Systematic Review of Modern Load Balancing Algorithm in Cloud Environment

Mukesh Kashinath Beniwal[1] Dr. Santosh S. Lomte[2]
[1]Ph. D. Student [2]Principal
[1,2]Department of Computer Science
[1]Dr. Babasaheb Ambedkar Marathwada University, Aurangabad, India [2]VDF College of Engineering & Technology, Latur, India

Abstract— Cloud computing is the quickest growing computing era in current years. The cloud storage service gets the data from customer and store the cloud data hub. Load balancing is emerging research issue in cloud computing. This strategy is required to disperse the dynamic workload over different center points to ensure that no single center is over loaded. Load balancing techniques offer helps in optimal utilization of resources and consequently upgrading the performance of the system. The trustworthiness of clouds depends on the manner it handles the loads, to conquer such difficulties clouds must be featured with the load balancing techniques. Improvement in the resource usages the primary goal of load balancing. This paper provides discussion about the modern load balancing techniques and their comparison on various parameters.

Key words: Modern Load Balancing Algorithm in Cloud Environment, Cloud Environment

I. INTRODUCTION

Load balancing is one of significant issues in cloud processing [1]. This technique takes care that dynamic workload will be equally distributed to all nodes of clouds so that situation can be avoided where number of nodes remains idle and others are get heavily loaded. High resource optimization ratio can be archived by load balancing technique. Thus, this will advance the general execution and resource utility of the system. It as well ensures all computing resource is circulated fairly and efficiently. It in addition prevents bottlenecks of the device which may additionally occur due to load imbalance. Load balancing enables in continuation of the services when any components of any service fail, by using enforcing honest-over, i.e. in allocation and de-allocation of programs without any failure. It ensures that useful resource is distributed efficiently. [1] [3].

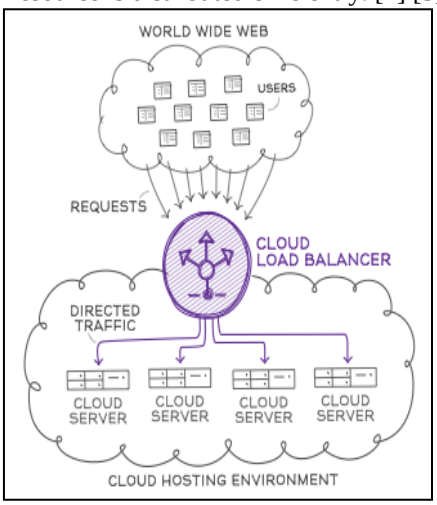

Fig. 1: An example of Cloud Load Balancing

II. NEED OF LOAD BALANCING IN CLOUD COMPUTING?

Load balancing is a mechanism that shifts the excess dynamic workload equally throughout all the nodes. It issued to achieve a high resource usage ratio [10], ensuring that no single node is beaten, consequently improving the overall performance of the system. Right load balancing can assist in making use of the available resources optimally, thereby minimizing there source intake. It also helps in imposing fail-over, permitting scalability, warding off bottlenecks and over provisioning lowering response time etc.

III. LOAD BALANCING: IT'S GOAL

The objectives of load adjusting are:

- To enhance the execution generously
- To have a reinforcement plan in case of system failure even in part
- To keep up the system strength
- To suit future alteration in the system.

IV. CATEGORY OF LOAD BALANCING ALGORITHM

Based totally on process orientation they're categorized as:

- Sender Initiated: On this sender initiates the method; the consumer sends request until a receiver is assigned to him to acquire his workload
- Receiver initiated: The receiver sends a request to well-known sender who is prepared to proportion the workload.
- Symmetric: It is grouping of each sender initiated and receiver initiated form of load balancing algorithm.

A. Static Load Balancing Technique

This algorithm needs knowledge of applications and system resources. Decision about moving the load does not depend on the present system state. Master processor assigns workload to slave processors by relying upon their execution performance. Slave processor performers the assigned job and the result is returned back to the master processor. Virtual machines performance is determined on the arrival of the work

As static load balancing algorithms are non-preemptive so no less than one task assigned to each machine. Its goals are to minimize task execution times and limit communication costs and delays. This algorithm has an inconvenience that the task is assigned to processors or machines only after it is created and that the task cannot be moved during execution to any different machine for balancing the load.

The four different types of static load balancing techniques

- Round Robin Algorithm
- The Central Manager Algorithm
- The Random Algorithm
- The Threshold Algorithm

1) Round Robin Algorithm

In this algorithm jobs get uniformly distributed by master node to each of the slave processors. All jobs are assigned to a list of available slave processors primarily based at the rotation order, which means that that the selection of a processor is finished in a series and could return to the primary processor if the closing processor is reached [5] [3]. Important benefit is that it does not cause overhead in the system because it does not need inter-process communication. Alternatively, it does not consider running time and that's why it does not achieve good performance [5]. It is suitable for web server application where HTTP requests are of a similar nature [5].

2) Randomized Algorithm

Randomized algorithm utilizes irregular to pick slave processors. Slave processors are selected randomly following irregular numbers produced depending on a statistic distribution. For Particular special applications Randomized algorithm gives the best result as compared to all available load balancing algorithms[5][12]. This algorithm works nicely if the processes are of equivalent load. It does no maintain a deterministic approach and presents the satisfactory overall performance when the Round Robin algorithm produces overhead for the process queue.

3) Central Manager Algorithm

In this algorithm Central Manager uses a central processor as a coordinator for distribution of workload between slave processors. The rule below to choose the slave processor is assigning the job to the processor that has the lowest load. The central pro gathers all slave processors to load the information and then uses it for making the load balancing decision so we expected better performance when applying this algorithm. The drawback of this type is the very high inter-process communication will create due to exchange of the load information which will result in lower performance of the system that could make bottleneck at central processor[5][12].

4) Threshold Algorithm

In Threshold calculation [3], when a new process is produced it will be allocated straightforwardly to the machine where it is made. For choice of first processor for any process no remote messages are exchanged and the choice is taken locally. Every processor maintains a duplicate of the system's present load. There are three levels to depict the load of a processor. Under loaded, normally loaded and overloaded. tunder and tupper are the two threshold factors can be used to explain these three levels:

- Under loaded = load < tunder
- Medium = tunder $\leq$ load $\leq$ tupper
- Overloaded = load > tupper

First, all processors are under load. When a processor load exceeds the upper limit of the load, it sends new load status messages to all other processors, and then regularly updates the actual status of the system. When the process is created and if its local state is not overloaded then process is allocated locally. However, if the node is overloaded, the process is assigned to the remote node under load, but if the remote node is not found, the process is allocated locally. This algorithm has very low communication between processes and a large number of local process assignments that, decreases the performance of system in general.

B. Dynamic Load Balancing Algorithm

This algorithm uses system's present state for managing balance of overhead. Decision about moving the load is taken by considering the system's present state. Processes are permitted to shift from a machine which is over-utilized to a machine which is under-utilized dynamically in no real time.

This algorithm searches the complete network and later selects lightest server for load balancing. For this real time communication with network is needed which can increase the traffic in the system.

1) Policies in dynamic load balancing

Following policies are used in dynamic load balancing.

- Transfer Policy: This policy chooses a job for transferring from local processor to remote processor.
- Selection Policy: It specifies the processors involved in exchange of load.
- Location Policy: It selects the target processor for the transferred task to be executed.
- Information Policy: This policy collects information about the nodes in the system.
- Load Estimation Policy: This policy determines the correct workload of a specific node.
- Process Transfer Policy: This policy decides whether process executes locally or process executes remotely.
- Priority assignment Policy: This policy determines the priority of local and remote process for a particular node.
- Migration Limiting Policy: This policy determines the total amount of times need by a process to transfer from one processor to another processor.

Two types of dynamic load balancing: Central Queue Algorithm and Local Queue Algorithm

a) Central Algorithm

The Central Hills Algorithm [12] works on the principle of dynamic distribution environment. To store new requests and the activities the main manager host maintains a cyclical FIFO. When a new activity or request arrives at the queue manager puts it in the queue. Each time a task is requested the queue manager removes the first queue activity and sends it to the applicant. When there is not a single queue activity, the application will be stored as long as the new activity becomes available. Once again when the the task comes to the queue manager, the first request assigned the activity. When a processor detects that the state is under load, send a request for a new activity to the central load handler and respond promptly to the request in case of business listings is available in the process queue.

b) Local Queue Algorithm

The key feature of this algorithm is that it supports dynamic process migration [12]. Basically it supports static allocation to all new processes. When host is under loaded, it initiates migration of all new processes. The load Capacity is defined by user-defined parameters. The parameter defines how many minimum jobs are ready the load manager can participate. When a new process is created in the primary host, it will be assigned to hosts under load. Because the first element of parallel activity is sufficient for load mapping to all remote hosts, all other processes created in the primary host and other

hosts are then mapped locally. When a host is under load, randomly sends the request with the number of local read jobs to remote load manager. When a load handler receives this type of request, it will compare the local ready processes with the received number and if local processes are richer, some running processes will be transferred to the applicant. The requesting node will send a confirmation to the sender after receiving the processes.

C. Ant Colony Optimization Algorithm

This algorithm [16] was inspired by the actual ants' behavior to find the optimal solution. Ants have the ability to find food, when they move on the road, put the pheromone on the ground to detect from other ants and follow that path. When more ants select the same path they create a denser pheromone. It attracts more ants. A pheromone table is created for each ant based on the use of resources. Below the loaded and overload nodes found when the ants move forward. To update the pheromone table, the table uses a strategy when it finds that the overhead node moves backwards to report under the loaded node. Each ant has its own pheromone table and its consistency of updating the path after the completed trip. Consistency remain same while proceed from source node to destination node. This algorithm [17] is used to find the optimal allocation of resources for each activity in the dynamic cloud system. If the ant pass through the wrong path, the pheromone value will be low, on the other hand, if the ants pass through with good path, the pheromone value will be high. The best scenario is that the under loaded node is found at the start of the search.

D. Honey Bee Foraging Algorithm

This is the nature that inspired and the self-organizing algorithm [11] used to solve load balancing in a dynamic cloud environment. Honeybees are able to find foods and notify other bees for the food in the bee colony. Every discovered bee finds their food initially and then informs to other bees. Foraging bees used different dance movements for food position and direction. When bees have found more food, the most energetic dance represents the bee scout, then follows the foraging bees and gets food. This phenomenon applies to the overloaded and under load virtual server. When clients request the server is found overloaded, they redirect the request to another loaded virtual server

V. QUALITATIVE PARAMETERS

Important qualitative parameters for load balancing are as follows:

- Throughput: The total number of tasks that gets finished their execution is referred as throughput. To achieve better system performance, high throughputs are required.
- Overload associated: The amount of overload produced by performing the load balancing algorithm. Minimum overhead is expected for proper accomplishment of the algorithm.
- Error Tolerance: It is the capability of the algorithm to function correctly and even in fault conditions in any arbitrary system node.
- Migration time: Time spent while migrating or transferring a task from one machine to another machine in the system. This time should be minimal to get better system performance.
- Response Time: This is the lowest amount of respond time required by distributed system for executing a specific load balancing algorithm.
- Resources utilization: the degree of use of system resources. A good load balancing algorithm ensures maximum utilization of resources.
- Scalability: It checks the capacity of the system to finish the load balancing algorithm with a limited number of machines or processors.
- Performance: It is system's usefulness after performing load balancing. If all the parameters mentioned above are met optimally, system performance will improve

Parameters	Central Queue	Local Queue	Honey Bee	Ant Colony	Round Robin	Random	Central Manager	Threshold
Nature	Dynamic	Dynamic	Dynamic	Dynamic	Static	Static	Static	Static
Response Time	More	More	More	More	Less	Less	Less	Less
Resources Utilization	Less	More	More	More	Less	Less	Less	Less
Fault Tolerant	Yes	Yes	Yes	Yes	No	No	Yes	No
Waiting Time	Less	Less	Less	Less	More	More	More	More
Reliability	More	More	More	More	Less	Less	Less	Less
Throughput	High	High	High	High	Low	Low	Low	Low
Turnaround Time	More	More	More	More	Less	Less	Less	Less
Adaptability	More	More	More	More	Less	Less	Less	Less
Stability	Small	Small	Small	Small	Large	Large	Large	Large
Process Migration	No	Yes	Yes	Yes	No	No	No	No
Predictability	Less	Less	Less	Less	More	More	More	More
Cooperative	Yes	Yes	Yes	Yes	No	No	Yes	Yes
Processor Thrashing	Yes	Yes	Yes	Yes	No	No	No	No

Table 1: Comparative Analysis of Load Balancing Algorithms

VI. CONCLUSION

In this paper based on different qualitative parameters' observe and analysis we have compared different load balancing algorithms. The load balancing depends at the time the activity will allocate, i.e. throughout the compilation time or at execution time. From this evaluation, we finish that static load balancing algorithms are strong than dynamic. But dynamic algorithms are always better in comparison to static ones because of the equal above stated parameters. In future work, we need to implement all these algorithms and check for specific parameters to choose good load balancing algorithm.

According to analysis on different load balancing algorithms, we consider ant colony optimization algorithm as our future work. In this algorithm, main parameters minimize make span, response time and throughput. Our goal to improve performance of all standard parameters and include more parameters like capacity of memory, overhead in communication in heterogeneous cloud environment.

REFERENCES

[1] B. P. Rima, E. Choi, and I. Lumb, "A Taxonomy and Survey of Cloud Computing Systems", Proceedings of 5th IEEE International Joint Conference on INC, IMS and IDC, Seoul, Korea, August 2009, pages 44-5l.
[2] A. M. Alakeel, "A Guide to dynamic Load balancing in Distributed Computer Systems", International Journal of Computer Science and Network Security (IJCSNS), Vol. 10, No. 6, June 2010, pages 153-160.
[3] B. P. Rimal, E. Choi, and I. Lumb, "A Taxonomy, Survey, and Issues of Cloud Computing Ecosystems, Cloud Computing: Principles, Systems and Applications", Computer Communications and Networks, Chapter 2 , pages 21-46, DOl 10.1007/978-1-84996-241-42, Springer - V erlag London Limited, 2010.
[4] Soundarabai. P, Sahai .R , Venugopal .R, Patnaik .L " Comparative Study on Load Balancing Techniques in Distributed Systems", International Journal of Information Technology and Knowledge Management Volume 6, No. 1, pp. 53-60 , December 2012.
[5] Abhijit A. Rajguru, S.S. Apte "A Comparative Performance Analysis of Load Balancing Algorithms in Distributed System using Qualitative Parameters", International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-1 Issue-3, August 2012.
[6] Rajani .SH, Garg. N "A Clustered Approach for Load Balancing in Distributed Systems", SSRG International Journal of Mobile Computing & Application (SSRG-IJMCA) – volume 2 Issue 1, Jan to Feb 2015.
[7] Kanungo .P "Measuring Performance of Dynamic Load Balancing Algorithms in Distributed Computing Applications", International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 10, October 2013.
[8] Gupta. R, Ahmed. J "Dynamic Load Balancing By Scheduling In Computational Grid System", International Journal of Advanced Research in Computer Science and Software Engineering Volume 4, Issue 6 ISSN: 2277 128X, June 2014.
[9] Razzaque .M, Hong. ch "Dynamic Load Balancing in Distributed System: An Efficient Approach", Department of Computer Engineering, Kyung Hee University , 1, Seocheon, Giheung, Yongin, Gyeonggi, Korea, 449-701.
[10] Alakeel .A "A Guide to Dynamic Load Balancing in Distributed Computer System", IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.
[11] Ruhi Gupta, "Review on Existing Load Balancing Techniques of Cloud Computing" in International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 2, February 2014
[12] Deepika, Wadhwa. D Kumar.N "Performance Analysis of Load Balancing Algorithms in Distributed System", Advance in Electronic and Electric Engineering. ISSN 2231-1297, Volume 4, Number 1, 2014.
[13] Sharma.S, Singh.S, Sharma.M "Performance Analysis of Load Balancing Algorithms", International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol. 2, No:2, 2008.
[14] Naaz. S, Alam. A, Biswas.R "Load Balancing Algorithms for Peer to Peer and Client Server Distributed Environments", International Journal of Computer Applications (0975 – 888), Volume 47– No.8, June 2012
[15] The jovathi JNTUCEA .M, Anantapur, India "Dynamic Load Balancing Algorithms for Distributed Networks", International Journal of Computer Science And Technology, Vol. 4, Issue 1, Jan - March 2013.
[16] Kun Li, Gaochao Xu, Guangyu Zhao, Yushuang Dong, Dan Wang, "Cloud Task scheduling based on Load Balancing Ant Colony Optimization" in 2011 IEEE Sixth Annual ChinaGrid Conference,pp.3-9,2011.
[17] Kwang Mong Sim and Weng Hong Sun,"Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions", IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, Vol. 33, No. 5, September 2003.

