PUNE RESEARCH SCHOLAR ISSN 2455-314X
AN INTERNATIONAL MULTIDISCIPLINARY JOURNAL VOL 2, ISSUE 5

EFFICIENT LOAD BALANCING ALGORITHMS FOR EFFICIENT UTILIZATION OF RESOURCES IN CLOUD

R. DENIS
Assistant Professor,
PG Dept. of Computer Science
Sacred Heart College (Autonomous),
Tirupatur, Tamil Nadu, **INDIA**

S. JOHN BOSCO
Assistant Professor,
PG Dept. of Computer Science
Sacred Heart College (Autonomous),
Tirupatur, Tamil Nadu, **INDIA**

ABSTRACT

Cloud Computing is new emerging cost cutting service accessed via internet. It has become popular to provide various services to user such as multi-media sharing, on-line office software, game and on-line storage. It aims to share data, calculations, and service transparently over a scalable network of nodes. Since Cloud computing stores the data and disseminated resources in the open environment, the amount of data storage increases quickly. Therefore storage, load balancing is an important key issue in cloud computing. It would consume a lot of cost to maintain load information, since the system is too huge to timely disperse load. Load balancing is one of the main challenges in cloud computing which is required to distribute the dynamic workload across multiple nodes to ensure that no single node is overwhelmed. It helps in optimal utilization of resources and hence in improving the performance of the system. A few existing scheduling algorithms can maintain load balancing and provide better strategies through efficient job scheduling and resource allocation techniques as well. This paper discusses some of the existing load balancing algorithms in cloud computing and also their contests. In order to gain maximum profits with optimized dynamic load balancing we propose a new balancing algorithm for smoother distribution and efficient utilization of resources.

Keywords *–Algorithms, Cloud Storage, Dynamic Load Balancing, Scheduling.*

I. INTRODUCTION

A Cloud computing is emerging as a new paradigm of large scale distributed computing. It has moved computing and data away from desktop and portable PCs, into large data Centre's. It provides the scalable IT resources such as applications and services, as well as the infrastructure on which they operate, over the Internet, on pay-per-use basis to adjust the capacity quickly and easily. It helps to accommodate changes in demand and helps any organization in avoiding the capital costs of software and hardware . Thus, Cloud Computing is a framework for enabling a suitable, on-demand network access to a shared pool of computing resources (e.g. networks, servers, storage, applications, and services). These resources can be provisioned and de-provisioned quickly with minimal management effort or service provider interaction. This further helps in promoting availability. Due to the exponential growth of cloud computing, it has been widely adopted by the industry and there is a rapid expansion in data-center.

II. CHARACTERISTICS OF CLOUD COMPUTING

According to the National Institute of Standards and Technology (NIST), cloud computing exhibits several characteristics:

On-demand Self-service- A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

Broad Network Access- 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).

Resource Pooling- The providers 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 datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

Rapid Elasticity- Capabilities can be elastically provisioned and released, 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.

Measured Service- Cloud systems automatically control and optimize resource use by leveraging a metering capability 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.

III. PROBLEM STATEMENT

As cloud computing is in its evolving stage, so there are many problems prevalent in cloud computing . Such as:

- Ensuring proper access control (authentication, authorization, and auditing)
- Network level migration, so that it requires minimum cost and time to move a job
- To provide proper security to the data in transit and to the data at rest.
- Data obtainability issues in cloud
- Data integrity and confidentiality
- Legal bog and transitive trust issues
- Data lineage, data provenance and inadvertent disclosure of sensitive information is possible.
- Load balancing

Among the above listed problems Load Balancing is a computer networking method to distribute workload across multiple computers or a computer cluster, network links, central processing units, disk drives, or other resources, to achieve optimal resource utilization, maximize throughput, minimize response time, and avoid overload. Using multiple components with load balancing, instead of a single component, may increase reliability through redundancy. The load balancing service is usually provided by dedicated software or hardware, such as a multilayer switch or a Domain Name System server. Load balancing is one of the central issues in cloud computing.

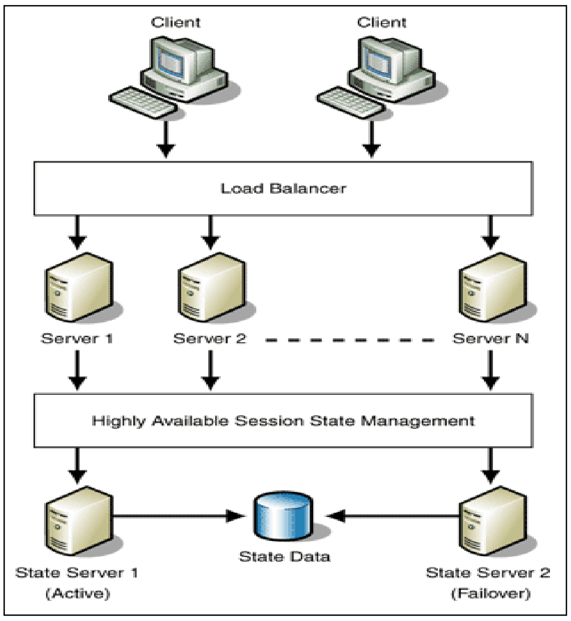
It is a mechanism that distributes the dynamic l workload evenly across all the nodes in the whole cloud to avoid a situation where some nodes are heavily loaded while others are idle or doing little work. It helps to achieve a high user satisfaction and resource utilization ratio, hence improving the overall performance and resource utility of the system. It also ensures that every computing resource is distributed efficiently and properly. It further prevents bottlenecks of the system which may occur due to load imbalance. When one or more components of any service fail, load balancing helps in continuation of the service by implementing fair-over, i.e. in provisioning and de-provisioning of instances of applications without fail.

3.1 Maneuver/Task of Load Balancer

The goal of load balancing is improving the performance by balancing the load among these various resources (network links, central processing units, disk drives.) to achieve optimal resource utilization, maximum throughput, maximum response time, and avoiding overload. To distribute load on different systems, different load balancing algorithms are used.

In order to distribute the necessary tasks, load balancers go through a series of steps. First, the load balancer will query the available servers to ensure their availability. The load balancer pings a server, and if the expected response occurs, it will be included in the available list. If the server fails to respond, it will not be used until another test is performed and it returns with the appropriate response. Load balancing software is very flexible in this environment, as the administrator can quickly tweak the system to ensure it is checking servers appropriately and accurately.

Distributing the load between the active servers can be done in several different ways. The load balancer may use a round-robin method, where each server is used in turn. It can also use a weighted round robin system, where servers are assigned traffic based on their configured capabilities.



3.2 Load Balancing Approaches

There are different variations of load balancing. Link load balancing, network load balancing and server load balancing are among the most common forms. These load balancing forms operate on the same basic principle - distributing tasks and processing among servers to ensure availability and performance. They are found in both hardware and virtual solutions.

Load balancing configurations can vary between organizations. Some will operate by spreading pre-defined groups of users over specified servers. Others, more commonly, will split users across servers as they visit the system. The latter approach provides more flexibility and more effective load balancing in a dynamic environment, whereas the first approach is used more often on intranets.

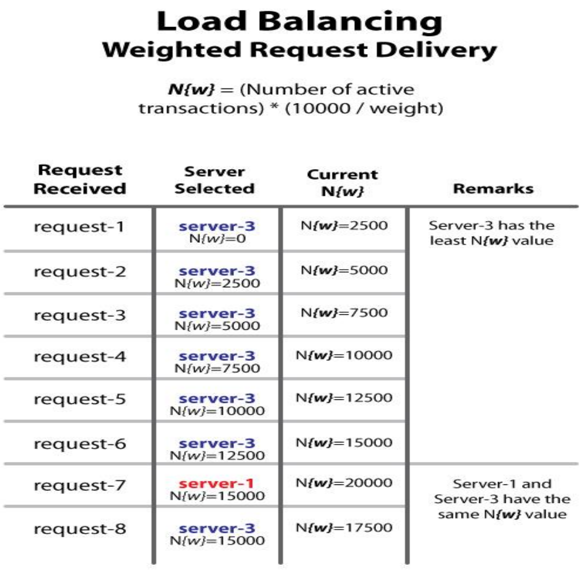
Splitting the users across multiple servers can be accomplished with a load balancing hardware solution or a virtual load balancing appliance. If a physical piece of hardware is used for load balancing, the load balancer will route users to different local physical servers. This is sometimes known as "direct server return", as the information from the servers goes directly to users instead of returning through the load balancing device. There are other methods of physical load balancing as well - including tunneling and IP address translation.

3.3 Classifications of load balancing

In general, load balancing algorithms follow two major classifications: a) Depending on how the services are distributed and how processes are allocated to nodes or servers (the system load);b)Depending on the information status of the nodes (System Topology) .In the first case it designed as centralized approach, distributed approach or hybrid approach and in the second case as static approach, dynamic or adaptive approach.

3.4 Selecting a load balancing method

Several different load balancing methods are available to choose from. If you are working with servers that differ significantly in processing speed and memory, you might want to use a method such as Ratio or Weighted Least Connections.



Load Balancing
Weighted Request Delivery

N{w} = (Number of active transactions) * (10000 / weight)

Request Received	Server Selected	Current N{w}	Remarks
request-1	server-3 N{w}=0	N{w}=2500	Server-3 has the least N{w} value
request-2	server-3 N{w}=2500	N{w}=5000	
request-3	server-3 N{w}=5000	N{w}=7500	
request-4	server-3 N{w}=7500	N{w}=10000	
request-5	server-3 N{w}=10000	N{w}=12500	
request-6	server-3 N{w}=12500	N{w}=15000	
request-7	server-1 N{w}=15000	N{w}=20000	Server-1 and Server-3 have the same N{w} value
request-8	server-3 N{w}=15000	N{w}=17500	

3.5 Analysis of load balancing algorithms

In order to balance the requests of the resources it is important to recognize a few major goals of load balancing algorithms:

a) Cost effectiveness: primary aim is to achieve an overall improvement in system performance at a reasonable cost.

b) Scalability and flexibility: the distributed system in which the algorithm is implemented may change in size or topology. So the algorithm must be scalable and flexible enough to allow such changes to be handled easily.

c) Priority: prioritization of the resources or jobs need to be done on beforehand through the algorithm itself for better service to the important or high prioritized jobs in spite of equal service provision for all the jobs regardless of their origin.
Following load balancing algorithms are currently prevalent in clouds:-

Round Robin: In this algorithm the processes are divided between all processors. Each process is assigned to the processor in a round robin order. The process allocation order is maintained locally independent of the allocations from remote processors. Though the work load distributions between processors are equal but the job processing time for different processes are not same. So at any point of time some nodes may be heavily loaded and others remain idle. This algorithm is mostly used in web servers where http requests are of similar nature and distributed equally.

Least 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.

Randomized: 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. Randomized algorithm does not maintain deterministic approach. It works well when Round Robin algorithm generates overhead for process queue.

Equally Spread Current Execution Algorithm: Equally spread current execution algorithm process handle with priorities. it distribute the load randomly by checking the size and transfer the load to that virtual machine which is lightly loaded or handle that task easy and take less time , and give maximize throughput. It is spread spectrum technique in which the load balancer spread the load of the job in hand into multiple virtual machines.

Throttled Load Balancing Algorithm: Throttled algorithm is completely based on virtual machine. In this client first requesting the load balancer to check the right virtual machine which access that load easily and perform the operations which is given by the client or user. In this algorithm the client first requests the load balancer to find a suitable Virtual Machine to perform the required operation.

A Task Scheduling Algorithm Based on Load Balancing: It is 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.

Biased Random Sampling: This algorithm uses 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. Here a virtual graph is constructed, with the connectivity of each node (a server is treated as a node) representing the load on the server. Each server is symbolized as a node in the graph, with each in degree directed to the free resources of the server. The load balancing scheme used here is fully decentralized, thus making it apt for large network systems like that in a cloud. The performance is degraded with an increase in population diversity.

Min-Min Algorithm: It begins with a set of all unassigned tasks. First of all, minimum completion time for all tasks is found. Then among these minimum times the minimum value is selected which is the minimum time among all the tasks on any resources. Then according to that minimum time, the task is scheduled on the corresponding machine. Then the execution time for all other tasks is updated on that machine by adding the execution time of the assigned task to the execution times of other tasks on that machine and assigned task is removed from the list of the tasks that are to be assigned to the machines. Then again the same procedure is followed until all the tasks are assigned on the resources. But this approach has a major drawback that it can lead to starvation.

Max-Min Algorithm: Max-Min is almost same as the min-min algorithm except the following: after finding out minimum execution times, the maximum value is selected which is the maximum time among all the tasks on any resources. Then according to that maximum time, the task is scheduled on the corresponding machine. Then the execution time for all other tasks is updated on that machine by adding the execution time of the assigned task to the execution times of other tasks on that machine and assigned task is removed from the list of the tasks that are to be assigned to the machines.

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 agents 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 neighbor's working load. To make their decision where to pass the token they actually build their own knowledge base. This knowledge base is actually derived from the previously received tokens. So in this approach no communication overhead is generated.

IV. PROPOSED ALGORITHMS

1. Efficient Load Balancing Using Least Utilized Server

Load balancing should take place when the load situation has changed. There are some particular activities which change the load configuration in Cloud environment.

The activities can be categorized as following:

- Arrival of any new job and queuing of that job to any particular node.
- Completion of execution of any job.
- Arrival of any new resource
- Withdrawal of any existing resource.

Whenever any of these four activities happens activity is communicated to master node then load information is collected and load balancing condition is checked. If load balancing condition is fulfilled then actual load balancing activity is performed.

Following is the proposed algorithm for Load Balancing:

```
Loop
 wait for load change
// depends on happening of any of four defined activities  if (activity_happens ())
 If (LoadBalancing_start ())
 while HeavilyLoaded_list is not empty
 Determine tasks which can be migratable using criteria of CPU consumed by
each job which has least CPU consumption selected for being migrated.
 Selected job = j;
 If  LightlyLoaded_list is empty
 PendingJob_list = PendingJob_list + j;
 Else
```

```
Migrate (LightlyLoaded_list [first], HeavilyLoaded_list[n], j);
//update the database
    End while
End Loop
```

Following are some functions used in the above algorithm:

Activity_happens (): this function return Boolean value. If any of above defined activity occurs it returns true otherwise it returns false.

LoadBalancing_start (): this function also return Boolean value. If on the basis of given parameters (CPU utilization and queue length) load balancing will be required it will return true else it will return false. This function also updates two lists: HeavilyLoaded_list and LightlyLoaded_list

Threshold heavy load and threshold light load is defined initially which depends on the traffic of application on the Cloud.

Function: LoadBalancing_start

```
Return Type: Boolean
 Start:
If (Standard Deviation of Load of nodes <  SD_Threshold)
If (Load of any node is greater then average Load value of nodes)
 HeavilyLoaded_list= HeavilyLoaded_list + l (new selected node);
 End if
 Else (Load of any node is greater then threshold   heavy load value)
 HeavilyLoaded_list= HeavilyLoaded_list + l (new
 selected node);
 Else if (Load of any node is less then threshold
 light load value)
 End
```

Here actual load distribution is performed at a centralized controller or manager node. The central controller polls each workstation and collects state information consisting of a node's current load as well as the number of jobs in the node's queue. The polling is done on basis of occurrence of some defined activity. It is not done periodically. Periodic checking approach is used in Condor. In case of periodic approach Load Balancer collects Load sample periodically which is not required and infect creates an overhead.

2. Dynamic Load Balancing Using– Hash Method

When a Load balancer is configured to use the hash method, it computes a hash value then sends the request to the server.

Hash load balancing is similar to persistence based load balancing, ensuring that connections within existing user sessions are consistently routed to the same back-end servers even when the list of available servers is modified during the user's session.

The hash value is computed as follows:

1. The load balancer computes two hash values using:
   a. The back-end server IP Address and Port (X).
   b. One of the incoming URL, Domain name, Destination IP, Source IP, Source & Destination IP, Source IP & Source Port, Call ID, Token (Y).

2. The load balancer computes a new hash value (Z) based on (X) and (Y).

3. The hash value (Z) is stored in cache.

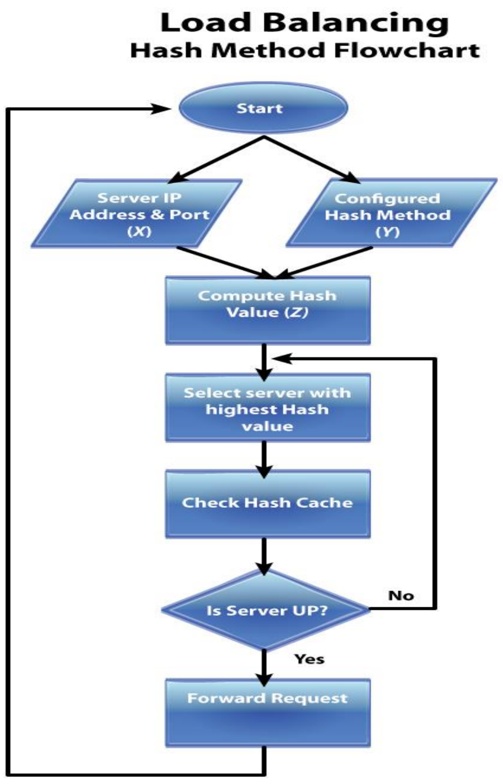
The load balancer forwards the request to the server with highest hash value, by using the value (Z) from the computed hash values. Subsequent requests with the same hash value (cached) are sent to the same server.

The following example shows how a Load Balancer works using the hash method.
The load balancer delivers the request based on the value of Hash (Z) as follows:

- Server-1 receives the first request.

- If server-1 is down, the hash value is calculated again.

- The load balancer selects the server with the highest hash value, and forwards the request.



V. CONCLUSION

This paper is based on cloud computing technology which has a very vast potential and is still unexplored. The capabilities of cloud computing are endless. Cloud computing provides everything to the user as a service which includes platform as a service, application as a service, infrastructure as a service.

One of the major issues of cloud computing is load balancing because overloading of a system may lead to poor performance which can make the technology unsuccessful. So there is always a requirement of efficient load balancing algorithm for efficient utilization of resources. Our paper analyzes on the various load balancing algorithms and their applicability in cloud computing environment. Then we proposed two algorithms namely the efficient load balancing using least utilized server and dynamic load balancing using Hash method to improve the performance of the cloud. In future we have planned to work on deploying Load Balancing as a Service (LBaaS) model.

VI. ACKNOWLEDGEMENTS

We are thankful to the Staff of PG Department of Computer Science, of Sacred Heart College College, Tirupatur for their support and encouragement.

BIBLIOGRAPHY

[1]. N. Ajith Singh, M. Hemalatha, "An approach on semi distributed load balancing algorithm for cloud computing systems" International Journal of Computer Applications Vol-56 No.12 2012.

[2]. Nitika, Shaveta, Gaurav Raj, International Journal of advanced research in computer engineering and technology Vol-1 issue-3 May-2012.

[3].Zenon Chaczko, Venkatesh Mahadevan, Shahrzad Aslanazadeh, and Christopher, IPCSIT Vol-14, IACSIT Press Singapore 2011.

[4].T. Kokilavani, Dr. D. I. George Amalarethinam "Load Balanced Min-Min Algorithm for Static Meta Task Scheduling in Grid computing" International Journal of Computer Applications Vol-20 No.2, 2011.

[5].Graham Ritchie, John Levine, ""A fast effective local search for scheduling independent jobs in heterogeneous computing environments" Center for Intelligent Systems and their applications School of Informatics University of Edinburg.

[6]. Shu Ching Wang, Kuo-Qin Yan Wen-pin Liao, and Shun Sheng Wang Chaoyang University of Technology, Taiwan R.O.C.

[7].Che-Lun Hung, Hsiao-hsi Wang, and Yu- ChenHu "Efficient Load balancing Algorithm for cloud computing network".

[8]. Yatendra sahu, M. K. Pateriya "Cloud Computing Overview and load balancing algorithms", Internal Journal of Computer Application Vol-65 No.24, 2013.

[9]. Nayandeep Sran, Navdeep kaur "Comparative Analysis of Existing Load balancing techniques in cloud computing", International Journal of Engineering Science Invention, Vol-2 Issue-1 2013.

[10].Nidhi Jain Kansal, Inderveer Chana "Existing Load balancing Techniques in cloud computing: A systematic review" Journal of Information system and communication Vol-3 Issue-1 2012

[11]. http://blogs.citrix.com/2010/09/04/load-balancing-hash-method/.

[12].Armstrong, R., Hensgen, D., Kidd, T.: The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions. In: 7th IEEE Heterogeneous Computing Workshop, pp. 79—87, (1998).

[13].Freund, R., Gherrity, M., Ambrosius, S., Campbell, M., Halderman, M., Hensgen, D.,Keith, E., Kidd, T., Kussow, M., Lima, J., Mirabile, F., Moore, L., Rust, B., Siegel, H.: Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet. In: 7th IEEE Heterogeneous Computing Workshop, pp. 184—199, (1998).

[14]. Freund, R. F., Siegel, H. J. : Heterogeneous processing. IEEE Computer, vol. 26, pp.13—17, (1993).

[15]. Ritchie, G., Levine, J.: A Fast, Effective Local Search for Scheduling Independent Jobs in Heterogeneous Computing Environments. Journal of Computer Applications, vol. 25, pp.1190—1192, (2005).

[16].Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L. L., Maheswaran, M., Reuther, A. I.,Robertson, J. P., Theys, M. D., Yao, B., Hensgen, D., Freund, R. F.: A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing, vol. 61,pp. 810—837, (2001).

[17].Wang, S. C., Yan, K. Q., Liao, W. P., Wang, S. S.: Towards a Load Balancing in a threelevel cloud computing network. In: Computer Science and Information Technology, pp. 108—113, (2010).