# Load Rebalancing in Cloud Computing Environment

Anagha Meher
Department of Computer Engineering
St. Francis Institute of Technology
Mumbai, India

Bidisha Roy
Department of Computer Engineering
St. Francis Institute of Technology
Mumbai, India

Rajkumar Shende
Department of Computer Engineering
St. Francis Institute of Technology
Mumbai, India

***Abstract*—Cloud computing is distributed computing over a network and it means the ability to run a program on many connected computers at the same time.Large scale distributed systems such as cloud computing applications are becoming very common. These applications come with increasing challenges on how to transfer and where to store and compute data. Load balancing is the one of the challenging task. Load balancing is the process of reassigning the total loads to the individual nodes of the collective system to make the best response time and also good utilization of the resources and to remove the situation where some nodes are over loaded and some other are under loaded. To decrease the total number of heavy nodes (servers) in the system by moving load from heavy nodes (servers) to light nodes (servers) is the main aim of balancing the load. Our objective is to allocate the files as uniformly as possible among the nodes such that no node manages an excessive number of loads. In this paper we present K-means algorithm, Min-Min and Max-Min algorithm for load balancing on cloud.**

**Keywords—** ***Load balance, clouds, K-means algorithms, Min-Min algorithm, Max-Min algorithm***

## I. INTRODUCTION

The concept of Cloud computing has significantly changed the field of parallel and distributed computing systems today. Cloud Computing (or cloud for short) is a compelling technology. In clouds, clients can dynamically allocate their resources on-demand without sophisticated deployment and management of resources [1]."Cloud" simplifies the many network connections and computer systems involved in online services. Cloud Computing is a technology, which connects so many nodes together for allocating resources dynamically [2].Cloud computing is a internet based development and use of computer technology. It is a style of computing in which dynamically scalable and often virtualized resources are provided as a service over the internet. Different types of technologies are used in clouds such as Map Reduce programming paradigm, distributed file systems, virtualization. These kinds of technologies are scalable which can add or delete new nodes or systems making it reliable [2].In large cloud we can connect hundreds or thousands of node together. By shifting of workload (processes) among the processor (servers), it is a process of load balancing for improving the performance of the system. Load balancing is a methodology to distribute workload across multiple computers, or other resources over the network links to achieve optimal resource utilization, maximize throughput, minimum response time, and avoid overload. This project is based on simulation technique.

When analyzing the existing system, clouds rely on central nodes to balance the loads of storage nodes, there comes the performance bottleneck because the failure of central nodes leads to the failure of whole system and it will leads to many technical and functional difficulties.

The term which is generally used in reference to internet is called as cloud computing. The cloud is changing the worldwide network of computer into largest single computer. Resource sharing increases the load on single machine. Therefore overall performance decreases and this problem are called as load balancing. Load balancing is one of the major issues now days. It is a process of reassigning the total load to the individual nodes of the collective system to make resource utilization effective and to improve the response time of the job, simultaneously removing a condition in which some of the nodes are over loaded while some others are under loaded.

Load balancing is one of the central issue in cloud computing.

It is a mechanism that distributes the dynamic local 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.

Load balancing simultaneously removing a condition in which some of the nodes are over loaded while some others are under loaded.

## II. RELATED WORK

In [1], The chunks can be distributed to the system evenly for reducing movement cost as much as possible it is a design to reallocate file chunks in load rebalancing algorithm. Number of chunks migrated to balance the loads of the chunk servers it is define as movement cost. To reduce demanded movement cost and to balance the loads of nodes these are the advantages of this paper. Physical network locality and node heterogeneity these advantages are taken in this paper. Existing centralized approaches are comparable with this proposal. The load imbalance factor, movement cost, and algorithmic overhead these terms are considerably outperform in prior distributed algorithm.

In [5], Distributed hash tables are shown to become a useful building block for a variety of distributed applications. To achieve desired load balancing goals, implementation complexity and substantial storage overhead these two are required for current schemes based upon consistent hashing. Author argues in this paper that these goals can be achieved more simply. First, author suggests the direct application of the power of two choices paradigm, whereby an item is stored at the less loaded of two (or more) random alternatives. Then consider how associating a small constant number of hash values with a key can naturally be extended to support other load balancing methods, including load-stealing or load-shedding schemes, as well as providing natural fault tolerance mechanisms.

In [6][7], Authors use the concept of virtual servers for load balancing. A virtual server looks like a single peer to the underlying DHT(Distributed Hash Table), but for more than one virtual server, each physical node can be responsible. For example, in Chord [8], for a neighboring region of the identifier space, each virtual server is responsible but by having multiple virtual servers, a node can own no neighboring portions of the ring. We can move a virtual server from any node to any other node in the system it is a key advantage of splitting load into virtual machine. This paper presents three simple load-balancing schemes that differ primarily in the amount of information used to decide how to rearrange load. All these schemes try to balance the load by transferring virtual servers from heavily loaded nodes to lightly loaded nodes. The amount of information required to make transfer decision this is the key difference between these three schemes. First is one-to-one, second is one-to-many and third are many-to-many. The first scheme is based on a one-to-one assignation mechanism, where randomly picked the two nodes. If one of the nodes is heavily loaded and the other is light then virtual server transfer is initiated. Unlike the first scheme, second scheme allows more than one light node to a heavy node is to be considered. Third scheme is a logical expansion of the first two schemes. While in the first scheme we consider one heavy node to a light node and in the second scheme we consider one heavy node to many light nodes, in this scheme we consider many heavy nodes to many light nodes.

In [9], basic concepts of Load balancing and cloud computing are discussed. Some existing load balancing algorithms which can be applied to clouds are studied in this paper. Different load balancing strategies with reporting time for single level tree networks and the closed-form solutions for minimum measurement these additional points were studied in this paper. The performance of these strategies with respect to the timing and the effect of link and measurement speed were studied.

In [10], Authors discuss the Min-Min and Max-Min algorithm for load balancing. The main drawback of Min-Min algorithm is it delays the execution of the smaller jobs and because of the dynamic nature of the cloud, execution of the smaller jobs may be postponed indefinitely. Min-Min algorithm's disadvantages are overcome in Max Min algorithm which is static in nature.

One of the feature of the Max-min algorithm is it selects the largest job and is executed on the fastest available resource. In Max-Min algorithm, maximum execution time's job is selected and it is send it to the machine which has min completion time. In other word complicated jobs runs on idle machine. When the number of small tasks is more than number of the large tasks in a meta-task, the Max-min algorithm schedules tasks, in which the make span of the system relatively depends on how many, executing small tasks concurrently with large one[11]. The enhancement to this Max-min Algorithm is instead of selecting maximum execution time task, selects an Average or nearest greater than average task then overall makespan is reduced and also balance load across resources [12].

In [13], Jobs are equally distributed to all slave processors in Round Robin algorithm. According to the round robin algorithm, all jobs are given to the slave processors. It mean that it perform the processor selection in series and if the last processor has been reached then it will be back to the first processor. Selections of the processors are performed locally, independent of allocations of other processors. Inter process communication is not require in Round Robin algorithm this is the main advantage of this algorithm. In general Round Robin is not expected to achieve good performance in general case. However when the jobs are of unequal processing time this algorithm suffers as the some nodes can become severely loaded while others remain idle. Round Robin is generally used in web servers where generally HTTP requests are of similar nature and thereby be distributed equally.

In Randomized algorithms [13], random numbers are used to select slave processors. The slave processors are selecting randomly following random numbers generated based on a statistic distribution. Out of the load balancing algorithms for particular special purpose applications, randomized algorithm can obtain the best performance.

In Central Manager Algorithm [14], in each step, slave processor is selected from the central processor to be assigned a job. The processor having minimum load is as the slave processor which is selected. In this algorithm, selection is based on central manager algorithm and it is possible to perform because load information about slave processor is able to gather by central processor. Depending on the system load information, load balancing decision is made by load manager. When process created it will allow the best decision. Obstruction could occur because of high degree of inter-process communication. On the different hosts dynamic activities are created then performance of this algorithm is expected to be better than parallel applications.

## III. PROPOSED WORK

To decrease the total number of heavy nodes (servers) in the system by moving load from heavy nodes (servers) to light nodes (servers) is the main aim of balancing the load. Transferring the load from heavily loaded server to the lightly loaded server this equal distribution of the may improve utilization of resource. Proposed system is showing the simulation of load balancing on cloud. This system is proposed for balancing the load on private cloud.

In our proposed system, each module server first estimate whether it is lightly loaded or moderately loaded or heavily loaded based on the server's colour. All the over loaded servers in the system becomes under loaded server this process repeats. If there is a situation where all the servers are heavily loaded no other server is remaining for balancing the

load. In this situation some servers are keeping for this emergency situation. Server replica is created and it will balance load of the server. Server replicas are those servers which are kept in the side for emergency situation this is the concept behind rebalancing.
Detailed steps are given below.

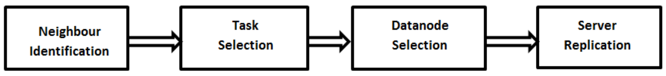


Fig 1. Block diagram of Proposed System

In proposed system, Load of the server is decided on the size of RAM and CPU cycle where how may processes are performed and type of that processes. The proposed system consists of the modules as shown in the Fig.1. The important phases as part of the proposed solution are described in brief as follows:

*Step 1: Neighbour Identification*

First we need to find neighbour of that sever which is heavily loaded where we can transfer the load. Load of the neighbour server is very important, while assigning processes (tasks) in distributed computing. This helps in reassigning tasks to the neighbour easily. Neighbour server should be lightly loaded. Neighbour identification is done with help of k nearest neighbour algorithm.

File is partitions into number chunks and different chunks are migrated to different chunk server for balancing the load. For balancing the load by using nearest neighbour algorithm migrate one whole process into any one server therefore no need to keep the track of the process which is transfer for load balancing.

*Step 2: Task Selection*

Once Nearest server is decided then task selection process is performed. Depending on the nearest server's capacity algorithm decides either transfer complicated processes, medium complicated processes or simple processes. If Nearest server has more capacity that time complicated processes will transfer and if server doesn't have that much capacity then simple and medium processes will transfer. Task selection is done with the help of Min-Min and Max-Min algorithm.

*Step 3: Datanode Selection*

Once the tasks (processes) are decided for transfer, the datanode selection for assigning the task forms the next important phase of the whole process. Here we have to select the datanode (server) which is lightly loaded. Because if we select the nearest server which is moderately loaded and we transfer the processes to that server, there is possibility it will become overloaded. Datanode selection is done with the help of Min-Min and Max-Min algorithm.

*Step 4: Server Replication*

Replication of the server is required for decreasing the load on the system (server). If the situation occurs, where all the servers are heavily loaded there is no lightly loaded server where we transfer the load. In this emergency situation replica of the server is automatically created where we transfer the load for balancing the server.

## IV. IMPLEMENTATION

The proposed framework is implemented using java with help of Net Beans IDE 8.0.2. Design Swing GUIs by dragging and positioning GUI components from a palette onto a canvas. Drag components from the palette and drop them onto the canvas. Server's properties are entered in edit properties window which is in right side of the canvas. Severs are created on canvas. It shows simulation of load balancing in private cloud.

*Steps for Create New Server*

Click on Start Adding Server button

When we click on Start adding server button this button will get disabled and Stop adding server button is enabled.

1. *Click Anywhere on Canvas*

Once we clicked on the canvas, on right hand side we can see edit properties window in that we have to enter following details (Each of the details by default values are given): Server name, RAM size(1TB),CPU capacity(MHz),Process count(it
Should be greater than 1)

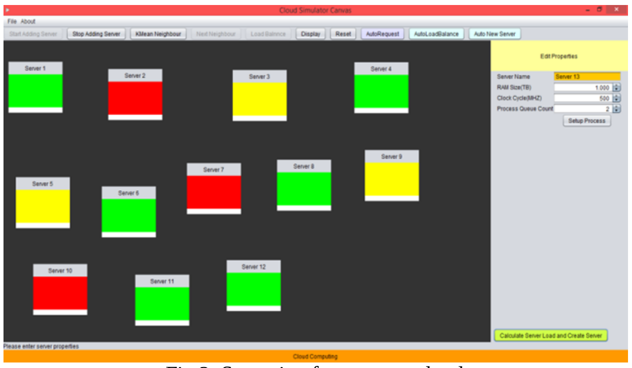
   a. Then Click on Setup process. When we clicked on that button, number of process has been displayed. Then Enter Process name (which are performed on that server) and select process Type. On server three processes have been performed (1) Simple process (2) Medium Complex process (3) Complex process.

2. Once information is entered, then finally Click on calculate server load and Create Server. For adding each server, we have to follow above steps.
3. We can recognized server's load, depending on server colour. If Server colour is Green, server is lightly loaded. If it is Yellow then it has medium load. But server colour is Red, it is heavily loaded.
4. Select server which has heavy load on the canvas, we need to balance load of that server .Once we select heavily loaded server, firstly we need to find the nearest neighbour of that server.
5. To find nearest neighbour server, we have to perform neighbour identification with the help of k-means algorithm.
6. Once server is selected click on K-mean neighbour button, so that it shows the nearest neighbour server which has less load.
7. If load is more than neighbour server's capacity, then click on next neighbor button to find a next neighbour which is lightly loaded.
8. Once nearest Neighbour found, Click Load Balance for transfer load to that neighbour. It will remove some Processes from selected server and add the same to Balancing Server.
9. Display button shows the details of server and process. Here we can see how much load transferred to the nearest server.
10. Reset button is used for reset all the information from the canvas.
11. AutoRequest button is used for showing the user's request (processes) which are come in the server. How

many users are active is shown. All this is done automatically.

12. If we click AutoLoadBalance button, it will automatically balance the load between the servers.
13. When automatically requests are come and balancing the load, if there is situation where all the servers are heavily loaded there is no other server for balancing the load then we click on the Auto new server button. Once we click this button automatically new server gets created (these are the servers which are kept for emergency situation).
14. Click Export from menu to export Server Grid to XML And Exported XML would be saved as ServerFile.xml in same.
15. Click Import from menu It will import ServerFile.XML from root folder and All Server from XML would be loaded in canvas.

## V. RESULTS



Server Name: Server 13
RAM Size(TB): 1.000
Clock Cycle(MHZ): 500
Process Queue Count: 2

Please enter server properties

Fig 2. Scenario of servers on cloud

Fig 2. Shows that Overall scenario of severs on cloud in which some servers are heavily loaded, some are moderately loaded and some are lightly loaded.

Load balancing for three different situations is explained below

Situation 1:

1. Server 1,Server 2 and Server 6 are lightly loaded
2. Server 3 and Server 5 are heavily loaded
3. Server 4 has medium load

This can be explain by figure 3(a) below

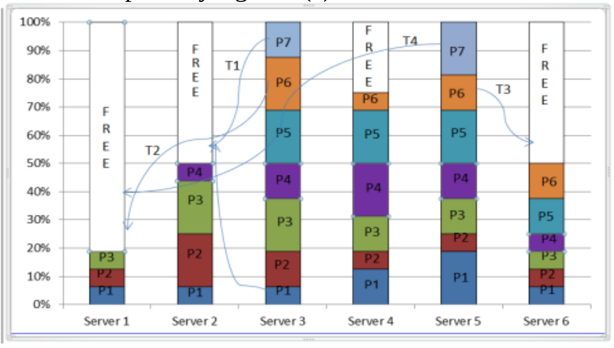


Fig 3(a). Situation 1

Solution:

1. Select Server 3 which is heavily loaded.
2. Click K means neighbour button to find nearest neighbour of server 3 by using k-means neighbour algorithm.
3. After that Server 2 is highlighted because it is neighbour of Server 3 which is lightly loaded.
4. Then click load balance button for transferring some server 3's processes.
5. Process p1 and p7 with the help of max-min algorithm which is shown with arrow T1 (transfer 1).
6. Then find next nearest neighbour which is Server 1.Server 3 transfer (which is shown with the direction T 2) process p6 to the Server 1.
7. Like this Server 5 transfer (T 3) process p6 to Server 6(nearest neighbour) and then find next nearest neighbour which is lightly loaded.
8. Server 1 is next possible neighbour which is lightly loaded .Server 5 transfer (T 4) process p7 to the Server 1.
Note: Number of T's is used to show which process is transfer first from Server.
This can be explain by figure 3(b) below

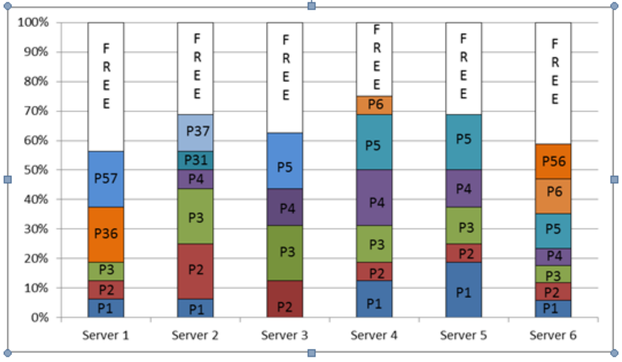


Fig 3(b). Load are balance in all the severs

Situation 2:

1. Here, all servers are heavily loaded so we can't transfer the load to the nearest neighbor.

This can be explain by figure 4(a) below

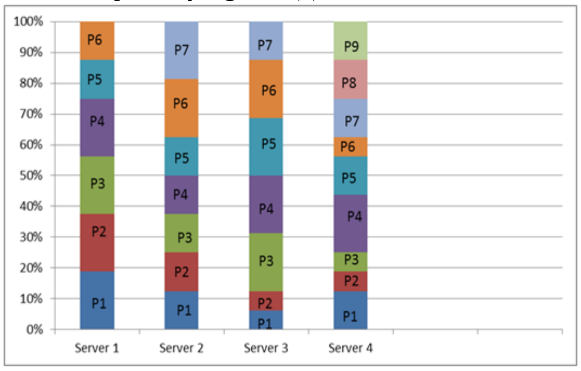


Fig 4(a). All servers are heavily loaded

Solution:

1. Some servers are preserved for emergency situation like this
2. If we click auto request and auto load balance, automatically tasks are transfer for balancing the server.
3. In this situation all the available servers get heavily loaded then there is option as auto new server when we clicked on that button automatically new servers are generated , with this server we can perform load balancing process And balance the load from heavily loaded server to lightly loaded server.

4. Select Server 4 which is heavily loaded and click k-means neighbour button to find nearest neighbour which is Server 5.
5. Server 4's process p7, p8, p9 are transfer (T1) to the Server 7.Then select Server 3 which is heavily loaded. Then click K-means neighbour button for finding nearest neighbour of this server
6. Server 3's nearest neighbour is server 5 but if we transfer tasks to server 5, server 5's load get increase so for balancing the load we find next neighbour by clicking next neighbour button.
7. Next neighbour is sever 6.server 3' process p1, p2, p6 transfer (T2) to the server 6 on the bases of max-min algorithm.
8. Then select server 2(heavily loaded) and click k-means neighbour and then click next neighbour button to avoid imbalance situation.
9. Next neighbour is server 7.server2's process p5, p6, p7 is transfer (T3) to server 7.
10. Then select server 1 which is heavily loaded. Server 1's process p6 is transfer (T4) to the server 5 on the bases of max-min algorithm.
11. Again select server 1 for balancing the server. Server 1's nearest neighbour is server 6 with the help of k-means and next neighbour.
12. Server 1's process p4, p5 are transfer (T5) the server 6.

This can be explain by figure 4(b,c) below

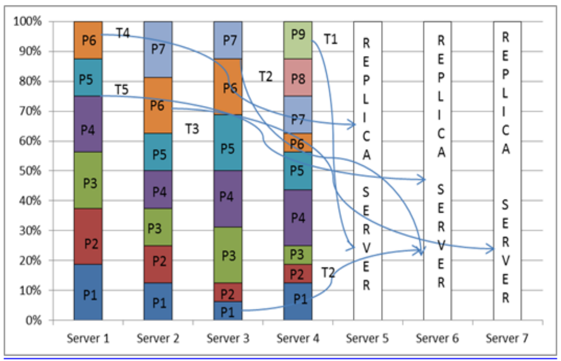


Fig 4(b) Replication servers are created automatically

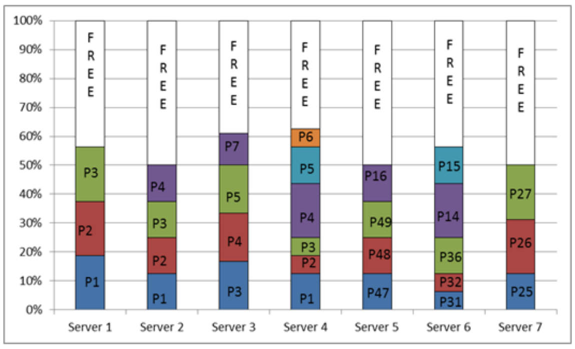


Fig 4(c) Load are balance between all the servers

Situation 3:

1. Server no 2 and 3 are heavily loaded
2. First we select server 2 for load balancing, for that we have to find nearest neighbour but its nearest neighbour is server 3 which is also heavily loaded(red colour).

This can be explain by figure 5(a) below

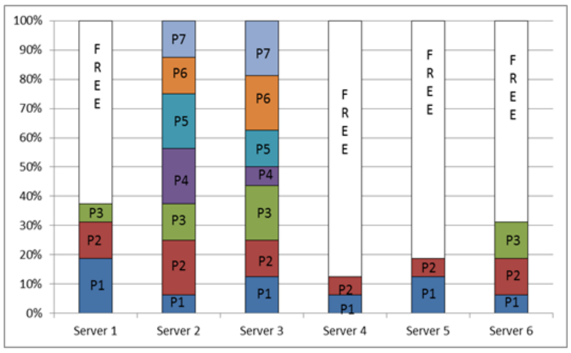


Fig 5(a). Some servers are heavily loaded and others are lightly loaded

Solution:

1. Select server 2 which is heavily loaded. click k-means neighbour button.
2. Server 2's nearest neighbour is server 3. In this situation one message box is displayed that server 3 is loaded please select possible server and then automatically next nearest neighbour which is lightly loaded gets highlighted.
3. Server 4 gets highlighted. Server 2's Processes p5, p6, p7 transfer (T1) to the server 4 with the help of max-min algorithm.
4. Then Select Server 3 which is heavily loaded. Click k-means neighbour button and again click next neighbour for finding lightly loaded neighbour.
5. Server 3's nearest neighbour is sever 5. Server 3's processes p6, p7 are transfer (T2) to the server 5.

This can be explain by figure 5(b,c) below

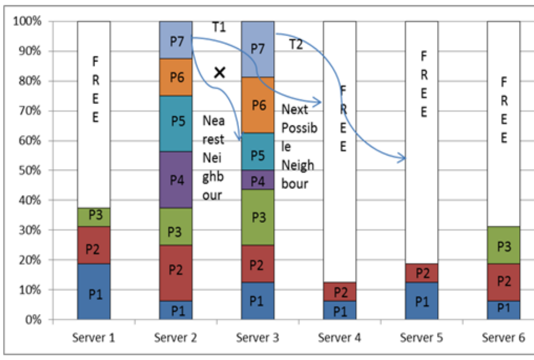


Fig 5(b)Nearest neighbour is also heavily loaded so can't transfer to it

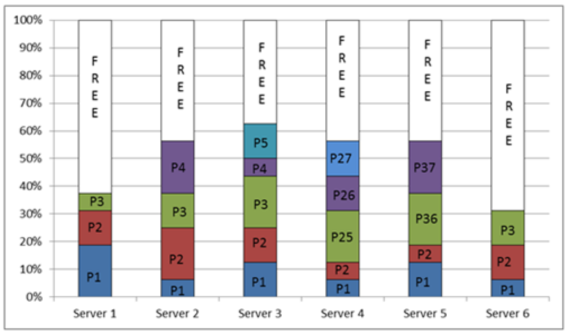
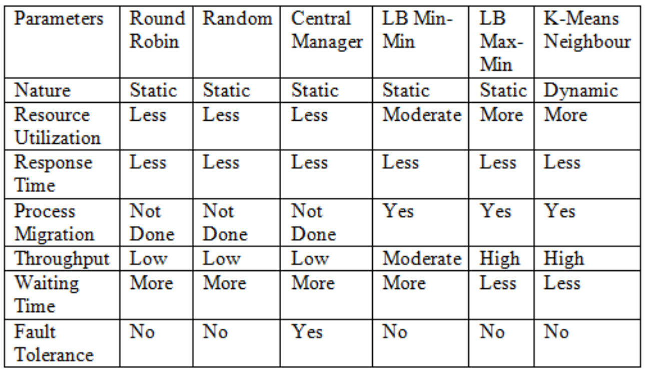


Fig 5(c). Balancing the load between all the servers

## VI. PERFORMANCE ANALYSIS

Table I. Comparative study on load balancing algorithms



| Parameters | Round Robin | Random | Central Manager | LB Min-Min | LB Max-Min | K-Means Neighbour |
|---|---|---|---|---|---|---|
| Nature | Static | Static | Static | Static | Static | Dynamic |
| Resource Utilization | Less | Less | Less | Moderate | More | More |
| Response Time | Less | Less | Less | Less | Less | Less |
| Process Migration | Not Done | Not Done | Not Done | Yes | Yes | Yes |
| Throughput | Low | Low | Low | Moderate | High | High |
| Waiting Time | More | More | More | More | Less | Less |
| Fault Tolerance | No | No | Yes | No | No | No |

The performance analysis is as shown in Table I. The Round Robin, Randomized, Central manager algorithms are compared with Min-Min, Max-Min and K-means algorithms which are present in this paper.

We can balance the load of the server by transferring the load from heavily loaded server to the lightly loaded server. This maximizes the resource utilization, minimizing the response time, minimizing the waiting time. K means neighbour algorithm is dynamic in nature and resource utilization of this algorithm is more than existing algorithms. Waiting time and response time is less therefore k nearest neighbour algorithm is used for neighbour identification.

## IV. CONCLUSION

Load balancing is a major issue in cloud computing. In this proposed work balance the load of the servers from heavily loaded server to the lightly loaded server. K-means neighbour algorithm is used to migrate one whole process into any one server so that keep the track of information about processes are avoided. Proposed system increases resource utilization and minimizes the waiting and response time.

## REFERENCES

[1] H. Hsiao,Member, H. Chung, H. Shen, and Y. Chao "Load Rebalancing for Distributed File Systems in Clouds" *IEEE Trans. on Parallel and Distributed Systems,* vol. 24, no. 5, May 2013.

[2] R. Revathy and A. Illayarajaa, "Efficient Load Re Balancing Algorithm for Distributed File Systems" *International Journal of Innovative Technology and Exploring Engineering* Vol-2, Issue-6,pp. 2278-3075, May 2013.

[3] Load Balancing,www.f5.com/glossary/load-balancing/,Nov-2013.

[4] Ali M.Alkeel,"a Guide to Dynamic Load Balancing in distriduted computer systems",*IJCSNS International Journal of Computer Science & Network /security*,vol.10 no.6,June 2010.

[5] .W. Byers, J. Considine, and M. Mitzenmacher, "Simple Load Balancing for Distributed Hash Tables,"Proc. First Int'l Workshop Peer-to-Peer Systems),pp. 80-87, Feb. 2003

[6] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica,"Load Balancing in Structured P2P Systems,"Proc. Second Int'lWorkshop Peer-to-Peer Systems ,pp. 68-79, Feb. 2003.

[7] Dabek and M. F . Kaashoek and D. Karger and R. Morris and I. Stoica. "Wide-area Cooperative Storage with CFS", Proc. ACM SOSP 2001.

[8] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications,"IEEE/ACM Trans. Networking,vol. 11, no. 1, pp. 17-21, Feb. 2003

[9] R. P. Padhey, P. Goutam Prasad Rao, " Load Balancing in Cloud Computing Systems", Department of Computer Science and Engineering, National Institute of Technology, May 2011.

[10] Santhosh, Dr. D.H.Manjaiah, "An Improved Task Scheduling Algorithm based on Max-min for Cloud Computing", International Journal of Innovative Research in Computer and Communication Engineering, Vol.2, Special Issue 2, May 2014

[11] U. Bhoi, P. N. Ramanuj, "Enhanced Max-min Task Scheduling Algorithm in Cloud Computing", International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 2, Issue 4, April 2013.

[12] O. M. Elzeki, M. Z. Reshad and M. A. Elsoud, "Improved Max-Min Algorithm in Cloud Computing" , *International Journal of Computer Applications* (0975 – 8887) Vol-50 – No.12, July 2012.

[13] H. Rahmawan, Y. S. Gondokaryono, "The Simulation of Static Load Balancing Algorithms", 2009 International Conference on Electrical Engineering and Informatics, Malaysia.

[14] S.Sharma, S. Singh, and Meenakshi Sharma, "Performance Analysis of Load Balancing Algorithms", academy of science, engineering and technology, issue 38, February 2008, pp. 269-272.