ISSN:2320-0790

An International Journal of Advanced Computer Technology

COMPREHENSIVE STUDY ON LOAD BALANCING TECHNIQUES IN CLOUD

B S Rajeshwari[1], Dr. M Dakshayini[2]

[1]Asst. Professor, Dept. of CSE, BMSCE, Bangalore

[2]Professor, Dept. of ISE, BMSCE, Bangalore

Abstract: Due to advancement in technology and growth in human society, it is necessary to work in an environment that reduces cost, utilizes resources effectively, reduces man power and minimizes space utilization. This led to the development of Cloud Computing technology. Cloud computing is a kind of distributed computing with a collection of computing resources located in distributed data centers. It provides massively scalable IT related capabilities to multiple external customers on "pay per use" concept using internet technologies. The increase in the web traffic and different services day by day makes load balancing a critical research topic. Load balancing is one of the central issues in cloud computing. It is the process of distributing the load optimally and evenly among various servers. Proper load balancing in cloud improves the performance factors such as resource utilization, job response time, scalability, throughput, system stability and energy consumption. Many researchers have proposed various load balancing techniques. This paper presents description of various existing centralized and distributed load balancing techniques in cloud environment.

I. Introduction

Cloud Computing is a latest technology in which all computing resources like hardware, software and platforms for developing applications are provided as services to the customers through internet. Customers do not have to invest capital to purchase, manage, maintain and scale the physical infrastructure. The customers can take required resources on demand from the cloud providers and pay for it as they use.

The services that are provided by the cloud providers are broadly classified into three categories:

Infrastructure-as-a-Service (IaaS): In Infrastructure as a Service model, the service provider owns the equipments including storage, hardware, servers and networking components and is provided as services to the clients. The client typically pays on per-use basis. Amazon elastic Compute (EC2) and Simple Storage Service (S3) are typical examples for IaaS.

Platform-as-a-Service (PaaS): In Platform as a Service model, the service provider provides virtualized server, operating system and development tools as service. Using these services, users can develop, test, deploy and manage new applications in a cloud environment or run existing applications. These applications are delivered to users via the internet. Google App Engine is a typical example for PaaS.

Software-as-a-Service (SaaS): In Software as a Service model, the service provider provides software as a service over the Internet, eliminating the need to buy, install, maintain, update and run the application on the customer's own computers. Google Docs is a typical example for SaaS.

A cloud service has four distinct characteristics as follows:

- It is elastic: A user can dynamically scale up and scale down resources as they want at any given time.
- Pay per use: Usage is metered and user pays only for what they consume.
- Operation: The service is fully managed by the provider.
- Self-service: Users can add a new CPU, a server instance or extra storage using the console offered by the cloud provider.

II. Load Balancing

Load Balancing is another important aspect of cloud computing to balance the load among various servers. It is a mechanism that distributes the excess workload dynamically and evenly across all the servers. It is used to achieve high user satisfaction and resource utilization ratio and hence improving the overall performance of the system. Proper load balancing can help in utilizing the available resources optimally, thereby reducing response time, cost and energy consumption. The different types of Load balancing are as shown in Fig. 1.

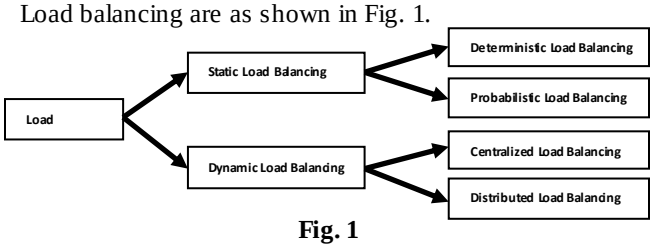

Fig. 1

Centralized Load balancing Techniques

In centralized load balancing approach, only one server acts as the central controller. It allocates jobs to the other servers. The centralized approach is a simple approach and has less communication cost. Various centralized load balancing techniques are as follows:

The **Minimum Execution Time** (MET) [1] [21] algorithm assigns each job to the server on which it has minimum execution time, regardless of the current load on that server. This algorithm tries to find a good job-server pair. Since it does not consider current workload of the server, it may cause load imbalance among the servers.

The **Minimum Completion Time** (MCT) [1] [2] [21] algorithm assigns each job to the server which has minimum expected completion time for the job. The completion time of the job is the sum of expected execution time of the job on that server and the server's ready time. This is much more successful heuristic as both execution time and the server loads are considered, but this MCT algorithm calculates the completion time only once when the job arrives.

In **Min-Min** (MM) scheduling algorithm [3] [21] the minimum completion time for every unscheduled job is calculated. Then, it selects the job with minimum completion time and this job is assigned to the server which offers minimum completion time. The ready time of all the servers is updated accordingly and this process is repeated for the next unscheduled job. In MCT algorithm, the completion time is calculated only once when the job arrives. In Min-Min scheduling algorithm, the completion time is recalculated every time a job is scheduled. This reduces the overall completion of all the jobs. The serious drawback of Min-Min algorithm is that the larger jobs may experience starvation.

In **Max-Min** scheduling algorithm [1] [2] the completion time for every unscheduled job is calculated. Then, it selects the job with maximum completion time. This job is assigned to the server which offers minimum completion time. The ready time of all the servers is updated accordingly and this process is repeated for the next unscheduled job. The Max-Min is based on the intuition that it is good to schedule larger jobs earlier.

The **Load Balance Min-Min** scheduling algorithm [3] adopts MM scheduling approach and load balancing strategy. In this scheduling algorithm, a job is divided into subtasks. The execution time of each subtask on each server and the threshold (Average execution time of each subtask on all servers) is calculated. Each subtask and the server that offers minimum execution time for that subtask is entered into Min time array. The subtask that has minimum execution time among all the subtasks in the Min time array is selected and assigned to the corresponding server, if execution time is less than or equal to the threshold. This subtask is deleted from the Min time array and the Min time array is rearranged. If the execution time of a subtask is greater than the threshold, the execution time is set to ∞ in the Min time array indicating execution time is too long to be considered now. The same process repeats for next unscheduled subtasks.

The **Load Balance Max-Min-Max** [4] scheduling algorithm calculates the average completion time of each task on all servers. The task that has maximum average completion time and the server that gives least completion time (time less than maximum average completion time) are selected. Now, the selected task will be executed by the corresponding server. If the server is already assigned, then the algorithm calculates completion time for both assigned and unassigned server. For assigned server, completion time is the sum of completion time of assigned task and completion time of current task. For unassigned server, completion time is completion time of the current task. This process is repeated for the next unassigned tasks.

Shu-Ching Wang et al., [3] proposed a two **phase scheduling algorithm OLB (Opportunistic Load Balancing) and LBMM (Load Balance Min-Min**) under three level cloud computing network. The first level is the request manager that assigns the task to the suitable service manager. The second level is the service manager that divides the task into subtasks. The third level is the service node that is used to execute a subtask. In the OLB scheduling algorithm, the request manager dispatches unexecuted tasks to currently available service manager at random order without considering the current workload of the service manager. The service manager divides the task into subtasks. The LBMM scheduling algorithm calculates execution time of each subtask on each service node and distributes subtasks to the service node that takes minimum execution time. The proposed two phase scheduling algorithms result in better execution efficiency and maintain good load balancing of a system.

Shu-Ching Wang et. al. [10] proposed a **three-phase scheduling (BTO+EOLB+EMM**) algorithm. In the proposed hierarchical network, fist level has a request manager, the second level has a set of service managers and the lowest level has a set of servers under each service manager. The servers that have the processing capability and memory greater than or equal to 0.8 are grouped under service manager 1. The servers that have the processing capability and memory greater than or equal to 0.6, but less than 0.8 are grouped under service manager 2. The servers that have the processing capability and memory greater than or equal to 0.4, but less than 0.6 are grouped under service manager 3. The servers that have the processing capability and memory greater than or equal to 0.2, but less than 0.4 are grouped under service manager 4. The servers that have processing capability and memory less than 0.2 are grouped under service manager 5. In the first phase, the Best Task Order (BTO) scheduling algorithm determines the execution order for each task. In the second phase, Enhanced Opportunistic Load Balancing (EOLB) algorithm assigns a task to the corresponding service

manager. The Service manager divides the task into subtasks. In the third phase, Enhanced Min-Min (EMM) algorithm assigns the subtask to a suitable service node. The experimental result shows that by combining Enhanced Opportunistic Load Balancing with Enhanced Min-Min algorithm enhances performance about 50% when compared with the combination Opportunistic Load Balancing with Min-Min algorithm and about 20% when compared with the combination Enhanced Opportunistic Load Balancing with Min-Min algorithm.

Wilhelm Kleiminger et. al. [22] presents a **Combined Stream Processing System.** Here, as the input stream rate varies, the workload is adaptively balanced between a dedicated local stream processor and a cloud stream processor. This approach utilizes cloud machines only when the local stream processor becomes overloaded. Load balancer dynamically distributes the load between the local processor and the cloud processor. In order to reduce bandwidth required, system outsource limited amount of data with techniques such as tuple filtering and compression.

Divya Thazhathethil et. al. [25] proposed a **Model for Load Balancing by Partitioning the Public Cloud**. The load balancing model includes main controller, balancers and servers. The cloud partition status can be idle, normal or overload. When a job arrives, the main controller communicates with the balancers in each partition and collects the status information. If status is idle or normal, the job is handled locally, otherwise another cloud partition which is not overloaded is found and the job is transferred to that balancer. The balancer further checks the load of each server under a partition and job is assigned to the server with minimum load. This system helps in dynamically allocating a job to the least loaded server, thus increasing the performance, resource utilization and availability of resources.

P. Jamuna et. al. [26] proposed O**ptimized Load Balancing by Cloud Partitioning Technique** that helps the service providers to simplify the load balancing process. The proposed model divides the cloud consisting of numerous servers into n clusters. The model consists of main controller, balancer and servers. The proposed system automatically supervises the load balancing work through load balancers, thus able to achieve high performance, stability, optimal resource utilization minimizes response time over the cloud environment.

Distributed Load Balancing Techniques

In distributed load balancing approach, all the servers in the cloud system are involved in making the load balancing decision. The distributed algorithms are scalable and have better fault tolerance. The distributed approach is preferred because elements of the network may vary in capacity or number during run time. Although the distributed approach is suitable for dynamic heterogeneous system, it increases the communication overhead to a large extent.

Kumar Nishant et al. [11] proposed an **Ant Colony Optimization (ACO)** algorithm, inspired from the ant colonies that work together in foraging behavior (The ants work together in search of new sources of food and simultaneously use existing food sources). In this ACO, first a Regional load balancing node (RLBN) is chosen, which will acts as a head node. The head node is selected in such a way that it has most number of neighboring nodes, as this helps ants to traverse in most possible directions of the network. Ants will use two types of pheromone for its movement, Foraging Pheromone (FP) and Trailing pheromone (TP). The ants after originating from the head node, by default do forward movement to trace an overloaded node and in the process they update FP trails according to a formula. Once an overloaded node is found, they do backward movement to find an under loaded node and update TP trails of the path. After reaching under loaded node, it updates data structure so as to move particular amount of load from overloaded node to an under loaded node. The ants then select a random neighbor of this node. If they encounter an over loaded node, they start doing forward movement to trace an under loaded node. Once an under loaded node is found, do backward movement to find previously encountered overloaded node. If it is still overloaded, then it updates data structure so as to move particular amount of load from overloaded node to an under loaded node. This task is repeated in a network to balance the load and improve the performance.

In this **Biased Random Sampling** [12] [18] load balancing technique, a virtual graph is constructed with each virtual node representing a server and the in-degree of the virtual node representing the server's free resources. The effectiveness of load distribution is considered by means of walk length w. An effective **w** threshold is around log (n) steps, where n is the network size.

The sampling walk starts at a specific node, at each step moving to a randomly selected neighbor node. When a node receives a job, if its current walk length is greater than or equal to walk length threshold, then a node executes a job by removing an incoming edge, decreasing its in-degree, indicating available resources are reduced. When the node completes a job, it creates a new incoming edge, indicating available resources are increased. If the jobs walk length is less than walk length threshold, then jobs walk length **w** value is incremented and send to a randomly selected next neighbor node.

This load balancing is both decentralized and easy to implement using standard network protocols as well as suitable for large network systems such as cloud computing. The performance of this load balancing can be further improved by orienting the random sampling towards specific nodes.

Active Clustering load balancing algorithm [12] [18] is a clustering based load balancing algorithm. The principle behind the active clustering is to groups similar nodes together and then work on this group. At a random point of time, a node becomes an initiator and selects a

matchmaker node randomly from its current neighbor, which is of a different type. The matchmaker node then creates a link between initiator and one of its neighbors which is of same type as initiator. The matchmaker node then detaches from the initiator and the process repeats iteratively to form a group of similar nodes and then work on this group**.** The performance of the system is enhanced with high availability of resources, thereby increasing the throughput.

Biologically inspired **Honeybee Foraging** algorithm [13] [12] [18] provides distributed solution for load balancing in a large scale complex network like cloud computing. (There is a class of bees called **Forager bees**. Forager bees are sent to find the suitable food source. Upon finding one, they come back to beehive to advertise this by means of waggle dance. The display of this dance tells quantity, quality and distance from the beehive. Scout bees then follow the foragers to the location of the food and begin to harvest it). Honey bee based load balancing technique uses collection of N servers partitioned into M groups called virtual servers v_0 ,v_1,- - - -v_{M-1}. There are M service request queues Q_0, Q1, - - - -Q_{M-1}, which buffers customer's requests to be served by respective virtual servers. A server can be either a forager or a scout. A dance floor of the hive be represented by an advert board, a waggle dance be represented by an advertisement and its duration by length of time an advertisement posting appears on the advert board.

A server $S_i \in V_j$, on completion of each request from a queue Q_j will attempt with the probability P to post an advert on an advert board. Also, it will attempt with the probability r_i to read a randomly selected advert from the advert board, if it is a forager or randomly select a V_j where j: 0, 1… (M-1) if it is a scout. A server servicing a request calculates its profit and compare with the colony profit. If profit was high, then server stays at current virtual server, posting an advertisement for it by a probability P. if it was low, then the server returns either to forager behavior or scout behavior.

Qiaomin Xie et. al. [14] proposed a novel class of algorithm called **Join-Idle-Queue** (JIQ) for distributed load balancing in large systems. The JIQ algorithm consists of primary load balancing system and secondary load balancing system, which communicates through a data structure called **I-Queue (Idle server queue)**. An I-Queue is a list of a subset of processors that have reported to be idle. Primary load balancing system exploits the information of idle servers present in the I-Queues and avoids the communication overhead with the servers. At a time of job arrival, the dispatcher consults its I-Queue. If the I-Queue is non-empty, then dispatcher removes first idle processor from I-Queue and directs the job to this idle processor. If I-Queue is empty, then dispatcher directs job to a randomly chosen processor. Secondary load balancing system balances idle processors across the dispatcher. When a processor becomes idle, it informs an I-Queue of its idleness or joins I-Queue. The paper proposed two secondary load balancing algorithms (JIQ-Random and JIQ-SQ (d)) in reverse directions to assign idle processors to the correct I-Queue so that there is high probability that the dispatcher will find an idle processor in its I-Queue. In JIQ-Random algorithm, idle processors choose an I-Queue uniformly at random. In JIQ-SQ (d), an idle processor chooses d random I-Queues and joins one with the smallest queue length.

III. Load Balancing Algorithms in Cloud with Virtual Machines [VM]

Another important concept that need to be considered in cloud computing is virtualization. Virtualization deals with the existence of the resources that are not physical. Through virtualization, physical servers can be partitioned into any number of virtual servers running their own operating systems in their allocated physical server and memory. This results in effective resource utilization, providing good response time, reducing operational cost.

Throttled load balancer [6] [17] maintains a record of the state of each virtual machines (busy/idle). When a request for allocation of virtual machine arrives, this balancer sends the ID of idle virtual machine to the data center controller and the data center controller allocates idle virtual machine for the request.

Active VM Load Balancer [6] [17] maintains information about each VM and the number of requests currently allocated to the VMs. When a request for the allocation of a new VM arrives, the balancer identifies the least loaded VM. If there are more than one, the first identified is selected. The balancer returns the VM id to the Data Centre Controller and the Data Centre Controller sends the request to the VM identified by that id. Data Center Controller notifies the balancer of the new allocation for table updation. When VM finishes processing the request, Data Center controller notifies the balancer for VM deallocation.

Meenakshi Sharma et al. [6] proposed a new E**fficient Virtual Machine Load Balancing Algorithm**. The proposed algorithm finds the expected response time of each resource (VM). When a request from the data center controller arrives, algorithm sends the ID of virtual machine having minimum response time to the data center controller for allocation to the new request. The algorithm updates the allocation table, increasing the allocation count for that VM. When VM finishes processing of request, data center controller notifies algorithm for VM deallocation. The experimental result compares proposed VM load balancing algorithm with the Throttled Load Balancer and Active VM Load Balancer. The efficient selection of a VM increases the overall performance of the cloud environment and also decreases the average response time and cost compare to Throttled Load Balancer and Active VM Load Balancer.

Jasmin James et al. [7] proposed **Weighted Active Monitoring Load Balancing (WALB) Algorithm** which has an improvement over the Active VM Load Balancer. This algorithm creates VM's of different processing

power and allocates weighted count according to the computing power of the VM. WALB maintains index table of VM's, associated weighted count and number of request currently allocated to each VM. When a request to allocate a VM arrives from the Data Center Controller, this algorithm identifies the least loaded and most powerful VM according to the weight assigned and returns its VM id to the Data Center Controller. The Data Center Controller sends a request to the identified VM and notifies the algorithm of allocation. The algorithm increases the count by one for that VM. When VM finishes processing, algorithm decreases the count of that VM by one. The experimental result shows that the proposed algorithm achieves better performance factors such as response time and processing time, but the algorithm does not consider process duration for each individual request.

Mintu M Ladani et. al. [8] proposed a new virtual machine load balancing algorithm '**Modified Weighted Active Monitoring Load Balancing Algorithm**'. This algorithm creates VM's of different processing power and allocates weighted count according to the computing power of the VM. It maintains index table of VM's, associated weighted count and number of request currently allocated to VM. When a request to allocate VM arrives from the Data Center Controller, this algorithm identifies VM with least loaded, least process duration and most powerful VM according to the weight assigned and returns its VM id to the Data Center Controller. Data Center Controller sends a request to the identified VM and notifies the algorithm of allocation. The algorithm increases the count by one for that VM. When VM finishes processing, algorithm decreases the count of that VM by one.

Modified Weighted Active Monitoring Load Balancing algorithm balances the load between the available VMs and considers most important factor process duration to achieve better performance parameters such as response time and processing time.

Vaidehi. M et. al. [28] proposed E**nhanced Load Balancing to Avoid Deadlock** technique to avoid deadlock among virtual machines while processing a request by migrating the virtual machine. The cloud manager in the data center maintains a data structure containing VM ID, job ID, and VM status. The VM status represents percentage of resource utilization. Cloud manager distributes the load as per the data structure and also analysis VM status routinely. If any VM is overloaded, which causes deadlock, then one or two jobs are migrated to a VM which is underutilized by tracking the data structure. If there are more than one available VM, then assignment is based on least hop time. On completion of the execution, the cloud manager automatically updates the data structure. The proposed algorithm yields less response time by VM migration from overloaded VM to underutilized VM by considering hop time to avoid deadlock without interacting with the data center controller in updating the data structure. This increases the number of jobs to be serviced by cloud provider, thereby improves working performance as well as business performance of the cloud.

Mayank Mishra et. al. [9] discusses **VM Migration Techniques** and their usage towards dynamic resource management in virtualized environment. The process of migration enables consolidation, load balancing and hotspot mitigation. The paper discusses critical factors such as when to migrate, which VM to migrate and where to migrate to achieve consolidation, load balancing and hot spot mitigation. The paper also presents details of migration heuristics aimed at reducing server sprawl, minimizing power consumption, balancing load across physical machines.

In **Round Robin Load Balancer** [2] [31], Data Center Controller assigns first request to a virtual machine, picked randomly from the group. Subsequently, it assigns requests to the virtual machines in circular order. Once request assigned to a virtual machine, then the virtual machine is moved to the end of the list. The advantage of Round Robin algorithm is that it does not require inter-process communication. Since the running time of any process is not known prior to execution, there is a possibility that some nodes may get heavily loaded.

Weighted Round Robin algorithm [2] is a modified version of Round Robin Load Balancer. This algorithm assigns a relative weight to all the virtual machines. If one VM is capable of handling twice as much load as the other, then the VM gets a weight of 2. In such cases, Data Center Controller will assign two requests to a VM with weight 2 against one request assigned to a VM with weight 1.

Cloud Hybrid Load Balancer [5] [20] is a framework for load balancing of websites in a cloud with Round Robin Domain Name System (RRDNS). This framework includes 3 main components, RRDNS virtual machines, Load Balancing System and Web System. RRDNS Virtual machines include all web IP information and these IP must be registered to global DNS provider. RRDNS virtual machines resolve IP information. Load balancing system receives http request and redirect them to the web system. Web system receives requests from load balancing system and then transfers data to the users.

1) The DNS of the ISP receives http request from user1.
2) The DNS of the ISP checks its registry database and sends http request to RRDNS Virtual Machine 1 (VM01).
3) RRDNS VM01 resolves IP information to user1.
4) User1 gets IP address of LBVM01 (Load Balance VM01) and connects to LBVM01. LBVM01 redirects request to web VM01 & finally user1 gets web data.

Since both LBVM01 and LBVM02 information is registered in both RRDNS VM01 and RRDNS VM02, next request is resolved by RRDNS VM 02, while RRDNS VM01 is busy.

Geetha V Megharaj [15] proposed a **Two Level Hierarchical Load Balancing** technique. This technique has Global Centralized Scheduler (GCS) at higher level and Local Centralized Scheduler (LCS) at next level to overcome high communication cost of distributed algorithms and single point of failure problem of

centralized algorithm. When a data center receives a new request for service, it queries GCS for allocation of virtual machine. The GCS collects the load information from every LCS and transfers a job to the appropriate LCS. This LCS collects the load information from the computing servers in that area and transfers the job to the appropriate computing server and thus balances the load in that local area. After computation, LCS collects results from every computing server and sends to GCS. In turn, GCS returns the results to the user.

Sudha Sadhasivam et. al. [19] present a **two level scheduler**, Efficient QoS based **Meta Scheduler** and Backfilling strategy based light weight **Virtual Machine Scheduler** for dispatching jobs. Meta Scheduler selects the data center based on user defined QoS specifications such as deadline and budget. Intra VMScheduler implements conservative backfilling by handling regular dispatch, backfill and backlog. Backfilling assures guaranteed start time for each job in the queue with a priority of dispatching smaller jobs. The paper also presents inter VMScheduler that creates VM with the required higher configuration by destroying the idle lower configuration VMs when there is a request for a higher configuration VM arrives. This minimizing the number of failures in VM creation and also reduces the job rejection ratio. The result demonstrate that conservative backfilling is highly suitable at the cluster level in grid as well as cloud computing.

Argha Roy et. al. [27] proposed **Dynamic Load Balancer** to avoid fault tolerance in cloud computing. Dynamic load balancer is used as an intermediate node between clients and cloud which monitors the load of each virtual machine in the cloud pool. When the users send the request to the dynamic load balancer, it gathers the processor utilization and memory utilization of each active server. If the processor utilization and memory utilization is less than 80%, the dynamic load balancer instantiates a new virtual machine on that server. Now, the request is assigned to this newly created VM. Otherwise, the algorithm instantiates a new VM on the next server with the lowest processor and memory utilization. The algorithm also checks fault occurrence of a server. If any fault occurs, then the VMs will be shifted to another server whose processor and memory utilization is less than 80%. The proposed dynamic load balancer algorithm achieves high scalability, dynamic load balancing, fault tolerance and low overhead.

Abhay Bhadani et. al. [29] proposed a C**entral Load Balancing Policy** with a central dispatcher which coordinates among all the active VMs to balance the load evenly based on global state information. The load information collector which is a daemon process runs continuously in each server collecting aggregate VM load periodically. Based on the data collected from the VMs, the server labels itself as Heavy (H), Moderate (M) or Light (L). Based on the information collected from each of the server, the central dispatcher takes decisions periodically for load balancing. Heavily loaded systems are first balanced with lightly loaded systems. The central dispatcher periodically executes load balancing algorithm for every N minutes to make system evenly balanced by instructing heavily loaded server to transfer lightly loaded VM to lightly loaded server. A simulation experiment is conducted by creating 3 VM's with similar configuration on 3 physical servers, each running a web server and applying load using http request. The simulation results show an overall improvement in the performance.

IV. Conclusion

Cloud Computing provides everything to the user as a service which includes application as a service, platform as a service and infrastructure as a service. One of the major issues in cloud computing is load balancing. Load balancing is required to distribute the load evenly among all servers in the cloud to maximize the resource utilization, increases throughput, to provide good response time, to reduce energy consumption. This paper discusses various centralized and distributed load balancing techniques in cloud computing environment.

V. References

[1] G. Ritchie, J. Levine, "A Fast, Effective Local Search for Scheduling Independent Jobs in Heterogeneous Computing Environments", Technical report, Centre for Intelligent Systems and their Applications, School of Informatics, University of Edinburgh, Edinburgh, 2003.

[2] Shanti Swaroop Moharana, Rajadeepan D. Ramesh, Digamber Powar, "Analysis of Load Balancers in Cloud Computing ", International Journal of Computer Science and Engineering (IJCSE), Volume 2, Issue 2, ISSN 2278-9960, pp 101-108, May 2013.

[3] Shu-Ching Wang, Kuo-Qin Yan, Wen-Pin Liao, Shun-Sheng Wang, "Towards a Load Balancing in a Three-level Cloud Computing Network", 3rd IEEE Conference on Computer Science and Information Technology[ICCSIT], Taiwan, Volume 1, pp 108-113, 9-11 July 2010, DOI: 10.1109/ICCSIT. 2010.5563889.

[4] Che-Lun Hung, Hsiao-hsi Wang, Yu-Chen Hu, "Efficient Load Balancing Algorithm for Cloud Computing Network", International Conference on Information Science and Technology (IST 2012), pp 251-253, April 28-30, 2012.

[5] Po-Huei Liang, Jiann-Min Yang, "Evaluation of Cloud Hybrid Load Balancer (CHLB)", International Journal of E-Business Development, Volume 3, Issue 1, pp 38-42, Feb 2013.

[6] Meenakshi Sharma, Pankaj Sharma, "Performance Evaluation of Adaptive Virtual Machine Load Balancing Algorithm", International Journal of

Advanced Computer Science and Applications, pp 86-88, Volume 3, Issue2, ISSN: 2156-5570, 2012.

[7] Jasmin James, Bhupendra Verma, "Efficient VM Load Balancing Algorithm for a Cloud Computing Environment", International Journal on Computer Science & Engineering, pp 1658-1663, Volume. 4, ISSN: 0975-3397, September 2012.

[8] Mintu M. Ladani, Vinit Kumar Gupta, "A Framework for Performance Analysis of Computing Clouds", International Journal of Innovative Technology and Exploring Engineering (IJITEE), pp 245-247, Volume 2, Issue 6, ISSN: 2278-3075, May 2013.

[9] Mayank Mishra, Anwesha Das, Purushottam Kulkarni, Anirudha Sahoo, "Dynamic Resource Management using Virtual Machine Migration", IEEE Communications Magazine, pp 34-40,Volume 50, Issue 9, September 2012 , DOI: 10.1109/MCOM.2012.6295709.

[10] Shu-Ching Wang, Kuo-Qin Yan, Shun-Sheng, Wang, Ching-Wei, Chen, "A Three-Phases Scheduling in a Hierarchical Cloud Computing Network", Third International Conference on Communications and Mobile Computing [CMC], Taiwan, pp 114-117, 18-20 April 2011, DOI: 10.1109/CMC.2011.28.

[11] Kumar Nishant, Pratik Sharma, Vishal Krishna,Chhavi Gupta, Kunwar Pratap Singh, Nitin, Ravi Rastogi, "Load Balancing of Nodes in Cloud Using Ant Colony Optimization", 14th International Conference on Computer Modeling and Simulation, pp 3-8, 2012, DOI: 10.1109/UKSim.2012.11.

[12] Martin Randles, David Lamb, A. Taleb-Bendiab, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing", 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp 551-556, 2010 DOI:10.1109/WAINA.2010.85.

[13] Sunil Nakrani, Craig Tovey, "On Honey Bees and Dynamic Server Allocation in Internet Hosting Centers", International Society for Adaptive Behavior, pp 223-240, Volume 12, Issue 3-4, September-December 2004, DOI: 10.1177/1059-7123.

[14] Yi Lu, Qiaomin Xie, Gabriel Kliot, Alan Geller, James R Larus, Albert Greenberg, "Join-Idle-Queue: A Novel Load Balancing Algorithm for Dynamically Scalable Web Services", Elsevier, Journal on Performance Evaluation, pp 1056-1071, Vol 68, Issue 11, Nov 2011, DOI: 10.1016/j.peva.2011.07. 015.

[15] Geetha C. Megharaj, Mohan K G, "Two Level Hierarchical Model of Load Balancing in Cloud", International Journal of Emerging Technology and Advanced Engineering, pp 307-311, Volume 3, Issue 10, ISSN 2250-2459, October 2013.

[16] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility", High Performance Computing and Communications, 10th IEEE International Conference, pp 5-13, September 2008, Digital Object Identifier: 10.1109/HPCC.2008.172.

[17] Meenakshi Sharma, Pankaj Sharma, Sandeep Sharma, "Efficient Load Balancing Algorithm in VM Cloud Environment", International Journal of Computer Science and Technology, pp 439-441, Vol 3, Issue 1, ISSN: 0976-8491[online], ISSN: 2229-433[print], Jan-March 2012.

[18] Ram Prasad Padhy, P Goutam Prasad Rao, "Load Balancing In Cloud Computing Systems", Thesis, Department of Computer Science and Engineering, National Institute of Technology, Rourkela-769 008, Orissa, India, May 2011.

[19] Sudha Sadhasivam, N Nagaveni, R. Jayarani, R. Vasanth Ram ,"Design and Implementation of an Efficient Two-Level Scheduler for Cloud Computing Environment", International Conference on Advances in Recent Technologies in Communication and Computing, pp 884-886, 2009, DOI: 10.1109/ARTCom.2009.148.

[20] Suresh M, Shafi Ullah Z, Santhosh Kumar B, "An Analysis of Load Balancing in Cloud Computing", International Journal of Engineering Research & Technology(IJERT), Vol 2, Issue 10, ISSN:2278-0181, October 2013.

[21] T. Kokilavani, D.I. George Amalarethinam, "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing", International Journal of Computer Applications, pp 43-49, Vol 20, No 2, April 2011, DOI: 10.5120/2403-3197.

[22] W Kleiminger, E Kalyvianaki, P Pietzuch, "Balancing Load in Stream Processing with the Cloud", 27th IEEE International Conference on Data Engineering Workshops[ICDEW], Zurich, Switzerland, pp 16-21, 11-16 April 2011, DOI: 10.1109/ ICDEW.2011.5767653.

[23] John W. Rittinghouse, James F.Ransome, "Cloud Computing: Implementation, management and Security", CRC Press, August 2009, ISBN: 9781439806807.

[24] Barrie Sosinsky, "Cloud Computing Bible", Wiley publishers, 1st edition, December 2010, ISBN: 978-0-470-90356-8.

[25] Divya Thazhathethil, Nishat Katre,Jyoti Mane-Deshmukh, Mahesh Kshirsagar, Anisaara Nadaph, "A

Model for Load Balancing by Partitioning the Public Cloud", International Journal of Innovative Research in Computer and Communication Engineering, pp 2466-2471, Vol. 2, Issue 1, ISSN(Online):2320-9801, ISSN(Print):2320-9798, January 2014.

[26] P.Jamuna, R.Anand Kumar, "Optimized Cloud Partitioning Technique to Simplify Load Balancing", International Journal of Advanced Research in Computer Science and Software Engineering, pp 820-822, Vol 3, Issue 11, ISSN: 2277128X, November 2013.

[27] Argha Roy, Diptam Dutta, "Dynamic Load Balancing: Improve efficiency in Cloud Computing", International Journal of Emerging Research in Management Technology, pp 78-82, Vol 2, Issue 4, ISSN:2278-9359, April 2013.

[28] Vaidehi. M, Rashmi. K. S, Suma. V, "Enhanced Load Balancing to Avoid Deadlock in Cloud", International Journal of Computer Applications on Advanced Computing and Communication Technologies for HPC Applications, pp 31-35, June 2012.

[29] Abhay Bhadani , Sanjay Chaudhary, "Performance Evaluation of Web Servers using Central Load Balancing Policy over Virtual Machines on Cloud", Proceedings of the Third Annual ACM Bangalore Conference, Article No. 16, ISBN: 978-1-4503-0001-8, January 2010, DOI: 10.1145/1754288.1754304.

[30] Uddalak Chatterjee, "A Study on Efficient Load Balancing Algorithms in Cloud Computing Environment", International Journal of Current Engineering and Technology, pp 1767-1770, Vol 3, ISSN 2277-4106, December 2013.

[31] Namrata Swarnkar, Atesh Kumar Singh, Shankar "A Survey of Load Balancing Technique in Cloud Computing", International Journal of Engineering Research & Technology, pp 800-804, Vol 2, Issue 8, August 2013.

[32] Demystifying_The_Cloud_eBook [1].pdf.

[33] Nidhi Jain Kansal, Inderveer Chana, "Cloud Load Balancing Techniques: A Step Towards Green Computing", International Journal of Computer Science, Vol 9, Issue 1, No 1, ISSN: 1694-0814, January 2012.

[34] Anthony T.Velte, Toby J.Velte, Robert Eisenpeter, "Cloud Computing: A Practical Approach", Tata McGraw-Hill Publishers, 1st Edition, 2009, ISBN: 0071626948.

[35] www.vmware.com/virtualization/what-is-virtualization.html.

[36] http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, The NIST definition of cloud computing, September 2011.