithms with pros and cons. Finally, Sect. 5 concludes the paper showing areas of improvement in load balancing algorithms for the future scope.

2 Cloud Computing Overview

As defined by NIST [1] *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g.,* networks, servers, *storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.* It provides various benefits like low hardware and maintenance cost, accessibility, flexibility, scalability, high reliability, multi-tenancy, quick response, faster deployment, location independence. It uses virtualization concepts, which are the key technology used in the cloud that hides the details of physical machines and provides virtualized resources to various applications. It can of two kinds, namely full and paravirtualization. In full virtualization, the whole system is installed on another system while in para, multiple operating systems execute on a single system providing only partial services.

Cloud has four deployment models named as private, public, hybrid, and community. Private cloud is used by a single organization and offers highest degree control over performance, security, maintenance, reliability, deployment, and use. They are more secure and expensive than public clouds. Public clouds on the other hand are hosted by cloud vendors and can be used by anyone but lack control over data, network, and security settings. Users need to pay depending on the service used. Its benefits are low cost, on demand scalability, flexibility, location independence, efficiency in shared resources. Hybrid cloud is a combination of public and private cloud offering improved flexibility, scalability, and security. Community cloud is shared among several organizations and managed internally or by third-party service providers and are secure and cost effective.

Cloud offers three service models named as IaaS, PaaS, and SaaS. In Infrastructure as a Service, the resources like servers, network, storage, virtual machines, data centers, load balancers are made available by cloud providers, which can be accessed by applications and operating systems. Examples include Amazon EC2, GoGrid. Platform as a service provides a platform including operating system, software development framework, database, and Web server, which makes the development, testing, deployment, and installation of applications in a quick and cost effective manner. Examples include Google App Engine, Microsoft Azure. Software as a service delivers various applications over the Internet, which are managed by a third party vendor. The users can get rid off installing and running applications on individual systems. Google Apps is an example.

3 Load Balancing in Cloud Computing

Load Balancing is a mechanism that plays a vital role in the cloud by distributing workload from overloaded nodes to under loaded nodes in an efficient manner to achieve optimal utilization of resources. The load can be of any kind like load on CPU, memory usage, delay, or load on the network. It aims to achieve maximum resource utilization, higher throughput, minimum response time, and increased user satisfaction. Its objective is to minimize energy consumption, enhance system performance, accommodate future modifications, build a fault-tolerant system, and maintain the stability of the system.

Load Balancing can be broadly classified as static and dynamic. Static algorithms do not take into account the current state of the system and aforementioned knowledge about system applications and resources is required to assign tasks at compile time and to process new requests. They are easy to implement and well suited for homogeneous environments. Dynamic algorithms are based on the current state of the system. The tasks are assigned to processors at runtime. They are complex to implement, but provide better fault tolerance and performance. They provide efficient load balancing, but may face runtime overheads and communication delays. Dynamic algorithms can be distributed or non-distributed. The distributed algorithm involves all the nodes of the system while in non-distributed one or some of the nodes perform load balancing. Further the distributed dynamic algorithms can be classified as cooperative and non-cooperative. In cooperative, the nodes try to achieve a common objective while in non-cooperative nodes work independently toward individual goals. The non-distributed dynamic algorithms can be classified further as semi-distributed and centralized. In semi-distributed, system nodes are divided into clusters performing load balancing of centralized kinds in each cluster. In centralized, the central node performs the balancing of workload.

3.1 Load Balancing Measurement Parameters

1. *Throughput*—It is used to estimate the number of tasks successfully completed in a given amount of time.
2. *Overhead Associated*—It tells the number of overheads associated while implementing an algorithm.
3. *Fault Tolerance*—It is the ability of an algorithm to perform well even after failure.
4. *Performance*—It checks the overall efficiency of the system.
5. *Scalability*—The algorithm should perform well with the increase in the number of nodes as per needs.
6. *Response Time*—It is the time interval between request sent and the response received.
7. *Resource Utilization*—It keeps the track of utilization of resources.
8. *Migration Time*—It is the amount of time taken for migrating tasks from one node to another.

4 Literature Review

4.1 Decentralized Content-Aware Load Balancing

A policy was given by Mehta et al. [2] known as workload and client aware policy (WCAP), which has a unique and special property (USP) that is used to define the property of the service provider's nodes as well as the user's requests for information. It enables the scheduler to decide an apt node that can process these requests. Its implementation is being done in a decentralized manner with minimum overheads.

4.2 Server-Based Load Balancing for Internet Distributed Services

This solution was proposed by Nakai et al. [3] for balancing load that decreases the service response time by sending requests to the nearest server avoiding their overload. A middleware is defined that implements it.

4.3 Join-Idle-Queue

The technique was suggested by Lua et al. [4] for distributed load balancing in large systems. Initially, the load is balanced on idle processors across dispatchers and then, jobs are assigned to processors in order to minimize the length of the queue at each processor. It minimizes system load and response time is decreased.

4.4 A Lock-Free Multiprocessing Solution for LB

Liu et al. [5] proposed a technique that avoids shared memory usage, unlike other multiprocessing load balancing techniques that require shared memory and lock to manage sessions. The performance is boosted.

4.5 A Task Scheduling Algorithm Based on Load Balancing

It was suggested by Fang et al. [6]. A two-level task scheduling mechanism is provided in which tasks are mapped to VMs and VMs to host resources. It effectively improves the response time and system performance.

4.6 Scheduling Strategy on Load Balancing of Virtual Machine Resources

This strategy was described by Hu et al. [7], which is based on genetic algorithm. It considers previous data and present state of the system. It avoids dynamic migration. It attains optimal resource utilization.

4.7 Central Load Balancing Policy for Virtual Machines

The algorithm has been suggested by Bhadani and Chaudhary [8] and in this load is balanced evenly across virtual machines. It makes the system function well.

4.8 LBVS: Load Balancing Strategy for Virtual Storage

The strategy described by Liu et al. [9] makes an available model for data storage and storage as a service model. A three-layered architecture is used to obtain storage virtualization, and two load balancing modules are required for balancing the load on the system. It improves the efficiency of concurrent access, minimizes the response time, and boosts disaster recovery. It offers flexibility and robustness.

4.9 Two-Phase Load Balancing Algorithm (OLB + LBMM)

This algorithm was suggested by Wang et al. [10] that integrates OLB (opportunistic load balancing) and LBMM (Load Balance Min-Min) scheduling algorithms to have better execution. The tasks are stored in a queue, which are performed by the manager. OLB scheduling manager assigns the job to the service manager. LBMM algorithm chooses the apt service node, which will execute the subtasks. OLB keeps every node in working state to accomplish the goal of load balancing, and LBMM is utilized to curtail the runtime of each task on a node, which helps to minimize the overall completion time. This combined approach helps in obtaining proper and efficient resource utilization and boosts the working efficiency of the system.

4.10 Compare and Balance

This distributed load balancing algorithm was suggested by Zhao et al. [11], which is based on sampling to attain an equilibrium solution. A model has been implemented to decrease the VM migration time using shared storage and to achieve the zero-downtime relocation of VM by changing them as Red Hat cluster services. Implementation is being provided by adaptive live migration of VMs.

4.11 Honeybee Foraging Behavior

A decentralized honeybee solution was investigated by Randles et al. [12], which is based on bee's behavior for finding and reaping food. Scout bees forage for food sources and advertise this through waggle dance, which helps to know about the quality, quantity, and distance of food from the beehive. They are followed by forager bees to the food location to reap it. The tasks are considered as honeybees, which are removed from overloaded VMs and submitted to under loaded VMs. VMs act as food sources. The task which is removed updates the remaining tasks about the status of the VM and gives an idea about the assignment of tasks to other VMs based on the VM availability and load. It improves the overall throughput and reduces waiting time of the task.

4.12 Biased Random Sampling

A distributed and scalable approach was explored by Randles et al. [12], which uses random sampling to realize self-organization. The server load is shown by its connectivity with each node in a virtual graph. Each server node represents a node in the graph, in which each in-degree is mapped to the free resources of the server. When a node starts a new job, an incoming edge is removed, which indicates that the resources available are decreased. When the node finishes a job an inward edge is created, which indicates that the resources available are increased. The addition and deletion process are executed by random sampling. The walk starts from a particular node and moves to a randomly chosen neighbor. The final node in the walk is chosen for the allocation of the load. When a job is received by the node, it will get executed if the job's present walk length is more than or equal to the threshold of walk length. Otherwise, the walk length of job, which is under consideration, will be incremented and will be sent to a random neighbor. On job completion, an edge is generated from the initiating node allocation process to the executing job node. A directed graph is obtained at last.

4.13 ACCLB (Load Balancing Mechanism Based on Ant Colony and Complex Network Theory)

This procedure was suggested by Zhang et al. [13]. It takes the characteristic of complex network into account. It has excellent fault tolerance, good scalability, and enhances system performance.

4.14 Ant Colony Optimization

Nishant [14] proposed an algorithm, which is a modified version of *ACCLB* and makes use of ant's behavior to collect information about nodes to assign the task. He tries to resolve the issue of synchronization in ACCLB by the addition of "suicide" feature to the ants. When a request is made the ant's movement starts from the "head" node. A forward movement indicates the ant's movement from one overloaded node in search of the other node. If an under-loaded node is found by ants, it will keep on moving to check the next node. If next node turns out to be overloaded, then ant will return back to the prior node. Ant commits suicide if the target node is found.

4.15 MapReduce

MapReduce [15] takes two major tasks: mapping of tasks and result reduction. It involves three methods known as part, comp, and group. Initially, the part method is executed for mapping the tasks. The request entity is divided into parts using map tasks. The hash key table saves the key of each part and comp method compares the parts. The group method combines the parts of similar entities through the reduce tasks. Reduce tasks may get overloaded due to parallel reading and processing of entities by map tasks. One more level is added, which decreases the load on the tasks. The large tasks are divided into smaller tasks, which are sent to the Reduce tasks.

4.16 Dual Direction FTP

The technique proposed by Al-Jaroodi and Mohamed [16] is a dual direction algorithm from FTP servers. It splits m sized into m/2 parts. Each and every server node processes the assigned task that depends on certain patterns. For instance, a server starts from block 0 and downloads incrementally while some other server begins from block m and keeps downloading decrementally. Both these servers work independently and download the complete file to the client on best time given the properties as well as the performance of these servers. The task is considered to be complete when two servers download two conservative blocks and remaining tasks are assigned to servers. It minimizes communication needed between client and nodes and hence reduce network overhead. Load on node, network, and speed are taken into account. It does not require any runtime monitoring.

4.17 LBMM

This algorithm has three level framework of load balancing [17]. It makes use of OLB, which is static in nature and might cause slower processing of tasks. LBMM enhances OLB by providing three-layer architecture. The request manager receives the tasks and assigns it to the service manager, which divides the tasks into subtasks to boost the processing of that request. The subtask is assigned to the service node based on CPU space, memory.

Table 1 Existing load balancing algorithms

Algorithm	Observations
Decentralized content aware [2]	Searching performance is enhanced with minimum idle time
LB for internet distributed services [3]	Service response time is reduced
Join-Idle-Queue [4]	No Communication overhead but power consumption is more
Lock-free multiprocessing [5]	Performance is improved
Task scheduling based on LB [6]	Improved response time and proper resource utilization
Scheduling strategy on LB of VM resources [7]	The issue of load imbalance is resolved, but migration cost is high
Central LB policy for VMs [8]	Improvement in overall performance but no fault tolerance
LBVS [9]	Provides flexibility, robustness, and data storage
Two-phase scheduling (OLB + LBMM) [10]	Enhanced work efficiency with optimal resource utilization and better execution time. It is suitable for static environment
Compare and balance [11]	Load is balanced among servers, and equilibrium is attained faster
Honeybee foraging behavior [12]	Performs well under heterogeneous resources
Biased random sampling [12]	Performance is better but does not suit a dynamic environment
ACCLB [13]	Suitable for dynamic environments and provides excellent fault tolerance, scalability
Ant colony optimization [14]	It is decentralized. Network overhead occurs. Provides fault tolerance
MapReduce [15]	Less number of overheads with high processing time. It has high implementation complexity
DDFTP [16]	The calculation is faster. It provides reliable file download. Full replication of data files requiring high storage. No network overheads. Provides fault tolerance. Low implementation complexity
LBMM [17]	Load unbalance of Min-Min is improved and reduces the execution time. Node selection for complex tasks is not specified
ESCE	Enhances response time and processing time but no fault tolerance
Throttled	Current load on the node is not considered
Modified throttled [18]	Provides better response time. The index table state may change

4.18 Equally Spread Current Execution (ESCE)

In this, VMs are scanned. If an available VM can handle the request, then request is assigned to it. If a VM is overloaded then some of its tasks are distributed to VM having minimum load. It faces single point failure.

4.19 Throttled

In this, state of each VM is recorded. On arrival of the request, a table is searched. If a match is found, then the request is accepted else −1 is returned and the request lies in the queue. It increases the response time.

4.20 Modified Throttled

In this, a table is maintained containing a list of VMs and their states. The first VM is selected in the same way as in throttled. On arrival of a subsequent request, VM which is next to already assigned VM is selected and steps are followed. It provides better response time in comparison with throttled [18] (Table 1).

5 Conclusion and Future Work

Cloud computing is a very vast domain. It is used widely in present times, and therefore load balancing has become a huge challenge to overcome. Load balancing is used to evenly distribute the workload among various nodes. Numerous techniques proposed by the researchers have been discussed in this paper, and a comparative analysis has been done. Each technique differs from the other and covers some of the parameters. There is a need to develop new techniques, which can satisfy all the parameters. In future, we will try to create new algorithms, which will maintain trade offs among various different performance parameters. Also, there is a requirement of energy efficient techniques that provide maximum resource utilization and reduce energy consumption which will contribute toward green computing.

References

1. Mell, P., Grance, T.: The NIST definition of cloud computing. National Institute of Standards and Technology, Computer Security Resource Center. www.csrc.nist.gov
2. Mehta, H., Kanungo, P., Chandwani, M.: Decentralized content aware load balancing algorithm for distributed computing environments. In: Proceedings of the International Conference Workshop on Emerging Trends in Technology (ICWET), pp. 370–375 (2011)
3. Nakai, A.M., Madeira, E., Buzato, L.E.: Load balancing for internet distributed services using limited redirection rates. In: 5th IEEE Latin-American Symposium on Dependable Computing (LADC), pp. 156–165 (2011)
4. Lua, Y., Xiea, Q., Kliotb, G., Gellerb, A., Larusb, J.R., Greenber, A.: Join-idle-queue: a novel load balancing algorithm for dynamically scalable web services. Int. J. Perform. Eval. **68**, 1056–1071 (2011)
5. Liu, S., Pan, L., Wang, C.-J., Xie, J.-Y.: A lock-free solution for load balancing in multi-core environment. In: 3rd IEEE International Workshop on Intelligent Systems and Applications (ISA), pp. 1–4 (2011)
6. Fang, Y., Wang, F., Ge, J.: A task scheduling algorithm based on load balancing in cloud computing. In: Web Information Systems and Mining. LNCS, vol. 6318, pp. 271–277 (2010)
7. Hu, J., Gu, J., Sun, G., Zhao, T.: A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In: Third International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), pp. 89–96 (2010)
8. Bhadani, A., Chaudhary, S.: Performance evaluation of web servers using central load balancing policy over virtual machines on cloud. In: Proceedings of the Third Annual ACM Bangalore Conference (COMPUTE) (2010)
9. Liu, H., Liu, S., Meng, X., Yang, C., Zhang, Y.: LBVS: a load balancing strategy for virtual storage. In: International Conference on Service Sciences (ICSS), pp. 257–262. IEEE (2010)
10. Wang, S., Yan, K., Liao, W., Wang, S.: Towards a load balancing in a three-level cloud computing network. In: Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), Chengdu, China, pp. 108–113 (2010)
11. Zhao, Y., Huang, W.: Adaptive distributed load balancing algorithm based on live migration of virtual machines in cloud. In: Proceedings of 5th IEEE International Joint Conference on INC, IMS and IDC, Seoul, Republic of Korea, pp. 170–175(2009)
12. Randles, M., Lamb, D., Taleb-Bendiab, A.: A comparative study into distributed load balancing algorithms for cloud computing. In: Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, Perth, Australia, pp. 551–556 (2010)
13. Zhang, Z., Zhang, X.: A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation. In: Proceedings of 2nd International Conference on Industrial Mechatronics and Automation (ICIMA), Wuhan, China, pp. 240–243 (2010)
14. Nishant, K., Sharma, P., Krishna, V., Gupta, C., Singh, K.P., Nitin, N., Rastogi, R.: Load balancing of nodes in cloud using ant colony optimization. In: Proceedings 14th International Conference on Computer Modelling and Simulation (UKSim), pp. 3–8. IEEE (2012)
15. Kolb, L., Thor, A., Rahm, E.: Load balancing for MapReduce based entity resolution. In: Proceedings 28th International Conference on Data Engineering (ICDE), pp. 618–629. IEEE (2012)
16. Al-Jaroodi, J., Mohamed, N.: DDFTP: dual-direction FTP. In: Proceedings 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 504–503. IEEE (2011)
17. Wang, S.-C., Yan, K.-Q., Liao, W.-P., Wang, S.-S.: Towards a load balancing in a three-level cloud computing network. In: Proceedings 3rd International Conference on Computer Science and Information Technology (ICCSIT), vol. 1, pp. 108–113. IEEE (2010)
18. Domanal, S.G., Ram Mohana Reddy, G.: Load balancing in cloud computing using modified throttled algorithm. In: IEEE, International conference on CCEM (2013)