Analysis On The Best Practices Of Load Balancing And Performance Evaluation Of LBDA

Mrs. J. Samatha[1], K. Rajendra Sreevatsa[2], P. Sai Sujith Reddy [3]

[1]Assistant Professor, Department of Computer Science and Engineering, Matrusri Engineering College, Saidabad, Hyderabad, Telangana.

[2,3] B.E Scholars, Department of Computer Science and Engineering, Matrusri Engineering College, Saidabad, Hyderabad, Telangana

Abstract

When numbers of computers (node) are connected together and forming a cluster in the cloud, it may be possible that some node become overloaded because of the random request of services by the clients. Because of the unbalanced cluster the performance of cloud will get worst. This condition is raising a high demand of load balancers or effective load balancing techniques. The load balancer distributes data depending upon how busy each server or node is. The load balancing becomes an important point for performance and stability of the system. Therefore, it is needed an algorithm for enhancing the system performance by balancing workload among VMs. We are in the quest of finding an efficient algorithm and comparing its practical performance with other existing algorithms. Task scheduling algorithms are used to achieve the load balancing and QoS. We aim to prove the proposed Load Balancing Decision Algorithm (LBDA) to manage and balance the load between the virtual machines in a datacenter along with reducing the completion time (Makespan) and Response time compared to with Generalized priority, SJF and opportunistic based algorithms.

1. INTRODUCTION

In today world the requests from the users and the data is increasing exponentially, so it is necessary to balance these requests accordingly. There are many factors that are to be considered to make an efficient load balancing algorithm, some of them are make-span, response time, execution time etc and storing and accessing of the date is done by using cloud. In a current scenario of IT industry cloud computing has become an emerging technology. Cloud computing is more than a simple virtualization, even virtual computers are only the component of cloud computing. To understand cloud computing it is necessary to understand the technologies that are involved in cloud computing. While using cloud computing it tries to separate the application from the operating system and operating system from the hardware that runs everything.

Effective load balancing results in minimizing resource consumption, implementing fail-over, enabling scalability, avoiding bottlenecks and overprovisioning. One more condition can arise when load balancer need to balance the traffic i.e. when an application requests to be uploaded to the cloud. Load balancing is the major concern in the cloud computing environment. Cloud comprises of many hardware and software resources and managing these will play an important role in executing a client's request. The load balancer distributes data depending upon how busy each server or node is. In the absence of a load balancer, the client must wait while his process gets processed, which might be too tiring and demotivating for him. Various information like jobs waiting in queue, CPU processing rate, job arrival rate etc. are exchanged between the processors during the load balancing process.

2. LITERATURE SURVEY

Shortest Job First is a dynamic load balancing algorithm which handles the process with priority basis. It determines the priority by checking the size of the process. This algorithm distributes the load randomly by first checking the size of the process and then transferring the load to a Virtual Machine, which is lightly loaded. In that case that process size is lowest, this process will get first priority to execute whether we suppose lowest sized process executes in minimum time. The load balancer spreads the load on to different nodes known as spread spectrum technique. The mechanism of Shortest Job First Algorithm is, to schedule the process with the shortest time to completion first, thus providing high efficiency and low turnaround time. In terms of time spent in the current program (job) began to enter in to the system until the process is finished the system, need a short time. Shortest Job First (SJF) scheduling algorithm can be said to be optimal with an average waiting time is minimal, which helps to improve the system performance.

Round Robin is the most widely deployed load balancing algorithm which is Easy to implement and conceptualize, using this method, client requests are routed to available servers on a cyclical basis. Round robin server load balancing works best when servers have roughly identical computing capabilities and storage capacity

In **Generalized priority algorithm** the tasks are initially prioritized according to their size such that one having highest size has highest rank. The Virtual Machines are also ranked (prioritized) according to their MIPS value such that the one having highest MIPS has the highest rank. Thus, the key factor for prioritizing tasks is their size and for VM is their MIPS. This policy is performing better than FCFS and Round Robin scheduling.

Opportunistic Load Balancing Algorithm is static load balancing algorithm so it does not consider the current workload of the VM. It attempts to keep each node busy. This algorithm deals quickly with the unexecuted tasks in random order to the currently available node. Each task is assigned to the node randomly. It provides load balance schedule without good results. The task will process in slow in manner because it does not calculate the current execution time of the node.

In **Token Routing** The main objective of the algorithm is to minimize the system cost by moving the tokens around the system. But in a scalable cloud system agent cannot have the enough information of distributing the work load due to communication bottleneck. So, the workload distribution among the agents is not fixed. The drawback of the token routing algorithm can be removed with the help of heuristic approach of token-based load balancing. This algorithm provides the fast and efficient routing decision. In this algorithm agent does not need to have an idea of the complete knowledge of their global state and neighbour's working load.

Randomized algorithm is of type static in nature. In this algorithm a process can be handled by a particular node n with a probability p. The process allocation order is maintained for each processor independent of allocation from remote processor. This algorithm works well in case of processes are of equal loaded. However, problem arises when loads are of different computational complexities.

Central queuing algorithm works on the principal of dynamic distribution. Each new activity arriving at the queue manager is inserted into the queue. When request for an activity is received by the queue manager it removes the first activity from the queue and sends it to the requester. If no ready activity is present in the queue the request is buffered, until a new activity is available. But in case new activity comes to the queue while there are unanswered requests in the queue the first such request is removed from the queue and new activity is assigned to it. When a processor load falls under the threshold then the local load manager sends a request for the new activity to the central load manager. The central manager then answers the request if ready activity is found otherwise queues the request until new activity arrives.

Connection mechanism Load balancing algorithm can also be based on least connection mechanism which is a part of dynamic scheduling algorithm. It needs to count the number of connections for each server dynamically to estimate the load. The load balancer records the connection number of each server. The number of connection increases when a new connection is dispatched to it, and decreases the number when connection finishes or timeout happens.

VectorDot- A. Singh et al. proposed a novel load balancing algorithm called VectorDot. It handles the hierarchical complexity of the datacenter and multidimensionality of resource loads across servers, network switches, and storage in an agile data center that has integrated server and storage virtualization technologies.

CARTON- R. Stanojevic et al. proposed a mechanism CARTON for cloud control that unifies the use of LB and DRL. LB (Load Balancing) is used to equally distribute the jobs to different servers so that the associated costs can be minimized and DRL (Distributed Rate Limiting) is used to make sure that the resources are distributed in a way to keep a fair resource allocation.

Compare and Balance- Y. Zhao et al. addressed the problem of intra-cloud load balancing amongst physical hosts by adaptive live migration of virtual machines. A load balancing model is designed and implemented to reduce virtual machines' migration time by shared storage, to balance load amongst servers according to their processor or IO usage, etc. and to keep virtual machines' zero-downtime in the process.

Event-driven- V. Nae et al. presented an event driven load balancing algorithm for real-time Massively Multiplayer Online Games (MMOG). This algorithm after receiving capacity events as input, analysis its components in context of the resources and the global state of the game session, thereby generating the game session load balancing actions.

Scheduling strategy on LB of VM resources – J. Hu et al. proposed a scheduling strategy on load balancing of VM resources that uses historical data and current state of the system. This strategy achieves the best load balancing and reduced dynamic migration by using a genetic algorithm.

CLBVM- A. Bhadani et al. proposed a Central Load Balancing Policy for Virtual Machines (CLBVM)

that balances the load evenly in a distributed virtual machine/cloud computing environment.

LBVS- H. Liu et al. proposed a load balancing virtual storage strategy (LBVS) that provides a large scale net data storage model and Storage as a Service model based on Cloud Storage. Storage virtualization is achieved using an architecture that is three-layered and load balancing is achieved using two load balancing modules. It helps in improving the efficiency.

Task Scheduling based on LB- Y. Fang et al. discussed a two-level task scheduling mechanism based on load balancing to meet dynamic requirements of users and obtain high resource utilization. It achieves load balancing by first mapping tasks to virtual machines and then virtual machines to host resources thereby improving the task response time, resource utilization and overall performance of the cloud computing environment.

Honeybee Foraging Behavior- M. Randles et al. investigated a decentralized honeybeebased load balancing technique that is a natureinspired 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.

Biased Random Sampling- M. Randles et al. 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.

Active Clustering- M. Randles et al. 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 in-creasing the throughput by using these resources effectively. It is degraded with an increase in system diversity.

ACCLB- Z. Zhang et al. proposed a load balancing mechanism based on ant colony and complex network theory (ACCLB) 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 excel-lent in fault tolerance and has good scalability hence helps in improving the performance of the system.

(OLB + LBMM) - S.-C. Wang et al. proposed a two-phase scheduling algorithm that combines OLB (Opportunistic Load Balancing) and LBMM (Load Balance Min-Min) scheduling algorithms to utilize better executing efficiency and maintain the load balancing of the system. OLB scheduling algorithm, keeps every node in working state to achieve the goal of load balance and LBMM scheduling algorithm is utilized to minimize the execution time of each task on the node thereby minimizing the overall completion time.

Decentralized content aware- H. Mehta et al. proposed a new content aware load balancing policy

named as work-load and client aware policy (WCAP). It uses a parameter named as USP to specify the unique and special property of the requests as well as computing nodes. USP helps the scheduler to decide the best suitable node for processing the requests.

Join-Idle-Queue- Y. Lua et al. proposed a Join-Idle-Queue load balancing algorithm for dynamically scalable web services. This algorithm provides large-scale load balancing with distributed dispatchers by, first load balancing idle processors across dispatchers for the availability of idle processors at each dispatcher and then, assigning jobs to processors to reduce average queue length at each processor.

3. PROPOSED SYSTEM

Input: List of un mapping tasks and available resources.

Output: Complete tasks with less execution time.

1. Initially set the characteristics of VMs and create the tasks with different length.
2. Calculate VM capacity
    a. For each task in tasks list:
    b. Calculate VM load.
3. Check VM state:
    a. If $VM_{Load} <= C_{VM} * TLL \rightarrow$means the state of the VM is Under loaded state.
    b. If $VM_{Load} > C_{VM} * TLL$ && $VM_{Load} <= C_{VM} * TFL \rightarrow$VM means the state of the VM is Balance state.
    c. If $VM_{Load} > C_{VM} * TFL$ && $VM_{Load} <= C_{VM} * TUL \rightarrow$ means the state of the VM is High Balance.
    d. If VM Load > CVM ∗ TUL → means the state of the VM is Overloaded state.
    e. If there are VMs' Under loaded state, then calculates ECT and mapping the task to VM that has less estimated completion time, if there are more than one VM has same ECT then selects VM with the largest capacity.
    f. If all VMs' Balance then mapping task to VM has less estimated time and with largest capacity, if there.
4. If virtual machines are in the state (Balance and High Balance):
    a. Calculate the less estimated completion time of VMs' Balance state, if there are more than one VM has same ECT then selects VM with the largest capacity.
    b. Calculate the less estimated completion time of VMs' High Balance state with the delay∗.
    c. Finally, select the lowest of ECT between the two types VMs' state (Balance, High Balance)

∗ Delay: The delay means is the amount of time for completed tasks *process*

5. when VM in High Balance loaded state Update load of VM

6. Calculate Average Makespan, Mean of Average Response Time and Total Execution Time.

4. SYSTEM ARCHITECTURE

Scheduler: The user will send the Cloudlet_List, cloudlet independency list as input to the scheduler that enter the request to the balancing.

Balancing: It assigns the task to the suitable VM that is determined from that assigns tasks to VMs based on the information that is collected.

VM Check: Check the VM state by comparing the capacity of VM with the VM load using three types of thresholds: threshold upper, threshold fair, and threshold lower.

VM Decide: Selection of suitable VM will be decided and we take into account the benefit of user and provider, where increasing the resources utilization, load balancing and less completion time.

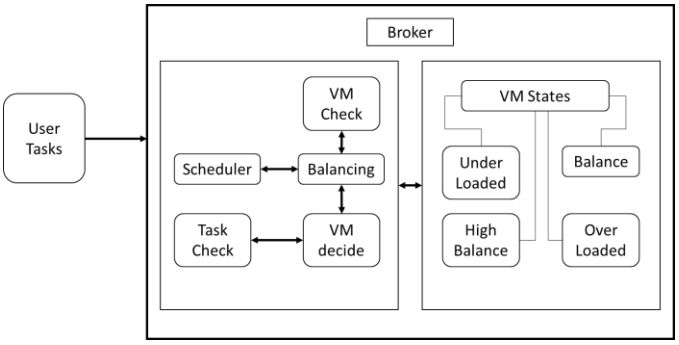

Figure 4.1. System architecture

5. PERFORMANCE EVALUATION

Then we have conducted evaluation by running the codes multiple times with different number of tasks and virtual machines

Evaluated the results of the performance of the LBDA, Generalized priority, SJF and opportunistic based algorithms by using the CloudSim toolkit. The metrics have tested the Average of makespan (completion time of the last task to be executed), the Mean of Average Response Time and Total of all Execution Time for all tasks in virtual machines. Five experiments 100, 150, 200,250 and 300 tasks executed in a 6, 8, 10, 12 and 14 number of VMs respectively. Below tables, shows the results are implemented. It is obvious that average makespan based approach LBDA produced better results than the others algorithms in all experiments.

Table 5.1. LBDA results

Experiments	Average makespan	Mean average response time	Total Execution time
100 tasks 6 vms	80.05	64.13	704.36
150 tasks 8 vms	88.13	68.06	1047.13
200 tasks 10 vms	97.56	70.04	1400.31
250 tasks 12 vms	102.68	71	1736.04
300 tasks 14 vms	109.63	73.13	2037.81
350 tasks 16 vms	116.87	73	2448.31
400 tasks 18 Vms	123.24	74.68	2810.04

Table 5.2. Average makespan of LBDA with other algorithms

Experiments	LBDA	GPA	OLB	SJF
100 tasks 6 vms	80.05	81.11	81.05	81.17
150 tasks 8 vms	88.13	89.16	89.13	89.32
200 tasks 10 vms	97.56	96.04	96.56	96.81
250 tasks 12 vms	102.68	103.59	103.68	103.99
300 tasks 14 vms	109.63	111.23	110.63	110.99
350 tasks 16 vms	116.87	117.13	117.47	117.89
400 tasks 18 Vms	123.24	124.52	124.24	124.72

Table 5.3. Mean average response time of LBDA with other algorithms

Experiments	LDBA	GPA	OLB	SJF
100 tasks 6 vms	64.13	65.11	65.17	65.16
150 tasks 8 vms	68.06	69.03	69.1	69.09
200 tasks 10 vms	70.04	71.38	71.41	71.45
250 tasks 12 vms	71	72.95	73.04	73.03
300 tasks 14 vms	73.13	74.08	74.15	74.16
350 tasks 16 vms	73	74.94	75.04	75.02
400 tasks 18 Vms	74.68	75.61	75.68	75.7

6. VISUALISATION

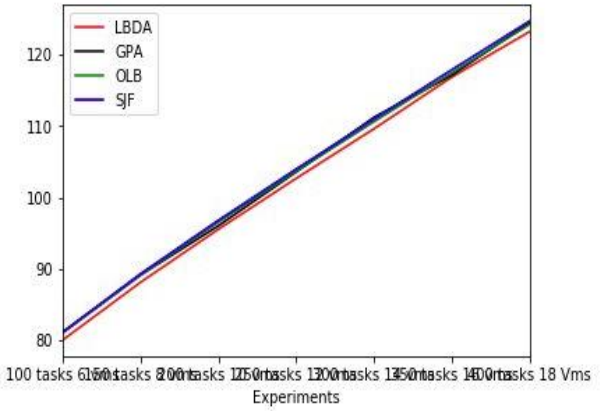

Figure 6.1. visualization of average makespan as line

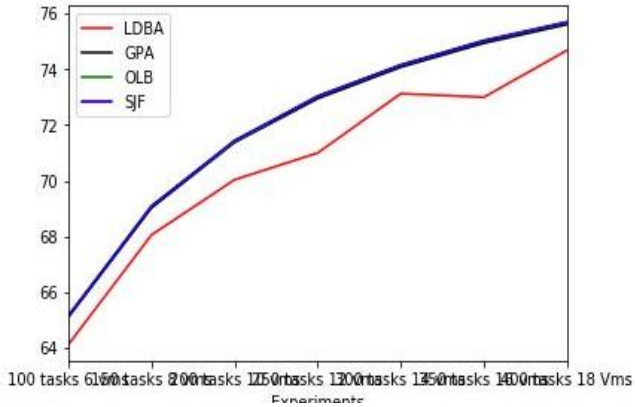

Figure 6.4. visualization of mean response times as line

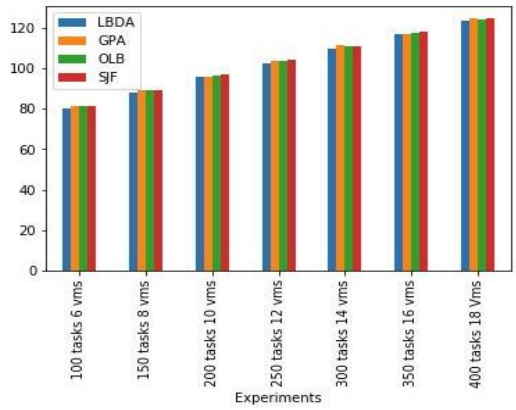

Figure 6.2. visualization of average makespan

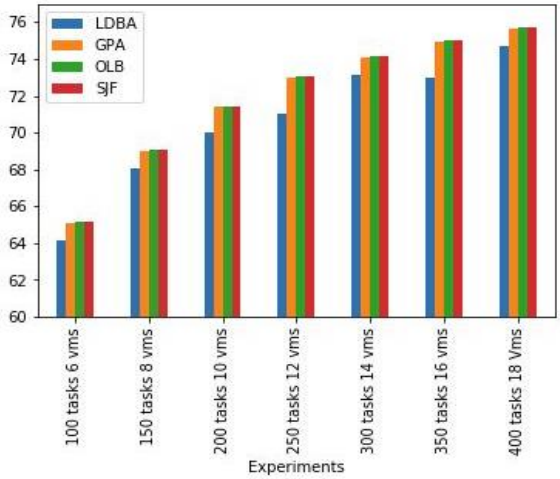

Figure 6.3. visualization of mean response

7. CONCLUSION

In cloud computing environment, there are many challenges, one of them is load balancing. It has an important impact on the performance. The maximize resources utilization and users' satisfaction can be improved by good load balancing. In this proposed algorithm, LBDA solving the tasks in load balancing problem by considering the VMs' capacity and the estimated completion time for each task to map the tasks to the most appropriate VMs. We conducted five experiments to evaluate the performance of proposed algorithm. The comparison of the proposed algorithm is done with Generalized priority, SJF and opportunistic based algorithms under same configuration environment. The simulation results show that the proposed algorithm has outperformed in all cases as compared to Generalized priority, SJF and opportunistic based algorithms by reducing

average makespan, Mean of Average Response Time and Total Execution Time in all VMs.

8. REFERENCES

[1] Chitra DD, Uthariaraj VR. Load balancing in cloud computing environment using Improved Weighted RoundRobin Algorithm for nonpreemptive dependent tasks. The Scientific World Journal. 2016; 2016.

[2] Solmaz A, Motamedi S, Sharifian S. Task scheduling using Modified PSO Algorithm in cloud computing environment.International Conference on Machine Learning, Electrical and Mechanical Engineering; 2014. p. 37–41.

[3] Imran MA, Pandey M, Rautaray SS. A proposal of resource allocation management for cloud computing. International Journal of Cloud Computing and Services Science. 2014; 3(2):79–86.

[4] Jamuna RMR, Gouda KC, Nirmala N. Load balancing technique for climate data analysis in cloud computing environment. International Journal of Science, Engineering and Computer Technology. 2013; 3(5):183–85.

[5] Namrata G, Garala K, Maheta P. Cloud load balancing based on ant colony optimization algorithm. IOSR Journal of Computer Engineering (IOSR-JCE); 2015. p. 11–18.

[6] Danilo A. Quality-of-service in cloud computing: modeling techniques and their applications. Journal of Internet Services and Applications. 2014; 5(1):1–17.

[7] Jia Z. A Heuristic clustering-based task deployment approach for load balancing using Bayes Theorem in cloud environment. IEEE Transactions on Parallel and Distributed Systems. 2016; 27(2):305–16.

[8] Crossref Kunjal G, Goswami N, Maheta ND. A performance analysis of load Balancing algorithms in Cloud environment. 2015 International Conference on Computer Communication and Informatics (ICCCI), IEEE; 2015. p. 4–9.

[9] Beghdad BK, Benhammadi F, Benaissa F. Balancing heuristic for independent task scheduling in cloud computing.2015 12th International Symposium on Programming and Systems (ISPS), IEEE; 2015.

[10] Aditi S, Sharma S. Credit based scheduling using deadline in cloud computing environment. International Conference on Resent Innovation in Science Engineering and Management; 2016. p. 208–16.

[11] R. P. Pandhy (107CS046) and P. G. P. Rao (107CS039), “Load balancing in cloud computing system” Department of computer science and engineering National Institute of Technology Rourkela, Rourkela769008, Orissa, India (2011) May.

[12] M. Randles, D. Lamb and A. Taleb-Bendiab, “A comparative study into distributed load balancing algorithms for cloud computing,” 2010 IEEE 24thinternational conference on advanced information networking and application workshops, (2010), pp. 551 - 556.

[13] J. Kaur “Comparison Load Balancing Algorithms In A Cloud”, International Journal Of Engineering Research And Applications, (2012), pp.1169 – 1173

[14] X. Nan, Y. He, and L. Guan, ―Towards Optimal Resource Allocation For Diffentiated Multimedia Services in Cloud Computing Environment‖, in theProc. IEEE International Conferenceon Acoustic, Speech and Signal Processing (ICASSP), pp. 1–5, 2014.

[15] B. R. Raman, R. Calheiros and R.N., ―Modeling and Simulation of Scalable Cloud Environment and the CloudSim Toolkit: Challenges and Opportunities’’, IEEE publication, pp. 1-11, 2009. October, 2014.

[16] Ren, X., R. Lin and H. Zou, "A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast" in proc. International Conference on. Cloud Computing and Intelligent Systems (CCIS), IEEE, pp: 220-224, September 2011.

[17] Lee, R. and B. Jeng, "Load-balancing tactics in cloud," in proc. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), IEEE, pp:447-454, October 2011.

[18] Al-Jaroodi, J. and N. Mohamed. "DDFTP: Dual-Direction FTP," in proc. 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), IEEE, pp:504-503, May 2011.

[19] Wang, S-C., K-Q. Yan, W-P. Liao and S-S. Wang, "Towards a load balancing in a three-level cloud computing network," in proc. 3rd International Conference on. Computer Science and Information Technology (ICCSIT), IEEE, Vol. 1, pp:108-113, July 2010.

[20] Sang, A., X. Wang, M. Madihian and RD. Gitlin, "Coordinated load balancing, handoff/cell-site selection, and scheduling in multi-cell packet data systems," in Wireless Networks, Vol. 14, No. 1, pp: 103- 120, January 2008.

ISSN NO: 1934-7197