A STUDY ON VARIOUS LOAD BALANCING ALGORITHMS FOR RESPONSE TIME REDUCTION IN CLOUD ENVIRONMENT

Saurabh Gupta[1], Amit Dixit[2], Harsh Dev[3]
[1]Research Scholar, Uttarakhand Technical University Dehradun India
[2]Professor, Quantum Global Campus Roorkee Uttarakhand India
[3]Professor, PSIT, Kanpur India

Abstract
At present, internet has become part of life in human beings life. Internet is now a gigantic library that is composed of documents, files, images, videos, content and websites. Unlimited amount of data is added regularly to the library from different mediums and in different formats. Cloud computing is associated with internet computing. Cloud computing has undoubtedly benefited both service provider and clients in great extent. There is rapid increase in cloud's customers constantly. Although, the cloud data centers comprised of tremendous power but due to expeditious requests of users there is sudden need of balancing load. However, load balancing emerged as the conspicuous issue in the cloud heterogeneous environment.
A comprehensive literature survey on various load balancing algorithms is presented here which addresses that there can be reduction in response time and data center request processing time by using efficient load balancing policies.
Keywords: Cloud Computing, Throttled, Round Robin, Response Time, Service Broker, Load Balancing.

I. INTRODUCTION

Cloud computing has been defined by various researchers so there are many definitions available to us. As defined by the IBM Cloud computing, often referred to as simply ―the cloud,‖ is the delivery of on-demand computing resources—everything from applications to data centers— over the Internet on a pay-for-use basis [1].

Dubrovnik [2] implied cloud computing as ―a service-oriented architecture, reduced information technology overhead for the end-user, greater flexibility, and reduced total cost of ownership, on-demand services, and many other things‖.

A Berkeley Report in February 2009 states ―Cloud Computing, the long-held dream of computing as a utility has the potential to transform a large part of the IT industry, making software even more attractive as a service‘‘[3].

From the Quality of Service perspective, Clouds have been defined as a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLA‘s [4].

In a nutshell, cloud computing can be explained by the definition given by National Institute of Standards and Technology (NIST) which is accepted worldwide: ―Cloud computing is a model for enabling 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 [5].

A. Characteristics of Cloud computing

The elements of Cloud Computing are that it offers huge measures of force regarding processing and stockpiling while offering

enhanced versatility and flexibility. Besides, with proficiency and financial matters of scale, Cloud Computing administrations are getting to be a less expensive arrangement as well as a much greener one to fabricate and convey IT services [6].

According to NIST [5], cloud computing has the following essential characteristics which differentiate it from other computing paradigm:

- On-demand self-service: As said in NIST [5] a consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service‘s provider. Consumers must be able to obtain cloud services – at the infrastructure, platform, or application level – whenever they want, without requiring significant assistance. For the provider [7], on-demand self-service requires that procurement, account management, service instantiation, security control, service management, metering, billing and payment mechanisms are established. These mechanisms will interface with operational systems so that services are created, started, run, and stopped in accordance with the consumer's instructions. Use of a service by a consumer might last only for minutes, or for weeks, months, or years.
- Broad network access: As said in NIST [5] capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).Network access is needed to establish the initial provider/consumer relationship, for subsequent use of the cloud services themselves, and for use of added services that the consumer may implement using the cloud services. The user of a cloud service or added service might have a PC or a device such as a tablet, a PDA, or a mobile phone. These could have browsers or be browser-less devices. Or the cloud services could be integrated into a consumer enterprise‘s IT architecture, with access from large and sophisticated computer systems. This characteristic of cloud computing means that a company can implement added services that can be successfully used by anyone, anywhere on the globe, using a variety of devices [7].
- Resource pooling: As said in NIST [5] the provider‘s computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location-independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines. Multi-tenancy means that a single instance of a computing resource serves multiple client organizations (the tenants) providing a separate environment for each. Examples of resources include instances of infrastructure, platform, software, and application. The concept of resource pooling includes location-independence. For providers, having resources in convenient locations means lower costs and having them in diverse locations means that services can be maintained in the event of loss of a data center, of power, or of network connectivity.
- Rapid elasticity: As said in NIST [5] capabilities can be rapidly and elastically provisioned, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time. One of the key benefits of cloud computing is the ability to have a flexible computing service which can expand or contract in line with business demand, giving you capacity which would be impossible to generate from an in-house implementation without significant investment in resources. Resource pooling helps

providers to achieve elasticity. A resource that is no longer needed by one consumer can be allocated to another consumer that needs more resources.

- Measured Service: As said in NIST [5] cloud systems automatically control and optimize resource use by leveraging a metering capability1 at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service. If services are to be charged on a per-use basis, it is immediately obvious that usage must somehow be measured.

Consumers require sufficient measurements from their cloud computing service providers to enable them to make purchasing and operational judgments.

II. LOAD BALANCING APPROACHES

In load balancing there are two approaches and each approach have their own algorithms. The two approaches are as follows:

A. Static Load Balancing

In the Static load balancing approach, the load balancing decisions are made deterministically or probabilistically at compile time according to the performance of computing nodes and remain constant during runtime. Number of tasks in each node is fixed in this approach [8]. Static load balancing methods are non-preemptive i.e. once the load is allocated to the node it cannot be transferred to another node.

Advantages:

- It has minimum communication delays.
- Algorithms are simple and easy to implement.
- System overhead is minimized.

Disadvantages:

- Task cannot be migrated while execution.
- Overall performance of system decreased due to load fluctuations.
- Less useful when task have different execution time and nodes are heterogeneous.

Among various proposed algorithms major static load balancing algorithms are as follows:

1. Round Robin Algorithm

Round Robin Algorithm (RRA) assigns tasks sequentially and evenly to all the nodes. All tasks are assigned to computing nodes based on Round Robin order, meaning that computing nodes choosing is performed in series and will be back to the first computing node if the last computing node has been reached [8]. Each node maintains its load index locally independent of allocations from remote node.

It is the simplest algorithm that uses the concept of time quantum or slices. Here, time is divided into multiple slices and each node is given a particular time quantum and within this time quantum the node will perform its operations.

In Cloud Data centers this algorithm works on random selection of the virtual machines. The datacenter controller allocates the requests to a list of VMs on a rotating basis. The first request is allocated to a VM chosen randomly from the group and then the Data Center controller assigns the requests in a circular order. Once the VM is allotted the request, the VM is shifted to the end of the list [9].

Round-robin is by far the simplest algorithm available to distribute load among nodes. Because of this reason it is often the first preference when implementing a simple scheduler. One of the reasons for it being so simple is that the only information required is a list of nodes.

The round robin algorithm is as follows:

Step1: Round Robin VM load Balancer maintains an index of VMs. At start all VM‘s have zero allocation.

Step 2:

a. The data center controller receives the user requests/cloudlets.

b. The requests are allocated to VMs in circular way.

c. The round robin VM load balancer will allocate the time quantum for user request execution.

Step 3: After the execution of cloudlets, the VMs are de-allocated by the Round Robin VM Load balancer.

Step 4: The data center controller checks for new /pending/waiting requests in queue.

Step 5: Continue from step-2.

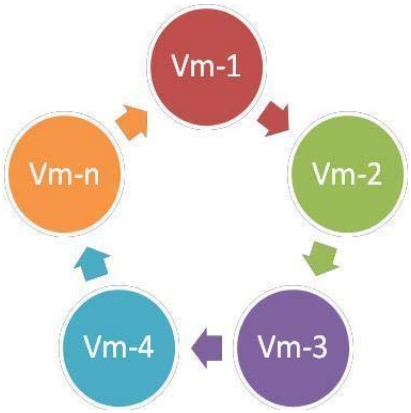

Figure 1 Round Robin Load Balancing

Advantages:

- Inter-process communication is not required.
- Useful for jobs of equal processing time and nodes of same capabilities.

Disadvantages:

- Without checking the capacity of sever, it‘s directly assign the request (like whether it is over loaded or not).
- It does not include the state of previous allocation of a VM to a request [10]
- Not useful when tasks have unequal processing time.
- Not useful when nodes have different capacities.

2. Randomized Algorithm

Randomized Algorithm (RA) uses random numbers in selecting computing nodes for processing, without having any information about the current or previous load on the node. The computing nodes are selected randomly following random numbers generated based on a statistic distribution [27, 23].

Advantages:

- It works well for particular special purpose applications.
- No inter process communication is required.

Disadvantages:

- It is not considered elegant solution.
- Maximum response time among all algorithms.

3. Central Manager Algorithm

In each step in Central Manager Algorithm (CMA), central node selects slave node to be assigned a task. Slave node having least load is being selected. The central node maintains the load index of all slave nodes connected to it. Whenever, load is changed, a message is send by the slave nodes to the central node [13]. The load manager makes load balancing decisions based on the system load information, allowing the best decision when of the process created [27, 24].

Advantages:

- Performs well when dynamic activities are created by different hosts.

Disadvantages:

- CMA needs high level of inter process communication.
- Single point of failure.

4. Threshold Algorithm

In Threshold Algorithm (TA) tasks/processes are assigned immediately upon creation to the computing nodes (processors). Computing nodes for new processes are selected locally without sending remote messages. Each node keeps a private copy of the system‘s load information. The load of a computing node can be characterized by one of the three levels which are: under loaded, medium and overloaded. Two threshold parameters t_under and t_upper can be used to describe these levels [11]:

Under loaded: workload < t_under,
Medium: t_under ≤ workload ≤ t_upper,
Overloaded: workload > t_upper.

In the initial stage, all the computing nodes are considered under loaded. Whenever the load state of a computing node exceeds the load level threshold, then it sends messages regarding the new load state to all of the other computing nodes, regularly updating them so that as the actual load state of the entire system can be known to every node.
If the local state is not overloaded then the process is allocated locally. Otherwise, a remote under loaded processor is selected, and if no such host exists, the process is also allocated locally.

Advantages:

- It has low inter process communication.
- Performance is improved because of large number of local process allocations which decreases the overhead of remote process allocations and remote memory accesses.

Disadvantages:

Whenever all remote processors are overloaded then all processes are allocated locally causing two major problems:-

- One overloaded processor may have much higher load than other overloaded processors, causing significant disturbance in load balancing.
- Increased execution time of an Application.

B. Static Load Balancing

There are three general classes of problems where a static load balancing is either impossible or can lead to imbalance of load, problems are:

The first class consists of problems in which all the tasks are available at the beginning of the computation but the amount of time required by each task is different.

The second class consists of problems in which tasks are available at the beginning but as the computation progresses, the amount of time required by each task changes.

The third class consists of problems in which tasks are not available at the beginning but are generated dynamically.

In static load balancing too much information about task and system is required before execution which is not possible every time like in these three classes of problems. So Dynamic load balancing was developed to address these constraints.

Dynamic load balancing makes more informative load balancing decisions during execution by the runtime state information [12]. In dynamic load balancing algorithms work load is distributed among the processors at runtime. These algorithms monitor changes on the system work load and redistribute the work accordingly.

Advantages:

- Dynamic load balancing works well for heterogeneous systems.
- Task can be redistributed to any processor while run time hence overloading and under loading problems become minimum.
- It works well for task having different execution time.
- The system need not be aware of run-time behavior of the applications before execution.

Disadvantages:

- High communication over heads occurs and becomes more when number of processors increase.
- Dynamic load balancing algorithms are complex hence not very easy to implement.
- System overhead increases because it is preemptive.

In general, dynamic load-balancing algorithms can be broadly categorized as follows:

i. Centralized or Decentralized

In centralized approach, only one processor acts as the main or central controller (Master). Master node process or holds the collection of tasks to be performed. Master node has global view on the load information of all nodes connected to it, and decides how to allocate jobs to each of the nodes. And rest of the nodes act as slaves [15]. Tasks are sent to the slave processes for execution.

It has advantage of ease of implementation, but suffers from single point of failure.

In Decentralized approach, all nodes in the system are involved in making the load balancing decision. The state information is spread among all the nodes and nodes are responsible in managing their own resources or allocating tasks residing in their queues to other nodes. It is more scalable and has better fault tolerance.

i. Cooperative or Non-cooperative

If a distributed load balancing algorithm is adopted, the next issue that should be considered is whether the nodes involved in the task scheduling process are working cooperatively or independently (non-cooperatively). In the non-cooperative approach, each node is independent has autonomy over its own resource scheduling, that is, decisions are made independently of the rest of the system and therefore the node may

migrate or allocate tasks based on local performance.
In the cooperative scheme, each load balancing decision maker (scheduler) has the responsibility to carry out its own portion of the scheduling task, but all schedulers are working toward a common system wide goal [8].

ii. Adaptive or Non-Adaptive

Dynamic load balancing is said to be adaptive if scheduled decisions take into consideration past and current system performance and are affected by previous decisions or changes in the environment.
It is said to be non-adaptive if parameters used in load balancing remain the same regardless of system's past behavior.

iii. Sender/Receiver/Symmetrical Initiated

Distribution of task in dynamic load balancing can be sender initiated, receiver initiated or symmetrical initiated. In sender initiated strategy, Congested nodes attempt to move work towards under-loaded nodes. In receiver-initiated strategy, under-loaded nodes request tasks to be sent to them from nodes with higher loads. In the symmetric approach, both the under-loaded as well as the overloaded nodes may initiate load transfers.

There are three basic dynamic load balancing algorithms which are as following:

1. Central Queue Algorithm

Central Queue Algorithm [14] stores new activities and unfulfilled requests in a cyclic FIFO queue on the main host. Each new activity arriving at the queue manager is inserted into the queue. Then, whenever a 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 there are no ready activities in the queue, the request is buffered, until a new activity is available.
When a processor load falls under the threshold, the local load manager sends a request for a new activity to the central load manager. The central load manager answers the request immediately if a ready activity is found in the process-request queue, or queues the request until a new activity arrives. This is a centralized initiated algorithm and need high communication among nodes.

2. Local Queue Algorithm

This algorithm supports inter process migration. The main concept in local queue algorithm is static allocation of all new process with process migration initiated by the host when its load falls under the predefined minimum number of ready processes (threshold limit). Initially, new processes created on the main host are allocated on all under loaded hosts. From then on, all the processes created on the main host and all other hosts are allocated locally [31, 24].
When the local host gets under load it request for the activities from the remote hosts. The remote hosts than look up its local list for ready activities and compares the local number of ready activities with the received number. If the former is greater than the latter, then some of the activities are passed on to the requestor host and get the acknowledgement from the host. This is a distributed co-operative algorithms and requires inter-process communication but lesser as compared to central queue algorithm.

3. Least Connection Algorithm

This algorithm decides the load distribution on the basis of connections present on a node [16]. The load balancer keeps track of the numbers of connections attached to each node and selects the node with least number of connections for load transfer. The number increases when a new connection is established and decreases when connection finishes or time out.

Least Connections methods work best in environments where the nodes have similar capabilities. Load imbalance may be caused when the tasks are all of different durations because Connection counting just doesn‘t account for that scenario very well.

4. Throttled Algorithm

Throttled load balancer is a dynamic load balancing algorithm. Throttled load balancer maintains an index table of all VMs and their respective state (i.e. available or busy). Whenever new request arrives the table is parsed by load balancer and VM having available state is chosen and its VM id is returned to data center controller which further assigns the request to that particular VM, if suitable VM is not found then -1 is returned to data center controller. The notification of new allocation after allocating request to VM and VM de-allocation after

completing of request is sent by data center controller to load balancer [17, 18].
The throttled algorithm is as follows:

Step1: Throttled VM Load Balancer keeps an index table of VMs and the state of the VM (BUSY/AVAILABLE). At the beginning all VM's are available.

Step2: Data Center Controller receives a new request.

Step3: Data Center Controller queries the Throttled VM Load Balancer for the next allocation.

Step4: Throttled VM Load Balancer parses the allocation table from top until the first available VM is found or the table is parsed completely.
If found:
i. The Throttled VM Load Balancer returns the VM id to the Data Center Controller.
ii. The Data Center Controller sends the request to the VM identified by that id.
iii. Data Center Controller notifies the Throttled VM Load Balancer of the new allocation.
iv. Throttled VM Load Balancer updates the allocation table accordingly.

If not found:
i. The Throttled VM Load Balancer returns -1.
ii. The Data Center Controller queues the request.

Step5: When the VM finishes processing the request, and the Data Center Controller obtains the response cloudlet, it informs the Throttled VM Load Balancer of the VM de-allocation.

Step6: The Data Center Controller checks if there are any waiting requests in the queue. If there are, it carries on from step 3.
Step7: Continue from step 2 [10].

Drawback:

- In throttled algorithm, where the index table is resolve from the first index every time when the data center queries load balancer for allocation of VM.
- It does not takes into account the advanced load balancing requirements such as processing times for each individual requests [19].

5. Equally Spread Current Execution

This algorithm distributes the tasks among VMs in a way to even out the number of active tasks at any given time on each VM. This load balancing algorithm is also known as Active Monitoring Load Balancing algorithm. It works quiet similar to throttled algorithm but with change in the VM index table. In this algorithm load balancer maintains an index table of all VMs along with the number of currently allocated requests to VM. Whenever new request arrive load balancer parses the table and VM having least load is chosen and its ID is returned to data center controller which further assigns the request to that VM and notifies the load balancer of this new allocation to increase the allocation count of that VM. After request gets completed load balancer is further notified about de-allocation of VM so that it decreases the allocation count of that VM [17, 18].
The ESCE Load Balancing algorithm is as follows:

Step 1: Find the next available VM.

Step 2. Check for all current allocation count is less than max length of VM list allocate the VM.

Step 3: If available VM is not allocated create a new one.
Step 4: Count the active load on each VM.

Step 5: Return the id of those VM which is having least load.

Step 6: The VM Load Balancer will allocate the request to one of the VM.

Step 7: If a VM is overloaded then the VM Load Balancer will distribute some of its work to the VM having least work so that every VM is equally loaded.

Step 8: The data center controller receives the response to the request sent and then allocates the waiting requests from the job pool/queue to the available VM & so on.

Step 9: Continue from step-2 [10].

Drawback:

- AMLB always find least loaded VM for assign new incoming request but it will not check whether it's previously utilized or not (so some VM over utilized and some are still ideal).

Now we will further describe existing Service broker algorithms which are further used for analysis.

6. Service Proximity based algorithm

It is also known as closest data center algorithm. Here service broker selects the data center which is closest to the request sender's location taking into consideration transmission latency [17, 20]. This is the simplest Service Broker implementation. Algorithm steps are as follows.

1. Service Proximity Service Broker maintains an index table of all Data Centers indexed by their region.
2. When the Internet receives a message from a user base it queries the Service Proximity Service Broker for the destination Data Center Controller.

3. The Service Proximity Service Broker retrieves the region of the sender of the request and queries for the region proximity list for that region from the Internet Characteristics. This list orders the remaining regions in the order of lowest network latency first when calculated from the given region.

4. The Service Proximity Service Broker picks the first data center located at the earliest/highest region in the proximity list. If more than one data center is located in a region, one is selected randomly.

1. Performance Optimized algorithm

It is also known as Optimize response time algorithm. Here service broker selects the data center according to best response time [17, 20]. This policy is implemented by the Best Response Time Service Broker, which extends the Service Proximity Service Broker. The algorithm steps are as follows.

1. Best Response Time Service Broker maintains an index of all Data Centers available.

2. When the Internet receives a message from a user base it queries the Best Response Time Service Broker for the destination Data Center Controller.

3. The Best Response Time Service Broker identifies the closest (in terms of latency) data center using the Service Proximity Service Broker algorithm.

4. Then the Best Response Time Service Broker iterates through the list of all data centers and estimates the current response time at each data center by:
i. Querying the last recorded processing time from Internet Characteristics.
ii. If this time is recorded before a predefined threshold, the processing time for that data center is reset to 0. This means the data center has been idle for duration of at least the threshold time.
iii. The network delay from Internet Characteristics is added to the value arrived at by above steps.

If the least estimated response time is for the closest data center, the Best Response Time Service Broker selects the closest data center. Else, Best Response Time Service Broker picks either the closest data center or the data center with the least response time with a 50:50 chance (i.e. load balanced 50:50).

III. COMPARISON ON DIFFERENT ALGORITHMS

Now we will compare the above given algorithms of static and dynamic load balancing considering the following parameters:

- Nature: It tells whether algorithm is static or dynamic.
- Overhead: It is amount of overheads like inter-process communication, migration of tasks etc. involved while implementing the algorithm and should be minimum.
- Resource Utilization: It tells whether the algorithm is able to utilize all the resources optimally or not means less idle processors.
- Process Migration: It tells when a system will migrate its process. The algorithm is capable of deciding when it should make changes of load distribution during execution of process or not.

- Fault Tolerant: It tells whether the algorithm can work continuously in event of failure or not, performance of algorithm is decreasing or not.
- Response Time: It is time a distributed system using a particular load balancing algorithm is taking to respond and must be less.
- Waiting Time: It is the time period spent waiting in the ready queue and should be less.
- Centralized or decentralized: It tells whether algorithm is centralized or decentralized.
- Cooperative: It defines the extent of independence that each processor has in concluding that how should it can use its own resources.
- Adaptability: It tells if algorithm can adapt to changing situations.

Algorithms **Parameters**	**Round Robin**	**Random ized**	**Central Manage r**	**Thresh old**	**Central Queue**	**Local Queue**	**Least Connecti on**
Nature	Static	static	Static	dynamic	dynamic	dynamic	Dynamic
Overhead	Low	low	Low	high	high	high	High
Resource Utilization	Less	less	Less	less	less	more	More
Process Migration	No	no	No	no	no	yes	No
Fault Tolerant	No	no	Yes	no	yes	yes	No
Response Time	less	less	Least	less	more	more	Less
Waiting Time	more	more	More	more	less	less	Less
Centralized/ Decentralized	D	D	D	D	C	C	D
Adaptability	less	less	Less	less	more	more	More

Figure 2 Comparison Chart of various Load Balancing Algorithms

References

[1] Cloud computing [online] available at http://www.ibm.com/cloud-computing/what-is-cloud-computing.html

[2] Mladen A. Vouk, ―Cloud computing - issues, research and implementations‖, In Proceedings of the 30th International Conference on Information Technology Interfaces, ITI '08, pages 31–40, Dubrovnik, Croatia, June 2008.

[3] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A.Rabkin, I. Stoica, and M. Zaharia, ―Above the Clouds: A Berkeley View of Cloud Computing‖, University of California at Berkley, USA, Technical Rep UCB/EECS-2009-28, 2009.

[4] Tikotekar, G. Vallee, T. Naughton, H. Ong, C. Engelmann, S. Scott, A. Filippi, ―Effects of virtualization on a scientific application running a hyperspectral radiative transfer code on virtual machines‖, in Proceedings of 2nd Workshop on System-Level Virtualization for High Performance Computing (HPCVirt 08), pp. 16–23, UK, 2008.

[5] Lee Badger , Tim Grance, Robert Patt-Corner and Jeff Voas, "Cloud Computing Synopsis and Recommendations", National Institute of Standards and Technology, Information Technology Laboratory, NIST Special Publication 800-146, 81 pages, May 2012.

[6] J. Grimes, P. Jaeger, J. Lin, "Weathering the storm: The policy implications of cloud computing", in Proceedings of Conference, University of North Carolina (Chapel Hill, USA, 2009).

[7] Cloud for business [online] available at http://www.opengroup.org/cloud/cloud/cloud_for_business/what.htm

[8] S. F. El-Zoghdy and S. Ghoniemy, ―A Survey of Load Balancing In High-Performance Distributed Computing Systems‖, International Journal of Advanced Computing Research, Volume 1, 2014.

[9] MR.Manan D. Shah, MR.Amit A. Kariyani and MR.Dipak L. Agrawal, ―Allocation Of Virtual Machines In Cloud Computing Using Load Balancing Algorithm‖, International Journal of Computer Science and Information Technology & Security, Vol. 3, No.1, February 2013.

[10] Nusrat Pasha, Dr. Amit Agrawal and Dr. Ravi Rastogi, ―Round Robin Approach for VM Load Balancing Algorithm in Cloud Computing

Environment‖, IJARCSSE, Volume 4, pages 34-39 Issue 5, May 2014.
[11] Daniel Grousa and Anthony T., ―Non-Cooperative load balancing in distributed systems‖, Journal of Parallel and Distributing Computing, 2005.
[12] Mohsen and Hossein Delda, ―Balancing Load in a Computational Grid Applying Adaptive, Intelligent Colonies of Ants‖, Informatica 32 (2008) 327–335.
[13] Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma, ―Performance Analysis of Load Balancing Algorithms‖, World Academy of Science, Engineering and Technology,2008.
[14] William Leinberger, George Karypis and Vipin Kumar, "Load Balancing Across Near-Homogeneous Multi-Resource Servers", in proc. of heterogeneous computing workshop (HCW 2000), IEEE, pp: 60-71, 2000.
[15] Mr.Gaurav Sharma and Jagjit Kaur Bhatia, ‖A review on different approaches for load balancing in computational grid‖, Journal of Global Research in Computer Science, Volume 4, No. 4, April 2013.
[16] Suriya and Prashanth,‖ Review of load balancing in cloud computing‖, IJCS, vol.10 issue.1, Jan 2013.
[17] B. Wickremasinghe, R.N.Calheiros, R.Buyya, ―Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing‖, in proc. of the 24th International Conference on Advanced Information Networking and Applications (AINA 2010), Perth, Australia, 2010.
[18] Sandeep patel, Ritesh Pael and HetalPatel, ―CloudAnalyst : A Survey of Load Balancing Policies‖, International Journal of Computer Applications (0975 – 8887) Volume 117 – No. 21, May 2015.
[19] Bhatia, J., Patel, T., Trivedi, H., and Majmudar, V,―HTV Dynamic Load Balancing Algorithm for Virtual Machine Instances in Cloud‖, In Cloud and Services Computing (ISCOS), 2012 International Symposium, IEEE, pp: 15-20, December 2012.
[20] Rakesh Kumar Mishra, Sandeep Kumar and Sreenu Naik B, ―Priority Based Round-Robin Service Broker Algorithm For Cloud- Analyst in proc. International Advance
Computing Conference (IACC), IEEE, pp.878-881, February 2014.
[21]Soumi Ghosh, Chandan Banerjee, "Priority based Modified Throttled Algorithm in Cloud Computing", 2016 International Conference on Inventive Computation Technologies (ICICT), Year: 2016, Volume: 3
[22]Lahar Singh Nishad, Sarvesh Kumar, Sumit Kumar Bola, Sumitra Beniwal, Ankita Pareek, "Round robin selection of datacenter simulation technique cloudsim and cloud analyst architecture and making it efficient by using load balancing technique", 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)
[23]Akash Dave, Bhargesh Patel, Gopi Bhatt,2016 International Conference on Communication and Electronics Systems (ICCES), Year: 2016
[24]Ojasvee Kaneria, R K Banyal, "Analysis andimprovement
of load balancing in Cloud Computing", 2016 International Conference on ICT in Business Industry & Government (ICTBIG), Year: 2016
[25]Manoel C. Silva Filho, Raysa L. Oliveira, Claudio C. Monteiro, Pedro R. M. Inácio, Mário M. Freire, "CloudSim Plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness", 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM),
2017,Pages: 400 - 406, DOI: 10.23919/INM.2017.7987304
[26]Umang Thakkar, Indrjeet Rajput, "A novel approach for dynamic selection of load balancing algorithms in cloud computing", 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), Year: 2016
[27]Lei Yu, Liuhua Chen, Zhipeng Cai, Haiying Shen, Yi Liang, Yi Pan, "Stochastic Load Balancing for Virtual Resource Management in Datacenters", IEEE Transactions on Cloud Computing, Year: 2017, Volume: PP, Issue: 99, Pages: 1 - 10.1109/TCC.2016.2525984
[28]Roberto Beraldi, Abderrahmen Mtibaa, Hussein Alnuweiri, "Cooperative load balancing scheme for edge computing resources", 2017 Second International Conference on Fog and Mobile Edge Computing(FMEC),

Year:2017,Pages: 94 100, DOI: 10.1109/FMEC.2017.7946414
[29]Awatif Ragmani, Amina El Omri, Noreddine Abghour, Khalid Moussaid, Mohammed Rida, "A global performance analysis methodology: Case of cloud computing and logistics", 2016 3rd International Conference on Logistics Operations Management (GOL), Year: 2016
[30] Peter Mell, Timothy, Grance, "The NIST Definition of "Cloud computing", National Institute of Standards and Technology Computing Security Resource Center (www.csre.nist.gov)
[31] Manoj Agnihotri, Sahil Sharma, "Execution analysis of load balancing particle swarm optimization algorithm in cloud data center", 2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC), Year: 2016
[32]Saleh Atiewi, Salman Yussof, Mohd Ezanee, Muder Almiani, "A review energy-efficient task scheduling algorithms in cloud computing", 2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT),Year:2016,Pages: 1 6, DOI: 10.1109/LISAT.2016.7494108
[33]Qi Liu, Weidong Cai, Jian Shen, Dandan Jin, Nigel Linge, "A load-balancing approach based on modified K-ELM and NSGA-II in a heterogeneous cloud environment", 2016 IEEE International Conference on Consumer Electronics (ICCE), Year: 2016
[34]Yatheendraprakash Govindaraju, Hector Duran-Limon, "A QoS and Energy Aware Load Balancing and Resource Allocation Framework for IaaS Cloud Providers", 2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing(UCC), Year: 2016
[35]Yan-Wun Qiu, Jen-Ing G. Hwang, "A Two-Level Load Balancing Method with Dynamic Strategy for Cloud Computing", 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), Year: 2016
[36]Jiadong Zhang, Qiongxin Liu, Jiayu Chen,"Advanced Load Balancing Strategy for Cloud Environment",2017 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), Year: 2016
[37]Shreenath Acharya, Demian Antony D'Mello,"Enhanced dynamic load balancing algorithm for resource provisioning in cloud", 2016 International Conference on Inventive Computation Technologies (ICICT), Year: 2016, Volume: 2
[38]Anureet Kaur, Bikrampal Kaur, "Load balancing in tasks using honey bee behavior algorithm in cloud computing", 2016 5th International Conference on Wireless Networks and Embedded Systems (WECON), Year: 2016
[39]Shubham Sidana, Neha Tiwari, Anurag Gupta, Inall Singh Kushwaha, "NBST algorithm: A load balancing algorithm in cloud computing", 2016 International Conference on Computing, Communication and Automation (ICCCA), Year: 2016
[40]Upasana Lakhina, Niharika Singh, Ajay Jangra,"An efficient load balancing algorithm for cloud computing using dynamic cluster mechanism", 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), Year: 2016, Pages: 1799 – 1804
[41]Hussain A Makasarwala, Prasun Hazari, "Using genetic algorithm for load balancing in cloud computing", 2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Year: 2016
[42]M. Buvaneswari, M. P. Loganathan, S. Sangeetha, "Cloud challenges of load balancing and security issues using ICLoS algorithm", 2017 2nd International Conference on Computing and Communications Technologies (ICCCT), 2017, Pages: 103 105, DOI: 10.1109/ICCCT2.2017.7972254
[43]Ronak R Patel , Swachil J Patel , Dhaval S Patel , Tushar T Desai , "Improved GA using Population Reduction for Load Balancing in Cloud Computing", Intl. Conference on Advances in Computing, Communications and Informatics (ICACCI), Sept. 21-24, 2016, Jaipur, India, 2016.
[44]M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," Journal of

Network and Computer Applications, vol. 66, pp. 106–127, may 2016.

[45]Manoel C. Silva Filho, Raysa L. Oliveira, Claudio C. Monteiro, Pedro R. M. Inácio, Mário M. Freire, "CloudSim Plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness", 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM),2017,Pages: 400 - 406, DOI: 10.23919/INM.2017.7987304

[46]Lei Yu, Liuhua Chen, Zhipeng Cai, Haiying Shen, Yi Liang, Yi Pan, "Stochastic Load Balancing for Virtual Resource Management in Datacenters", IEEE Transactions on Cloud Computing, Year: 2017, Volume: PP, Issue: 99, Pages:1- 1, DOI: 10.1109/TCC.2016.2525984

[47]Roberto Beraldi, Abderrahmen Mtibaa, Hussein Alnuweiri, "Cooperative load balancing scheme for edge computing resources", 2017 Second International Conference on Fog and Mobile Edge Computing(FMEC),Year:2017,Pages:94-100, DOI: 10.1109/FMEC.2017.7946414