ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

A Systematic study on Load balancing in Distributed Computing

Immidisetty Deepika, Badam Srujana, Salina Adinarayana

Student, Dept of IT, Shri Vishnu Engg. College for Women, Bhimavaram, AP, India

Student, Dept of IT, Shri Vishnu Engg. College for Women, Bhimavaram, AP, India

Associate Prof, Dept of IT, Shri Vishnu Engg. College for Women, Bhimavaram, AP, India

ABSTRACT: Load balancing is a technique used to distribute the work load on single server, distributed severs and cloud etc. It is used to provide reliability, high availability and scalability services to the end users. In this we have studied different areas of load balancing and made a comparative analysis. We have also presented performance measures for load balancing with a case study approach.

KEYWORDS: Load Balancing, Infrastructures, Distributed Systems, Clustering, Cloud Load Balancing, Map Reduce, Performance.

I. INTRODUCTION

Load balancing [1] improves the distribution of workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units, or disk drives. Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any single resource. Using multiple components with load balancing instead of a single component may increase reliability and availability through redundancy. Load balancing can be implemented with hardware, software, or a combination of both. Typically, load balancing is the main reason for computer server clustering. Rest of the paper is organized as section 2 discussing the importance of load balancing. Section 3 describes the architecture details. Section 4 discusses load balancing in Distributed environment. Section 5 explores the load balancing in cluster computing. Section 6 explores the load balancing in cloud environment. Load balancing in Hadoop environment is discussed in Section 7. case study on load balancing in Amazon cloud is discussed in section 8, finally the performance measures were discussed in section 9.

II. WHY LOAD BALANCING

Load balancing is a key component of highly-available infrastructures [2] commonly used to improve the performance and reliability of web sites, applications, databases and other services by distributing the workload across multiple servers.

For example in a web server, the user connects directly to the web server, at yourdomain.com. If this single web server goes down, the user will no longer be able to access the website. In addition, if many users try to access the server simultaneously and it is unable to handle the load, they may experience slow load times or may be unable to connect at all.

This single point of failure can be mitigated by introducing a load balancer and at least one additional web server on the backend. Typically, all of the backend servers will supply identical content so that users receive consistent content regardless of which server responds.

High availability: In this scenario, you've hit the point where your business can no longer tolerate the risk of housing everything in a single building or region. To protect the business, you typically either bring up a completely offline backup data center in the event of a disaster, or you run multiple active, always-on data centers.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

Data locality: Some companies need to run multiple data centers for jurisdictional or regulatory reasons. For example, for data privacy reasons you may need to keep European data within Europe, or for performance reasons you may want to make sure your customers in Asia get all their data from your data center in Asia.

To put a control point in front of your services: It gives you the ability to change how your service is implemented on the backend without exposing those changes to the people who consume your service on the frontend. That could be an external customer, an internal user, or even another service in the data center.

III. ARCHITECTURE

A load balancing technique makes sure that each system in the network has same amount of work at any instant of time. This means neither any of them is excessively over-loaded, nor under-utilized shown in fig-1.

The load balancer distributes data depending upon how busy each server or node is. In the absence of a load balancer, the client must wait while his process gets processed, which might be too tiring and demotivating for him.

Various information like jobs waiting in queue, CPU processing rate, job arrival rate etc. are exchanged between the processors during the load balancing process. Failure in the right application of load balancers can lead to serious consequences, data getting lost being one of them.

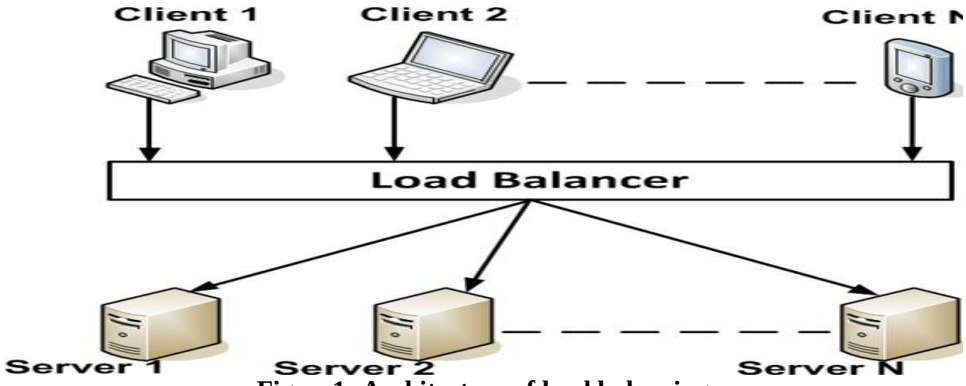

Figure1: Architecture of load balancing

Different companies may use different load balancers and multiple load balancing algorithms like static and dynamic load balancing.

It forwards client request to each connected server in turn. On reaching the end, the load balancer loops back and repeats the list again. The major benefit is its ease of implementation [3]. The load balancers check the system heartbeats during set time intervals to verify whether each node is performing well or not.

IV. LOAD BALANCING IN DISTRIBUTED SYSTEMS

Distributed systems offer the potential for sharing and aggregation of different resources such as computers, storage systems and other specialized devices. These resources are distributed and possibly owned by different agents or organization. The users of a distributed system have different goals, objectives and strategies and their behavior is difficult to characterize. In such systems the management of resources and applications is a very complex task.

The purpose of load balancing is to improve the performance of a distributed system through an appropriate distribution of the application load. A general formulation of this problem is as follows: given a large number of jobs, find the allocation of jobs to computers optimizing a given objective function .

One way to deal with the management of distributed systems [4] is to have an unique decision maker that will lead the system to its optimum. This is the approach used in most of the existing load balancing solutions.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

Load balancing in distributed system is of two types, they are

i. Static load balancing
ii. Dynamic load balancing

During the static load balancing(SLB) too much information about the system and jobs must be known before the execution. These information may not be available in advance. A thorough study on the system state and the jobs quite tedious approach in advance. So, dynamic load balancing algorithm(DLB) came into existence. The assignment of jobs is done at the runtime. In DLB jobs are reassigned at the runtime depending upon the situation that is the load will be transferred from heavily loaded nodes to the lightly loaded nodes. In this case communication over heads occur and becomes more when number of processors increase. In dynamic load balancing no decision is taken until the process gets execution. This strategy collects the information about the system state and about the job information. As more information is collected by an algorithm in a short time, potentially the algorithm can make better decision [10]. Dynamic load balancing is mostly considered in heterogeneous system because it consists of nodes with different speeds, different communication link speeds, different memory sizes, and variable external loads due to the multiple. The numbers of load balancing strategies have been developed and classified so far for getting the high performance of a system. Fig 2 shows a simple dynamic load balancing for transferring jobs from heavily loaded to the lightly loaded nodes.

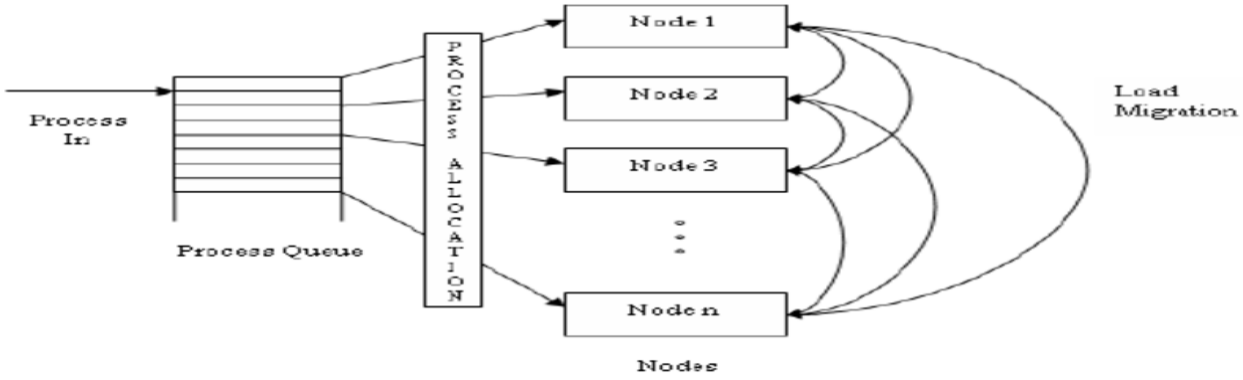

Figure2:Dynamic load balancing for transferring jobs from heavily loaded to lightly loaded nodes

V. LOAD BALANCING IN CLUSTER COMPUTING

A cluster is a group of resources that are trying to achieve a common objective, and are aware of one another. Clustering usually involves setting up the resources to exchange details on a particular channel (port) and keep exchanging their states, so a resource's state is replicated at other places as well. It usually also includes load balancing, wherein, the request is routed to one of the resources in the cluster as per the load balancing policy.

Load balancing can also happen without clustering when we have multiple independent servers that have same setup, but other than that, are unaware of each other. Then, we can use a load balancer to forward requests to either one server or other, but one server does not use the other server's resources. Also, one resource does not share its state with other resources.

Clustering [5] saves the user's state, and is more transparent to the user, but is harder to setup, and is very resource specific. Load balancing is comparatively more painless, and relatively more independent of application servers.

VI. LOAD BALANCING IN CLOUD COMPUTING

A website or a web-application can be accessed by a plenty of users at any point of time. It becomes difficult for a web application to manage all these user requests at one time. It may even result in system breakdowns. For a website owner, whose entire work is dependent on his portal, the sinking feeling of website being down or not accessible also brings lost potential customers.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

Cloud Load balancing [6] is the process of distributing workloads and computing resources across one or more servers. The workload is segregated among two or more servers, hard drives, network interfaces or other computing resources, enabling better resource utilization and system response time. Thus, for a high traffic website, effective use of cloud load balancing can ensure business continuity. The common objectives of using load balancers are:

i. To maintain system firmness
ii. To improve system performance
iii. To protect against system failures

Cloud providers like Amazon Web Services (AWS), Microsoft Azure and Google offer cloud load balancing to facilitate easy distribution of workloads. For ex: AWS offers Elastic Load balancing (ELB) technology to distribute traffic among EC2 instances. Most of the AWS powered applications have ELBs installed as key architectural component.

a)High Performing applications
Cloud load balancing techniques, unlike their traditional on-premise counterparts, are less expensive and simple to implement. Enterprises can make their client applications work faster and deliver better performances, that too at potentially lower costs.
b) Increased scalability
Cloud balancing takes help of cloud's scalability and agility to maintain website traffic. By using efficient load balancers, you can easily match up the increased user traffic and distribute it among various servers or network devices. It is especially important for ecommerce websites, who deals with thousands of website visitors every second. During sale or other promotional offers they need such effective load balancers to distribute workloads.

c) Ability to handle sudden traffic spikes
A normally running University site can completely go down during any result declaration. This is because too many requests can arrive at the same time. If they are using cloud load balancers, they do not need to worry about such traffic surges. No matter how large the request is, it can be wisely distributed among different servers for generating maximum results in less response time.
d) Business continuity with complete flexibility
The basic objective of using a load balancer is to save or protect a website from sudden outages. When the workload is distributed among various servers or network units, even if one node fails the burden can be shifted to another active node. Thus, with increased redundancy, scalability and other features load balancing easily handles website or application traffic.

VII. LOAD BALANCING IN HADOOP

Hadoop parturitions a job into several tasks and lazily assigns these tasks to available task slots in the cluster. Load balancing issues occur if there are some tasks significantly larger than others such that in the end only a few tasks are running while all others are finished. This situation happens in case of skewed reduce keys and can be easily identified .The Map Reduce[7]-algorithm and especially Apache Hadoop are powerful and cost-effective approaches to process large amounts of data in a distributed way. Hadoop offers the possibility to handle unstructured data and the possibility to provide stability throughout different platforms and different instances.
In Hadoop the load of work is distributed throughout a cluster of instances. Although Hadoop is very powerful, the efficiency stands and falls with the fact, that the load of work is distributed equally in order to provide the best throughput. Hadoop process the jobs by distributing the load among different nodes in a clustered environment where in which each node is implementing a specific task shown in fig3 . Modern Map Reduce implementations like Hadoop offer the user a large variety of possibilities to implement user-defined functions. Because of the many possibilities, many situations may occur where the uneven distribution of the data may lead to a longer execution time . One challenge is to evenly distribute the workload among the clusters. There are many examples of scientific data, that is skewed [6] and balancing the load may become necessary.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

It is assumed, that the processing time for each object is the same and the source of skew is only owed by the uneven distribution of the data. Even if it could be the goal of future projects to implement load-balancing for more general problems. Map-Phase The map-phase has the goal to build a key-value pair for each object in the split of the input data, that was assigned to this mapper. This phase consists of three sub-tasks: The acquisition of a split of the input data, the actual mapping and in the end the preparation of the map output files for further processing. Between the last two sub-tasks, the use of the partitioning function is explained. This step is necessary to explain the building of the

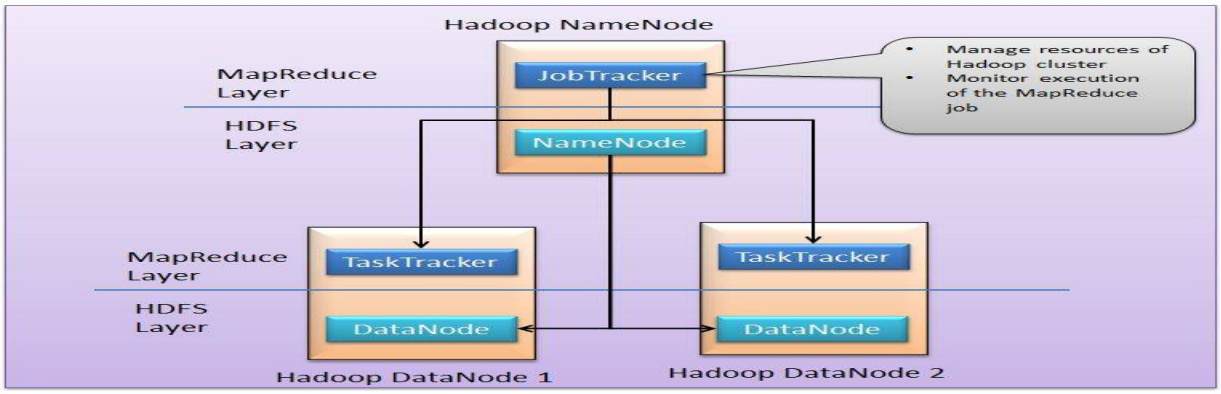

Figure3:The Execution of a Hadoop Job

Map output files. Acquisition of Data The input of the Hadoop-Job can consist of more than one file. A Java Virtual Machine is started to perform either a map- or a reduce-task.

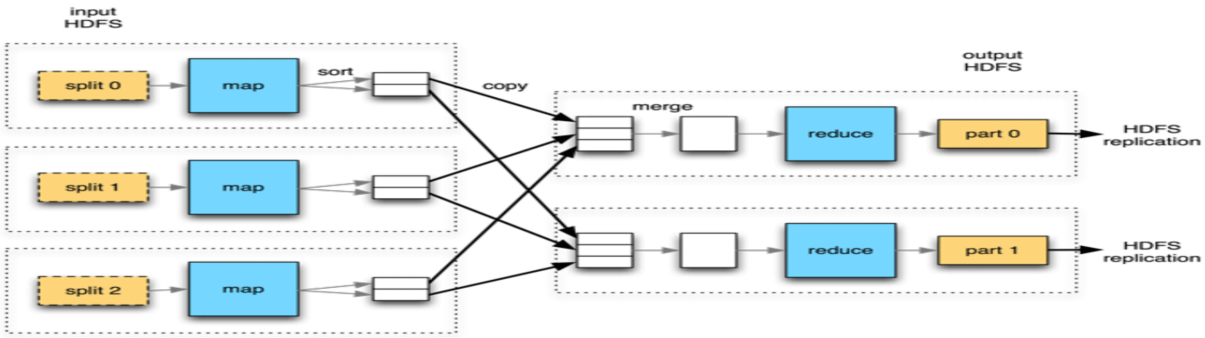

Figure 4: Overview of the Map–Reduce Phase

To map the input data, obviously a map-task is initialized. The number of map-tasks started during a Hadoop-Job depends on the amount of input data. The number of map-tasks is driven by the pre defined block size of the input. Up to a certain point, the number of mappers can be manipulated manually. On the one hand, this block size can be altered; on the other hand the number of map-tasks can be increased.

To illustrate how a map function is working and is building the key-value pairs, the map-function that is needed to perform the example presented is applied to the data. The key-value pairs are formed out of the input data for M1, where the key in this example is the year, and the value of every object is 1. The number of timestamps for each year can be evaluated. Table-1 shows the output key-value pairs for M1.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

Table1: Shows the output key-value pairs 1

Time stamp	Key	Value	Key-Value Pair
1993-04-10	1993	1	<1993,1>
1994-02-04	1994	1	<1994,1>
1996-01-30	1996	1	<1996,1>
1996-09-12	1996	1	<1996,1>
1992-01-01	1992	1	<1992,1>
1997-03-02	1997	1	<1997,1>
1994-11-24	1994	1	<1994,1>
1993-07-13	1993	1	<1993,1>

A partitioner, respectively the partitioning function, has the duty to divide the key-value pairs into partitions. It decides, which key-value pair is processed by which reducer. As it is explained the user has the possibility to manually define a partitioner in the Hadoop Program or to use the default Hash-Partitioner of Hadoop. The decision, which partitioning function should be used, depends on the nature of the problem Hadoop should manage. A partitioning function in Hadoop will produce exactly the same number of partitions as there are going to be reducing tasks. The choice of an appropriate partitioner can be very decisive and game changing. If, for example, a partitioner by accident sends all the key-value to one reducer, the whole data set would be processed by one reducer. While the others would be without work. The partitioning function is therefore responsible for the load distribution throughout the cluster. The partitioning function can be individually defined in the Hadoop Program. In contrast to map and reduce functions, Hadoop offers some standard partitioning function, which in many cases satisfies the need of the users. The default partitioning function uses a hash value generated out of the key of the key-value pair. The final number of the partition is this hash value modulo the number of reduce-tasks of the job.

Reduce-Phase the reduce-phase is the second main part of a Map Reduce-algorithm. The goal of the reduce phase is to evaluate all the key-value pairs with the same key and summarize these pairs to a output object. This phase consists of three sub-tasks: The acquisition of the necessary map output files, the consolidation of the key-value pairs with the same key and the actual reducing.
Similar to the execution of a map task, the Task Tracker starts a Java Virtual Machine, which then performs the reducing. A short overview of the three phases is shown in Fig-4.
Acquisition of the Map Output and Shuffling As it has been discussed in, the map output files for all the reduce tasks are built. To perform the reduce-step, it is necessary to acquire all the files for a specific reducer. This phase, the acquisition, is called the shuffle-phase.

Sorting the next phase after shuffling is the sorting-phase. During this phase, all the map output files have been acquired, and now have to be merged, sorted and converted to a proper format in order for the reduce function to iterate over the data. The data is not yet in the form the reducer requires it. Until now the data consists only of key-value pairs. The input format for the reducer is of the form <key, list of values>. Therefore the data has to be transformed. The task of transformation and merging is done in a separate class called Merger. The Merger-class merges the same keys together across all acquired map output files and constructs the requested format for the reduce function. During this merging process, the keys are sorted in alphabetic ascending order.

Reduce Input:
Once the merging and sorting is completed, the keys with the lists of values are in adequate form to be processed by the reduce function. The maps outputs files, which were collected during the shuffling-phase, are now not longer required and are deleted.
Reduce-Function to apply the reduce function on the data is the main task. All the phases before are for the preparation of the data in the right form to iterate over all the keys with the reduce function. As, all the data from all the mappers are in the form <key, list of values>. The reduce function is a while-loop, which iterates over all the keys. The actual reduce function is provided by the Hadoop Program and is stored in the Job Configuration. This actual reduce function, which is provided by a user (there is node fault implementation for the reduce function), has a for-loop. This loop

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

typically iterates over all the entries in the list of values. To illustrate the functionality, the evaluation of a year will be shown. Shows the inputs for the Reducer 1 (hash-value 0). This is the specific reduce function:

```
Public void reduce (Text key, Iterator<Text> values, Output Collector<Text, Text> output, Reporter reporter)
{
  int sum = 0;
  for (IntWritable val : values)
 {     sum  += val.get();    }
output.write (key, new IntWritable (sum));
}
```

This specific reduce function adds the values together, and gives in the end the number of occurrences of the specific key. The final output of the reduce functions OutputCollector can be seen in table-2.

Table2: Output of the Reduce Function 1

Key	Value
1992	2
1995	2
1998	2

The data has now been processed by the reduce function and is ready to be used as output of the job. Parallel to the reduce function of the reducer, the output file of Hadoop is written. The object OutputCollector directly writes the key-value pair of the reducer to the file. It is important to know, that each reducer produces an individual output file. By default the output files are not merged.

VIII. AMAZON CLOUD CASE STUDY

INITIAL SITUATION

The corrections are performed by multiple instances of the Duden Enterprise Servers that run at the Amazon cloud. So far load balancing occurred at the level of the HTTP protocol, which distributed the load onto individual nodes. A scaling of the server farm was responsible for the launch of additional nodes during heavy load or for the shutdown of nodes in times of decreasing usage. The existing load balancing did not support sessions based on Web Services. However, sticky sessions using the SOAP protocol were necessary to run more applications at the server farm.
Requirements to run all applications within the cloud a load balancing solution was needed that would meet the following requirements:

i. All requests of a session must be forwarded to the same server node (sticky sessions).
ii. Depending on the load, the list of servers has to be extended or shortened.
iii. The ability to work within the Amazon Cloud.
iv. The balancing should scale up to hundreds of thousands requests per hour.
v. Extraction of the session id out of an XML element inside the SOAP body.
vi. SSL encryption.

Membrane Service Proxy

When searching for a suitable load balancer among others the Membrane Service Proxy was investigated. At that time Membrane did not support SOAP sessions, but Membrane was easily extensible by Java and the source code was freely available.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

The decision was made in favor of an extension for SOAP-based sessions for Membrane Service Proxy. To implement the extension an open source sponsorship contract was signed with predic8, the manufacturer of Membrane Service Proxy. The implemented functions for Duden were published under the ASF open source license and thus are available to the general public. Duden benefits from bug fixes and the experience of other users. Since the extension was integrated into the Membrane distribution, Duden can upgrade to new Membrane versions without performing a migration project.
The extension consists of the following components:

i. An extractor that reads the session id from an XML element.
ii. A REST based interface for adding and removing nodes from the cluster.
iii. A Web frontend to manage clusters.

IX. PERFORMANCE M EASUREMENT

Load balancing facility improves the performance of the distributed system. Overall system performance can be measured by the following parameters [1, 3, and 5]:
A. Mean Response Time
Performance [8] of a load balancing algorithm can be measured by the response time. Response time is the time elapsed between start of execution of a process and its completion. To achieve the good response time, processes must be distributed evenly among the nodes using appropriate load
balancing technique. Good response time also means that the processes don't have to wait too much in the system.

Response time can be computed as follows:

$$RT = FT – ST.....(1)$$

Where,

RT is Response Time
FT is finish time of current CPU burst
ST is start time of current CPU burst
Note that a process arrives and executed several times on the central processing unit during its lifetime.
B. Processor Utilization
Utilization of processor means the percentage of time for which the node is busy in running processes. This index is useful at lower load conditions. At the higher load conditions, even after maximum utilization of the processor, some of the processes are waiting for execution. These
Processes can't be taken into account for measuring load.
C. Mean Slow Down
Sometimes response time or waiting time cannot give correct idea about the suffering of processes, particularly when there is huge variation in their processing times. Slowdown or penalty ratio can be used to measure proportionate suffering of processes in the system irrespective of whether the process is long or short. Slowdown of a process is defined as the ratio of total time spent by the process in the system to the execution time of the process and is defined as:

$$P= (t + w + m) / t \quad -------(2)$$

Where,
P is mean slowdown or penalty ratio
t is execution time
w is missed or waiting time of a process in queue
m is migration time

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

REFERENCES

1. Xia, Feng, et al. "Community-based event dissemination with optimal load balancing." IEEE Transactions on Computers 64.7 (2015): 1857-1869.
2. S.Ayyasamy and S.N. Sivanandam," A Cluster Based Replication Architecture for Load Balancing in Peer-to-Peer Content Distribution", International Journal of Computer Networks & Communications (IJCNC) Vol.2, No.5, September 2010
3. Michael Boyer and Kevin Skadron, Shuai Che and Nuwan Jayasena," Load Balancing in a Changing World: Dealing with Heterogeneity and Performance Variability" ACM 978-1-4503-2053-5.
4. Impervadatasheet,"Optimize Traffic Distribution and Application Delivery from the Cloud", https://www.incapsula.com/datasheets/load-balancing-failover.pdf
5. Safiriyu Eludiora , Olatunde Abiona , Ganiyu Aderounmu , Ayodeji Oluwatope , Clement Onime , Lawrence Kehinde," A Load Balancing Policy for Distributed Web Service", Int. J. Communications, Network and System Sciences, 2010, 3, 645-654 doi:10.4236/ijcns.2010.38087 Published Online August 2010.
6. Indresh Gangwar , Poonam Rana," Cloud Computing Overview: Services and Features", International Journal of Innovations & Advancement in Computer Science, ISSN 2347 – 8616 Volume 3, Issue 1 April 2014
7. Prabhjot Kaur and Dr. Pankaj Deep Kaur ," Efficient and Enhanced Load Balancing Algorithms in Cloud Computing", International Journal of Grid Distribution Computing Vol.8, No.2 (2015), pp.9-14 http://dx.doi.org/10.14257/ijgdc.2015.8.2.02
8. K.Yogeswara Rao , S.Adinarayana ," A Study on Tools of Big Data Analytics" , International Journal of Innovative Research in Computer and Communication Engineering , Vol. 4, Issue 10, October 2016, ISSN(Online): 2320-9801
9. Sagar Dhakal, Majeed M. Hayat, Jorge E. Pezoa, Cundong Yang, and David A. Bader," Dynamic Load Balancing in Distributed Systems in the Presence of Delays: A Regeneration-Theory Approach", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 18, NO. 4, APRIL 2007
10. Kanungo, Priyesh. "Measuring Performance of Dynamic Load Balancing Algorithms in Distributed Computing Applications." (2013).