Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7766414

A CENTRALIZED CLUSTER BASED DYNAMIC LOAD BALANCING FRAMEWORK USING TASK SCHEDULING IN CLOUD COMPUTING ENVIRONMENT

Ajay Jangra[1], Neeraj Mangla[2]
[1] Ph.D Scholar, Maharishi Markandeshwar (Deemed to be University) Mullana, INDIA and Assistant Professor, Department of Computer Science & Engineering, Kurukshetra University, Kurukshetra, INDIA
[2]Associate Professor, CSE Department, Maharishi Markandeshwar (Deemed to be University) Mullana, INDIA
[1]er_jangra@yahoo.co.in, [2]erneerajynr@gmail.com

ABSTRACT: Cloud computing is an influencing technology that emphasis on internet-oriented development and computing capabilities of machines. Factors like scalability, adaptability, availability, pay for use scheme, virtualization and data handling within infinite space make it as a good option to be adopted by customers. With all these benefits, cloud computing has intended several trends in the area of computing and there are still some challenges that users face and one of the most important is load balancing. In this paper, centralized load balancing algorithm has been proposed that dynamically balance the load and ensures overall performance of the system. It aims to serve both end users and service providers profitably by managing data efficiently and focus on achieving high resource utilization, reduced job rejections, improved computational capabilities and building a fault tolerant system by creating backups. The results of cluster-oriented load balancing algorithm show reduction in response time, communication overhead and improved processing time. The experimental results thus obtained are presented with a detailed discussion and a comparative analysis may be carried out for checking the merit of newly proposed idea.
KEYWORDS: Cloud computing, load balancing, virtual machine, scheduling, optimization, response time, resource utilization, dynamic clustering.

Introduction:

Cloud computing has become one of the major prosperous platforms for various broad-based technologies. For instance, computing like data from all the computers gets transferred into bigger desktops and thus, invents the studios of their customers again on personal computers [1]. Giant businesses show their keen interest in cloud computing for expanding and enhancing their technology, economy and reachability. Cloud computing is an unceasing technology evolving as a boon in an IT industry that ensures online, on demand services which are accessible anywhere. These services are dynamically reconfigured to reduce the total cost of ownership and increase return on investment to improve the reliability and efficiency of the computing model [2].

It emphases on secure and reliable data management with scalable cloud. The role of Cloud service providers is to help users in the accessing of data from cloud in efficient manner. With the growing volume of data in clouds whether it is private or public, it emphasis on heavy

demand of wide range multi-tenant clouds. Rapidly evolving cloud computing is facing many challenges like storage, data handling, security, concurrency, load balancing etc. Large scale clouds with big data deals with more user requests that may develop heavy traffic in the network with available bandwidth. Random distribution of requests over available virtual machines from different users by data center controller may imbalance the load over various nodes [3]. To overcome such concerns suitable load balancing mechanisms are required to smartly enhance the efficiency of various scheduling mechanisms and performance factors.

Balancing the Work Load on Cloud environment:

Load balancing is a strategy of uniformly distributing workload among different nodes to achieve high user satisfaction and enhance performance factors. It deals with the foremost "task scheduling" with set of specified rules to realize the real ability of dividing the load in cloud. Load balancing is proposed to avoid the wastage of resources in condition of heavy demand and schedule the request in an orderly manner in resource sharing distributed environment to improve the performance. Load balancers are thus applied to balance the heterogeneous user requests using various load balancing optimization techniques [4]. Despite of load balancing in cluster or grid computing, techniques applied in cloud are totally different because of the online and on demand multi-tenant services. There are various optimization techniques like honey bee forging, ant colony, genetic algorithm, swarm optimization etc. has been proposed by researchers to optimize the load in both centralized and decentralized environment. Existing load balancing strategies can be classified as [5, 6,8] :

Static load balancing: In this technique all the decisions for load balancing are predefined based on input. There is no provision for change of decision on the basis of current state of load. It requires prior knowledge of structure of system and availability of resources.

Dynamic load balancing: In this type of load balancing no need of prior knowledge of structure of system and input. The decision of load distribution depends on the current state of the system. The management of load is dynamic in nature.

Centralized load balancing: Centralized load balancing handle multiple instances in one session. Multiple request from different users request for numerous resources and centralized load balancing techniques succeed in serving various request in real time by providing requested resources. This focus on improving resource utilization and response time. There are many intellectually proposed centralized load balancing algorithms known to us.

Decentralized load balancing: In decentralized load balancing strategies, multiple requests are handled in different sessions and thus eliminates the problem of bottleneck nodes. This scheme has been considered better for service providers as well as users in terms of network by maintaining low network congestion. Several decentralized load balancing algorithms have been successfully framed.

Predictive load balancing: Predictive load balancing computes and estimate improved load division according to requirements from the trends that has been traced from previous techniques used in balancing load. It follows the previous study and strategically frame load balancing with its predictive nature. Predictive load balancing based algorithms has proved its significance by improving various computational capabilities.

Cluster load balancing: In this load balancing scheme, different groups of servers are formed that are allocated with different tasks to be executed. Formation of clusters can be either

static or dynamic in nature. Cluster based load balancing algorithms has been considered as most efficient as it supports effective resource utilization and low network overheads. Many novel ideas of cluster load balancing has been projected.

Performance Evaluation

i) ***Resource utilization***: Resource utilization is the extent to which the resources are utilized. As a cloud is the pool for shared resources that are served as services in various forms to clients on some request. Management of resources should be done in such a way that being a most important asset maximum resource utilization may takes place. Many algorithms and models has been framed that enhance the resource utilization of the server pool.

ii) ***Energy consumption***: Data center controller must consume minimal energy while distributing heterogeneous incoming load from users among various nodes. Reduced energy consumption drives towards effective load balancing. There are various energy consumption models like network model, AP energy consumption model, switch energy consumption model etc. that has been projected in the cloud world to provide an energy efficient environment. In AP energy consumption model, two energy modes has been considered low power consumption in which no active users are attached and high power consumption in which AP is attached to active users. Therefore for an energy efficiency, the maximum nodes in the network must be in power saving mode or by switching them off if there is no traffic to carry. The power utilization of AP can be expressed as:

$$P = \begin{cases} Ps\ if\ \text{load Ap} = 0 \\ Pprofile * (0.75 + 0.25 * \frac{load\ ap}{total\ AP\ capacity}) \end{cases}$$

Where, $load_{AP}$ = current load of AP

Total AP capacity = maximum size supported by AP

$P_{profile}$ = peak power utilization of corresponding profile.

iii) ***Throughput***: A good load balancing technique should have high throughput that describes the total no. of tasks being executed in a given unit of time. Several models and algorithms has been proposed that aims at achieving maximum throughput for optimized load balancing technique. So more the no. of tasks completed in per unit more is the throughput. Throughput of a system is given as:

$$T = \frac{N_{complete}}{N_{assigned}}$$

Where, $N_{complete}$= No. of tasks completed in per unit time of an event.

$N_{assigned}$ = No. of tasks assigned to servers for that particular event.

iv) ***Response time***: Response time is defined as time interval between the response to the request and the arrival time of that request. Minimum the response time maximum is the effective load balancing technique. Task will get execute as soon as it will get the service from the computing system. Response time is defined as:

$$\text{Response time} = first\ service\ time - arrival\ time$$

v) ***Fault tolerance and reliability***: Fault tolerance is the ability of system to work accurately in suitable manner even in adverse conditions. The finest load balancing techniques objects to build a desired fault tolerant and a reliable system by creating backups and gaining maximum replies within a cloud environment. Reliability of the system depends upon the favorable replies it get for the tasks of a particular event with respect to the total no of tasks. Higher the reliability, maximum is the system output.

$$P\ (success) = \frac{\sum_{i=0}^{n} no.of\ executed\ tasks}{Total\ no.of\ tasks\ assigned}$$

vi) ***Job rejection***: There may be a situation of some overloaded as well as under loaded nodes due to non-uniform distribution of load. An optimum load balancing technique aims at balancing load among nodes in such a way that maximum jobs can be handled at a time without inadequate rejections. Lesser the probability of failure, lesser is the no. of job rejections.

$$P\ (failure) = 1 - P\ (success)$$

$$1 - \frac{\sum_{i=0}^{n} no.of\ executed\ tasks}{Total\ no.of\ tasks\ assigned}$$

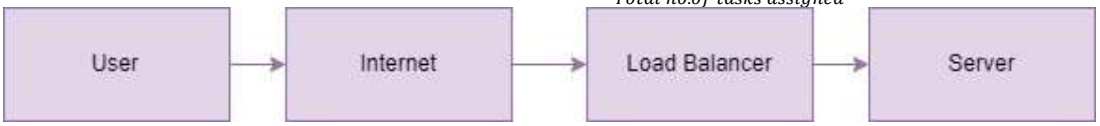

Fig.1 Load balancing

Previous Work:

Many researchers tried to contend this essential issue in cloud computing to improve functioning of cloud by proposing various load balancing techniques that considered different prime factors and enhanced the way of balancing load. Load balancing algorithms focus on mapping the load depending on various proposed methods and has been classified on different features to optimize the computational capabilities of cloud [8].

Load balancers drive the tasks to resources by employing various functions to achieve desired objectives. Cloud computing provides us resources in the form of services under the policy of "pay as you go" with an appropriate service level agreement. Sometimes, these load balancers are applied on services existing or served load balancer as a service. Mazedur Rahman, Samira Iqbal and Jerry Rao [9] proposed load balancer as a service (LBaaS) for balancing load in different OSI layers on the basis of strength, shortcomings and characteristics of load balancing techniques with excellent features like access control, usage tracking, billing, multi-tenancy, service connection, connection and security log etc. and Monika Simjanoska, SaskoRistov [10] gave a novel idea of low level load balancer (L3B) applied to infrastructure as a service (IaaS) that tends load in a way to provide maximum performance and resource utilization. It sustain features like cloud elasticity by dynamically providing resources on low network and reduce customer renting.

Different cost aware models describing work flow algorithms were announced to schedule resources among incoming requests. Relevant cost aware challenges are classified based on system functionality, system architecture, quality of service and performance refining the cost of proposed idea. Enhancement in existing algorithms or new cost scheduling algorithms have been introduced to achieve main goals. Antony Thomas [12] in his paper has shown improvisation in min-min algorithm based on user priority and task length without giving special attention to the highest priority task whereas Monir Abdullah and Mohamed Othman [13] has studied the divisible load theory (DLT) and investigated its uses for designing a cost based multi Qos job scheduling for efficient strategy to minimize the processing time. Mrs. Nagamani H. Shahapure and Dr. JayarekhaP [14] had introduced a cost scheduling algorithm called load balancing with optimal cost scheduling algorithm in which list of services from service pack are scheduled to virtual machines depending upon their configuration and computing power with minimum execution cost.

In cluster environment, network is divided into clusters that consist of various nodes on which various clustering algorithms are applied accordingly to obtain the desired objective. Mohammad S. Obaidat [15] has given a decentralized cluster based load balancing algorithm supporting scalability, heterogeneity, low network congestion using cloud sim whereas also the trend based prediction using ELB load balancing product of cloud has been introduced by He-Sheng WU to increase elasticity in cluster environment [16].

There are various optimization algorithm has been projected that involves dividing the load equally to resources and improves the throughput and performance of the system. Well known algorithms like ant colony, swarm optimization, honey bee, genetic algorithm followed best suitable scheduling algorithm and worked on different environment to improve the desired parameters representing fault tolerance, scalability, resource utilization and many others.

Different proposed models available that ensures to improve the load balancing in cloud computing whose working has been based on the amendments being done in existing model or

new novel models has been proposed for dynamic environment. Depending on Bayesian cognitive model, Wei Wang and Guosun Zeng [16] in their paper has proposed to reduce the failure probability of assigning task and assurance of secure task execution using a trust dynamic scheduling algorithm. Also, a better model has been given by researchers using switch mechanism to adapt different strategies according to environment to balance load and improving resource utilization and availability using ANFIS and GSO load balancing algorithms.

There are various computational capabilities that are being considered and achieved for load balancing in cloud computing. Among all these, resources are the most important assets whose effective allocation is gained with proper queuing system and scheduling. ShahinValkini [22] has modeled the systems with both constant and variable size jobs using birth-death process for resource allocation whereas Ritu Garg [23] in herpaperhas proposed adaptive workflow scheduling (AWS) that deals with resource availability heterogeneous dynamic environment .Multi resource load balancing algorithm for distributed cache aim at balancing cache and memory resources. An effective simulation has shown improvement in time efficiency and reconfiguration cost [27, 28, 29].

Characteristics	Literature	Nature	Environment	Proposed concept	Strength
LOAD BALANCER	**[9]**	Dynamic	Heterogeneous	Critically examined load balancing techniques. Proposed load balancing as a service.	Provides assessment on existing LBtechniques. Service model for LB.
	[10]	Dynamic	Heterogeneous	Realized need for real time resource orchestration. Applied Novellow level load balancer to IaaS.	Preserves cloud elasticity. Reduce power consumption. Reduced customer cost. Improves overall performance.
COST	[11]	Dynamic and static	Heterogeneous	Scheduling techniques on the basis of cost and workflow. Presents challenges / findings of cost-based scheduling techniques.	Facilitate various aspects to find out optimal cost aware routing mechanism. Reduce power utilization.
	[12]	Static	Homogeneous	Reviewed traditional algorithms based on user priority and task length.	Improved resource utilization. Achievement of user satisfaction. Less make span of tasks.

				Improved scheduling algorithm is proposed.	
	[13]	Static	Homogeneous	Investigated the use of distribution of load theory mechanisms. Designed efficient strategies for scheduling jobs	Minimized processing time. Maximum benefit gained by service provided. Enhanced quality of service.
	[14]	Dynamic	Heterogeneous	Examined various LB optimization techniques Proposed an Optimal Cost Scheduling Algorithm .	Minimized execution cost Reduced execution time. Dynamic scheduling improves overall performance.
CLUSTER	[15]	Decentralized	Heterogeneous	Reckoned the importance of LB. Realized goal of LB. Proposed decentralized cluster-based algorithm.	Supports heterogeneity. Achieved scalability. Low network congestion. Eliminated bottleneck nodes.
	[16]	Predictive	Heterogeneous	Utilize resources dynamically with less time consumption. Insufficient elastic management of resources. Prediction-based LB algorithm	Performed prediction based LB. Provided high availability. Improved response time. Achieved scalability.
OPTIMIZATION ALGORITHM	[17]	Dynamic and static	Homogeneous And heterogeneous	Discussed LB goals and concept. Compared parameters based on throttled scheduling and round robin algorithm.	Concluded that Throttled LB algorithm consumes less cost.
	[18]	Dynamic and static	Homogeneous And heterogeneous	Stated various issues related to LB. Studied several LB optimizing techniques.	Measured and discussed various performance parameters. Comparison among different techniques.

MODEL	[19]	Predictive	Heterogeneous	Predict and calculate requirements of virtual machines. Proposed a scalable framework for forecasting load	Accurate estimation of load. This load-balancing framework has better effect compared with the default method of Xen server.
	[20]	Dynamic	Heterogeneous	Inspired by Bayesian cognitive model.	Execution of tasks in more secure way Improved execution time and reliability
	[21]	Dynamic and static	Heterogeneous	Roots of this paper resides in ANFIS and GSO algorithm. Presented refined LB algorithm.	Improved resource utilization. Measureable increase in availability.
RESOURCE ALLOCATION	[22]	Dynamic	Homogeneous And heterogeneous	Developed various performance models.	Analyzed tradeoffs for power consumption. Determined service fragmentation probabilities.
	[23]	Dynamic and Static	Heterogeneous	Deals with heterogeneous dynamic grid environment	QoS based Efficient resource utilization Minimum execution time. Supports rescheduling of tasks. Adaptable to changes.
	[24]	Dynamic	Homogeneous	proposed a multi-resource LB algorithm. An extensive simulation is conducted.	Improved time efficiency. Reduced reconfiguration cost. Balanced CPU and memory usage. Low Execution time Minimized system imbalance degree.

Table 1 Summary of various LB approaches.

Problem identification:

Load Balancing has become a one of the most important challenge in research area of cloud computing for efficient working of distributed cloud environment. The need for balancing load comes into account when explosive rise of data increases service request that are expected to be scheduled equally among available nodes. Increased volume of data needs to handle more user requests at a given instance of time which initiates imbalance of load among available

nodes and depreciate the resource utilization and throughput of the system. To conquer over random distribution of requests an augmented Load Balancing strategies are needed to be followed to enhance system output and complement in the progress of this burning technology. [25, 27, 30]

In a cluster based Load Balancing algorithm in cloud computing with a concept of master and slave environment has been proposed by using cluster Load Balancing in which task of Load Balancing has been divided into two modules:

i) ***Load distribution among masters****:* Each cluster is consists of a single master and various slaves and on arrival of every task performance factor of a master is calculated that describes the capability of the master and slaves associated with it and then, the task size is matched with the capability.

If (performanceCurrentMaster<TaskSize)
{ Broadcast Task; }
Else {
Broadcast task;
Broadcast Performance factor;
}

The role of master executes only when restricted flooding algorithm finishes.
PerformanceCurrentMaster = memory (m) + bandwidth(b) + storage(s)
On receiving a task size master checks the performance factor, if it is less than task size and master is not an inter cluster node discard it else it is suitable and add it as current master. On time expiration either change the cluster or the master.
*PerformanceCurrentMaster = w_1*available Memory + w_2*available Storage + w_3*availableBandwidth*
Where, w_1, w_2 and w_3 are some weight factors.

The token passing algorithm is used in the presence of more than one intercommunication node that is master token is being propagated to select an operating ICC for that cluster.

ii) ***Load distribution from masters to slaves****:* The Cloud Analyst toolkit provides three inbuilt scheduling algorithms from master to slave: Round Robin which is used in this algorithm, Equally Spread Current Execution Load and Throttled. Performance metrics used are load per node, percentage of tasks executed and average processing time for execution of tasks.

*Percentage of tasks = Tasks Executed/Total Tasks Received * 100*

Although this is an efficient method for improving few computational capability and reduced network congestion but also increased communication overhead that effects resource utilization and response time of the system. To improve these consequences following solution has been offered.

Following are the goals to be achieved in this paper with respect to Load Balancing:

(i) To provide optimized task scheduling and resource utilization.
(ii) To reduce the job rejections.
(iii) Building a fault tolerant system by creating backups.
(iv) Avail the system with optimized throughput.
(v) Improving computational capabilities.
(vi) Efficient dynamic Load Balancing.

Proposed solution:

A likely solution to this problem has been proposed in an efficient task scheduling Load Balancing algorithm using dynamic cluster based mechanism. In fig 2, a working model demonstrates the projected concept by introducing a centralized dynamic cluster based algorithm that reduce communication overhead, improve response time and increase resource utilization. There are various services offered by the cloud service provider to serve the different requests from clients and resourceful LB is one of them.

In this novel scheme, the cloud network is divided into master-slave clusters that are created dynamically to balance the load among the slaves which are computing nodes of the cluster environment. Clients request for the services from the server pool through cloud based delivery followed by a supervisor that acts as an agent. The supervisor maintains a uniform distribution of incoming service request through the World Wide Web from

different clients that tends to access data as a resource by maintaining a check over different clusters through its database table which demonstrates which cluster is feasible to execute the task. When it finds an appropriate computing element to complete the execution of tasks, it command the load balancer to dynamically direct service request to master-slave cluster.

PerformanceCurrentMaster= no. of slaves with SLAVE_BIT_STATUS 0

Supervisor and leader node of clusters are updated on every incoming request and task completion through slaves and SLAVE_BIT_STATUS decides the availability of the slave.

If (SLAVE_BIT_STATUS==1)
{Slave is performing some tasks ;}
Else
{ Slave is free to execute incoming task ;}

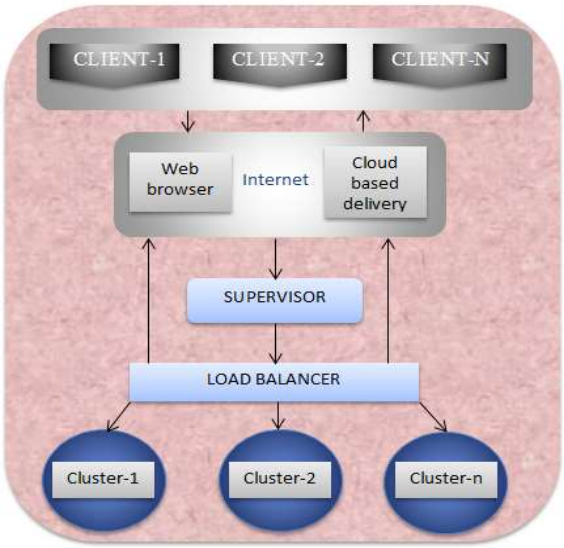

Fig.2.Cluster based LB model

In the case of heavy traffic, supervisor depicts the need of creating new clusters to balance the load as if all the computing nodes are engaged with some task. It only notifies the requirement whereas cloud service providers are responsible to create a cluster for smooth execution in master-slave environment. Every cluster is consists of a single master and various slaves where masters are responsible for inter cluster communication that communicates with all other ICC nodes and slaves are computing nodes that handles tasks execution.

Proposed solution:

A likely solution to this problem has been proposed in an efficient task scheduling Load Balancing algorithm using dynamic cluster based mechanism. In fig 2, a working model demonstrates the projected concept by introducing a centralized dynamic cluster based algorithm that reduce communication overhead, improve response time and increase resource utilization. There are various services offered by the cloud service provider to serve the different requests from clients and resourceful LB is one of them.

In this novel scheme, the cloud network is divided into master-slave clusters that are created dynamically to balance the load among the slaves which are computing nodes of the cluster environment. Clients request for the services from the server pool through cloud based delivery followed by a supervisor that acts as an agent. The supervisor maintains a uniform distribution of incoming service request through the World Wide Web from different clients that tends to access data as a resource by maintaining a check over different clusters through its database table which demonstrates which cluster is feasible to execute the task. When it finds an appropriate computing element to complete the execution of tasks, it command the load balancer to dynamically direct service request to master-slave cluster.

PerformanceCurrentMaster= no. of slaves with SLAVE_BIT_STATUS 0

Supervisor and leader node of clusters are updated on every incoming request and task completion through slaves and SLAVE_BIT_STATUS decides the availability of the slave.

If (SLAVE_BIT_STATUS==1)
{Slave is performing some tasks ;}
Else
{ Slave is free to execute incoming task ;}

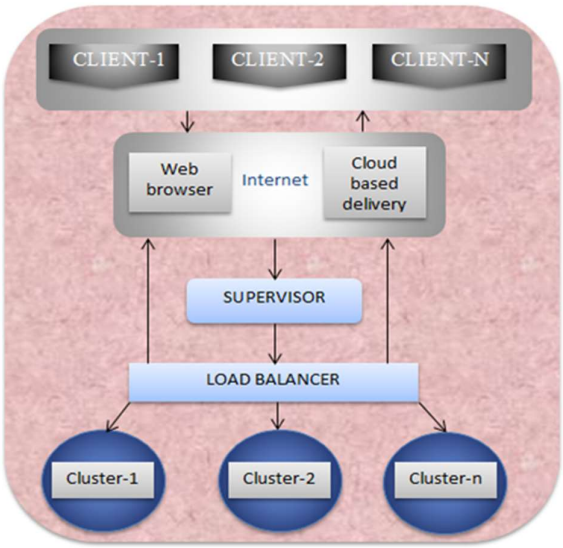

Fig.2.Cluster based LB model

In the case of heavy traffic, supervisor depicts the need of creating new clusters to balance the load as if all the computing nodes are engaged with some task. It only notifies the requirement whereas cloud service providers are responsible to create a cluster for smooth execution in master-slave environment. Every cluster is consists of a single master and various slaves where masters are responsible for inter cluster communication that communicates with all other ICC nodes and slaves are computing nodes that handles tasks execution.

Proposed Framework:

This section is divided into three subsections. Section I describes proposed architecture whereas section II gives the progressive algorithm on which the model works. Section III represents the technical process flow diagram with its explanation.

Architectural Description

Effective Load Balancing in cloud computing with increasing volume, velocity and veracity of data is a major concern to be achieved and improved. The proposed architectural design of master slave clustered environment focuses on entailing the improved Load Balancing in current scenario of cloud computing by improving various computational parameters. In distributed environment, clients request for the data from cloud servers using hypertext transfer protocol through web browser. There are many leading trends of cloud computing and challenges that are faced by intellectual researchers now a days and security is one of the most important. Authorized traffic is filtered at network level using firewall and intrusion detection system defined by some security policies for security. The request load is then monitored by cloud monitoring module and is then equally distributed among computing nodes using routing switch availability.

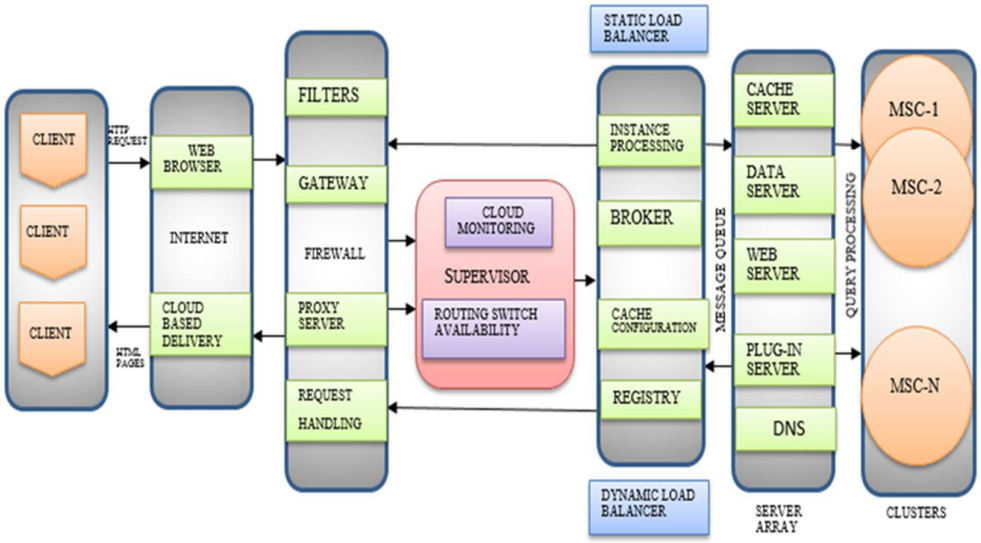

Fig.3 Dynamic Cluster based Load Balancing architecture

Load Balancing is done either dynamically or statically depending upon the type of data that is being requested by the user through static load balancer and dynamic load balancer respectively. Static load balancer do not require knowledge of current state and strictly depends upon predefined rules whereas dynamic load balancer balances the load depending upon the current state of the system and is preemptive in nature. In dynamic Load Balancing instant processing is done as the request arrives and forwarded to the server array that act as an interface between master cluster environment and request scenario. Different servers available in the array make information retrieval system and processing of data quiet easier and simple. Overall performance of the system is enhanced by effective query processing system that are processed in cluster environment consisting of computational units known as slave which are guided by inter cluster communication node. Tasks are directed to slaves through masters where computation is being done and results are given to clients through cloud based delivery module of interface in the form of html pages. This systematic query processing ensures efficient Load balancing in cloud environment and stated architectural model helps us to achieve improvement in computational capability of the system.

Progressive Algorithm:

This progressive algorithm is dynamic cluster oriented task scheduling algorithm which is centralized in nature that focuses on improved resource utilization and response time. It is distributed into different steps and work as follows:

a) STEP 1:*PROCESS_MAIN ()*

```
Initialize matrix Database_Table[i][j], query Q = {q1, q2, ... qn, qn+1, .. qm}, MAX_i =
[k], Master_Bit = [m];

Set arr[0: i − 1][0: j − 1], Limit[n];
Call SETUP_SERVER_HANDSHAKE();
Check query Q;
DATABASE_TABLE_CHECK()
{
        If (SLAVE_BUSY_COUNT MAX_i)
    {
            CASE MAX_i = 5 : master is 100% busy;
            CASE MAX_i = 4: master is 80% busy;
            CASE MAX_i = 3 : master is 60% busy;
            CASE MAX_i = 2 : master is 40% busy;
            CASE MAX_i = 1 : master is 20% busy;
    }
        SUPERVISOR_CHECK ()
        {
            If (CASE MAX_i = 5)
            {
                Toggle MASTER_BIT=1;
                Call SETUP_NEW_CLUSTER ();
            }
            Else
            {
                Master_Bit = 0;
                Identify SLAVE = free;
                ASSIGN work to MAX_i ≠ 5
            }
            SLAVE_LOCK = True
    }
}
MASTER updates database from MAX_i slaves with q_i path;
SUPERVISOR monitors Query q_j access;
SLAVE_LOCK = False
Balances load with Q;
END
```

STEP 2: SETUP_SERVER_HANDSHAKE()

```
Install SERVER_INTERFACE;
Create SOCKET (for UDP & TCP)
{
        Check for DNS servers
        Read MAC address(X.X.X.X) for DESTINATION UDP
        Bind SOCKET to WELL_KNOWN_TCP_PORT
        Foreach (3 − WAY HANDSHAKE)
            {
            Client: sends Synchronize (SYN)
            REQUEST from SOURCE_PORT
            Server: DESTINATION_PORT READ REQUEST
            Formulate REPLY
            Client: SOURCE_PORT receives REPLY
            Receive Acknowledge (ACK)
            }
}
```

STEP 3:SETUP_NEW_CLUSTER ()

```
Setup New_Server (MASTER, SLAVES)
{
        Do (create server)
        While (Master = 1, Slave_Limit = n)
        {
                Assign Cluster_Access_Value;
                Call MySQLdump() to set Backup_Table for
                cluster nodes;
                Pair (Backup_Table, Cluster_Access_Value);
                Update query q1 access results;
        }
}
```

Process Flow Diagrams:

Task is to be distributed by adopting different mechanism one is for Clusters and other is for Slaves.

Task scheduling among clusters:

For execution of any task, user request for resources through cloud service providers. Tasks are equally distributed in dynamic cluster environment using load balancers for effective scheduling supported by a supervisor. Supervisor is an agent applied for smooth mapping of tasks and keeps the track of all the clusters thus formed with their ID's. It checks the capacity of cluster whether it is capable of executing that task or not. If yes, then that task is assigned to cluster and processing of that task takes place at desired unit otherwise it commands the cloud service provider to create a new cluster and then allocate that task for execution. After executing the task, the database it updated at different related modules.

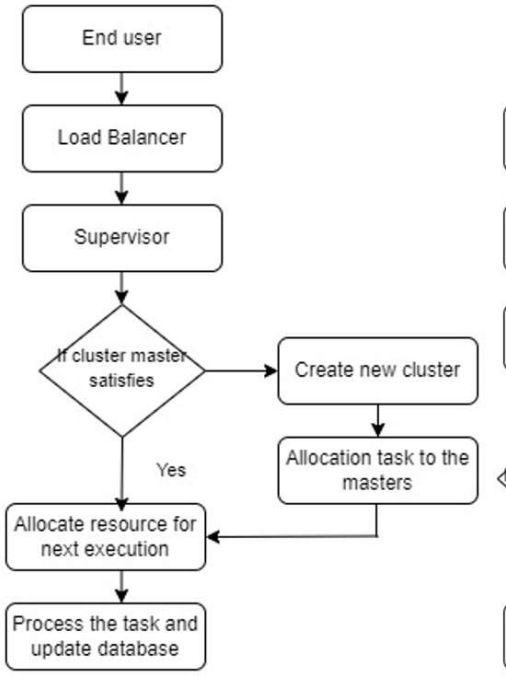

Fig.4 Task scheduling among clusters

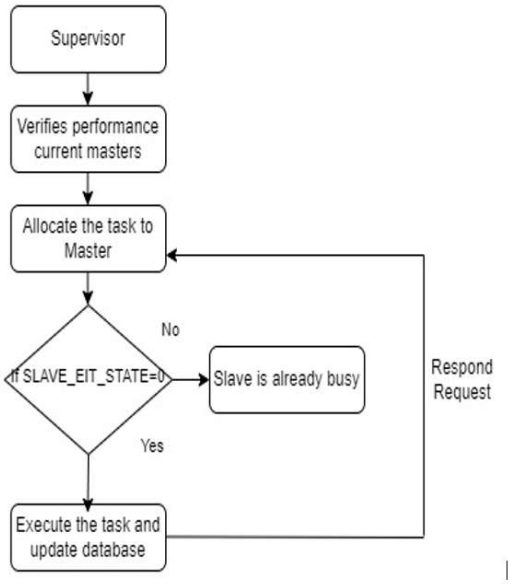

Fig.5 Task scheduling among slaves

Task scheduling among slaves:

Tasks are distributed among slaves through master of each cluster whose performance factor is verified by Supervisor. Performance factor of a master depends on the no. of computing

nodes that are available for execution of a task. After this the master checks for the Slaves availability through the SLAVE-BIT-STATUS. If it is 0, that means it is available for execution and the task is allocated to the slave and processing takes place otherwise it is busy and no more task can be assigned to it. After executing the task, slave respond request to master and also updates the database including all modules associated to it.

Implementation:

This section describes the details of modules thus generated foe execution environment and the setup phases for implementing the proposed concept:

A. Proposed Technical Module Description

a) ***Cluster**:* According to proposed methodology, the network is divided into clusters by applying above mentioned SETUP_NEW_CLUSTER () algorithm into one inter cluster communication node called master and other as slaves. There can be n number of clusters with their unique cluster ID's consist of masters and master's data. The set of cluster ID's is defined as: C_ID= {C_ID 1, C_ID 2, C_ID 3, C_ID 4……..C_ID n}.

S.NO	ATTRIBUTE	DESCRIPTION
1	SUPERVISOR_ID	supervisor id for each cluster is different
2	C_ID	unique identity no. of cluster initiated
3	MASTER_ID	This descripts identification of intercommunication node of a cluster. There is always a single master in a cluster.
4	NO_OF_SLAVES	Initially there are 3 slaves taken when cluster is formed dynamically. There can be n no. of slaves
5	REMANING_SLAVES	This shows the no. of slaves that are currently free in cluster and can be allotted with some task.
6	SLAVE_BIT_STATUS	0= if bit status is 0 it shows that some of the slaves are free in the cluster, it can be 1 or more than one. 1= if bit status is 1 it shows all the slaves of a cluster are busy.

Table 2 Attribute description of cluster module

b) ***Supervisor**:* Supervisor is responsible for dynamically balancing the load by uniformly assigning tasks to appropriate slave of a cluster. It maintains tables for all masters and their data with the attributes as BIT_STATUS, NO_OF_SLAVES, REMAINING_SLAVES, MASTER_ID and SUPERVISOR_ID at every incoming task and completion of task updating of the table will take pace. Before allocating the task to master, following process is followed:

- ✓ Checks load of all the slave by initiating the masters of all the clusters.
- ✓ Allocate the process to the desired slave.
- ✓

c) ***Master**:* Every initiated cluster will have a single inter cluster communication node called as master node. Master is a leader node of cluster that balance load among slaves. It will maintain data of slave nodes in database form that contains attributes as SLAVE_NO and SLAVE_BIT_STATUS describing the load on respective slave.

S.NO	ATTRIBUTE	DESCRIPTION
1	SLAVE_NO.	Slave no. is unique identity no. of a slave in a cluster
2	SLAVE_BIT_STATUS	0= if slave bit status is 0,it means it is free. 1= is slave bit status is 1, it means it is busy with some task.

Table 3 Attribute description of master module

d) ***Slave****:* Slave is a computing node of cluster environment that handles task execution and is directly connected to leader node. The role of slave is done when it updates the master table. There can be n no. of slaves in a cluster and set representation of slaves is as follows:

Slaves = $\{S_1, S_2, S_3, S_4 \ldots\ldots\ldots S_n\}$.

S.NO	ATTRIBUTE	DESCRIPTION
1	SLAVE_NO.	Slave no. is unique identity no. of a slave in a cluster
2	SLAVE_BIT_STATUS	0= if slave bit status is 0,it means it is free. 1= is slave bit status is 1, it means it is busy with some task.

Table 4 Attribute description of slave module

B. System and Experimental Setup Phases

For the evaluation of the proposed work, cluster slave environment is generated with the system attributes and its configuration shown below as well as incorporated with the supervisor for the improvement in execution of the tasks. For experimenting, this idea of dynamic generation of cluster environment is accomplished while implementing this cluster algorithm XAMP server is used with increased cache size and execution time.
The system configuration for implementing this scheme and experimental setup foe execution is tabulated below:

Source	Technical attributes	Configuration
Supervisor	Apache Server	Used XAMPP Server, an open source platform with Custom Settings with increased Cache Size and execution time for experimental purpose.
	MySQL Server	
Cluster 1	Apache Server	
	MySQL Server	
Cluster 2	Apache Server	
	MySQL Server	
Cluster N	Apache Server	
	MySQL Server	

Table 5 Technical configuration details for experiment phase

Results:

To implement and validated proposed mechanism the attributes are taken small at initial stage and varied or expanded according to demand of users. Fig.6 depicts the execution time of eight different processes.Processes are allocated to the modules efficiently even though earlier processes are being carried out simultaneously whereas Fig.7 demonstrates the execution time comparison between three processes each carrying n queries for execution. When the supervisor is applied to the cloud's LB environment, it improves the execution time as it do not

check the status of computing nodes every time instead only verifies its bit status and maps the task. Soit improves its response time as well as its execution time by applying a supervisor to provide a check on current status of the nodes.

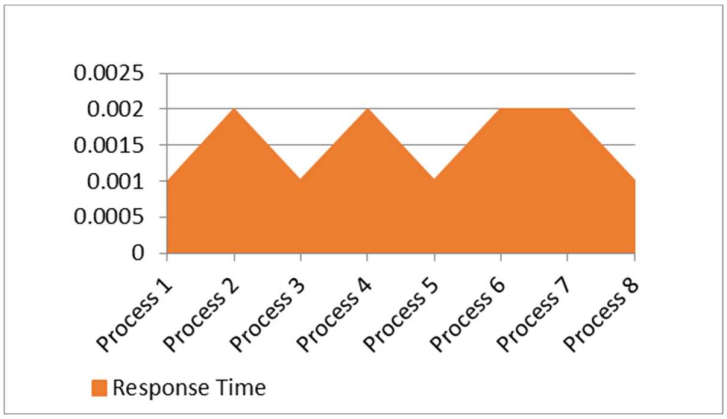

Fig.6 Execution time graph for 8 processes

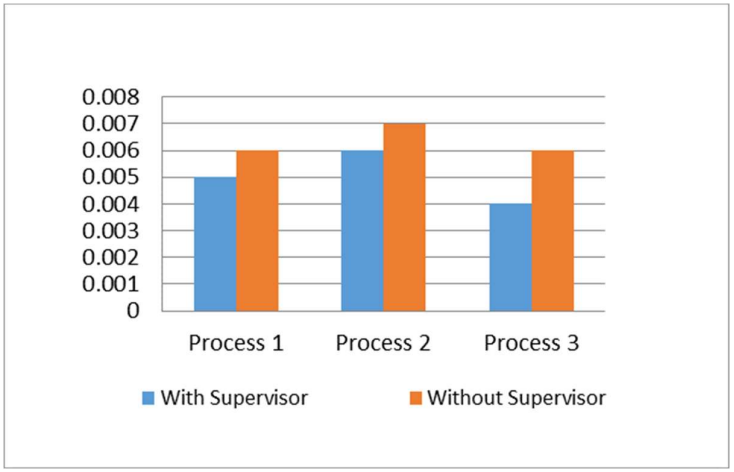

Fig.7 Execution time graph with and without supervisor

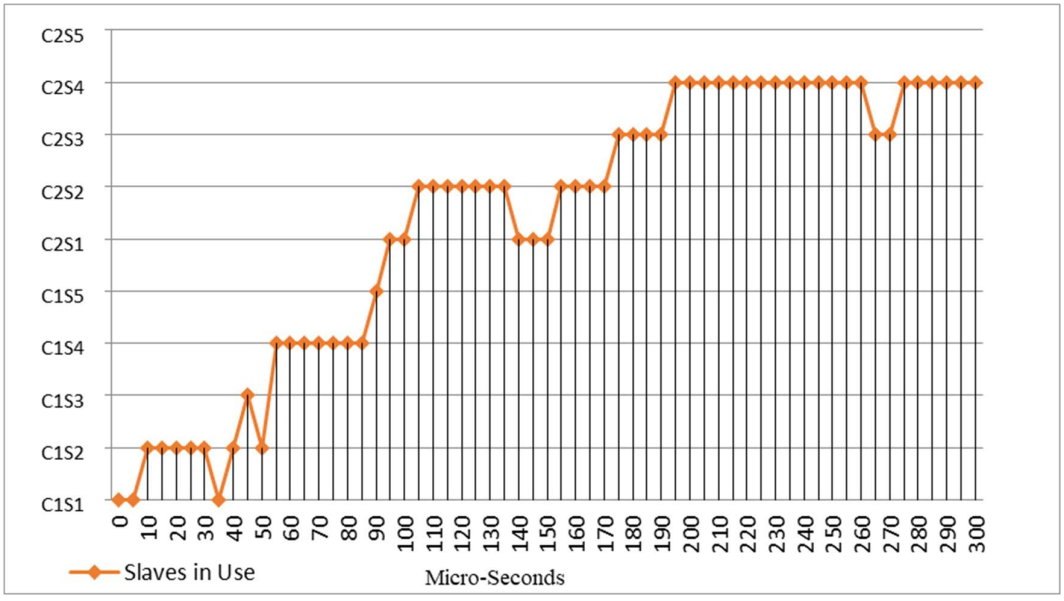

Fig.8 slave usage graph for query processing

In fig.8, it shows the slave usage for the various query execution. The distribution of task is dynamic and continues. Here if one slave is busy then task is assigned to next available slave for the execution.whereas in fig 9, the distribution of load with less consumption time is depicted. Here 40 processes are used for distribution the load distribution is dynamic. Fig.10 sampling is done in 5 main stages through which each process passes while execution. For different 5 process it shows the comparison between 5 stages initially from bit status checking followed by allocation till its execution time. Each sample the time is taken in a microsecond that shows the improved processing time.

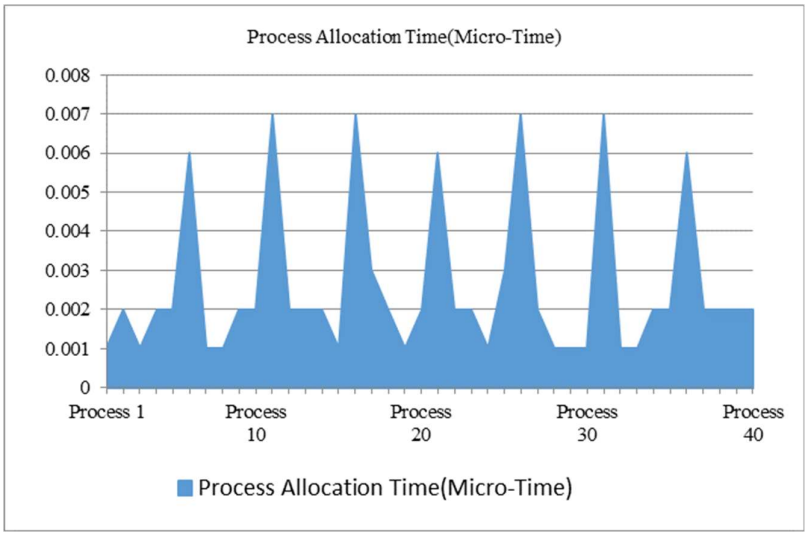

Fig.9 Process allocation response time graph

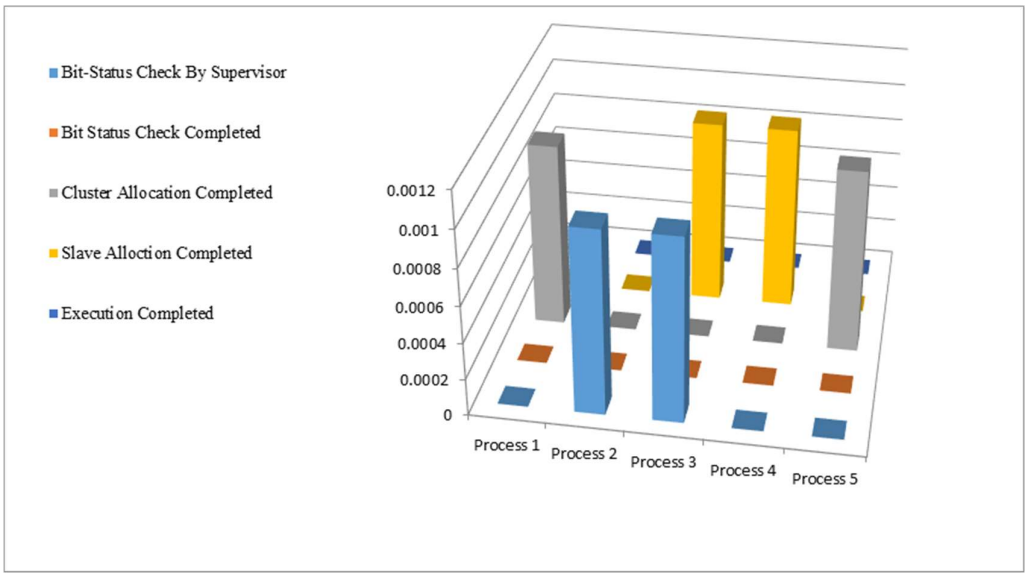

Fig.10 Comparison graph for process time improvising.

In accordance with other factors, improvement in response time has also been experienced. For implementation, 8 processes has been considered with respect to the time at which it get first service. Fig. 11 shows the response time thus observed when the supervisor is applied in computing environment whereas, in fig. 12 the comparison has been made between the

implementation of supervisor environment and without supervisor. This comparison shows the improvement in the response time when the supervisor is applied to it as it saves the time at every incoming task by avoiding the checking of every detail of intercommunication node instead it simply check the STATUS_BIT.

Fig.11 Response time graph for 5 processes

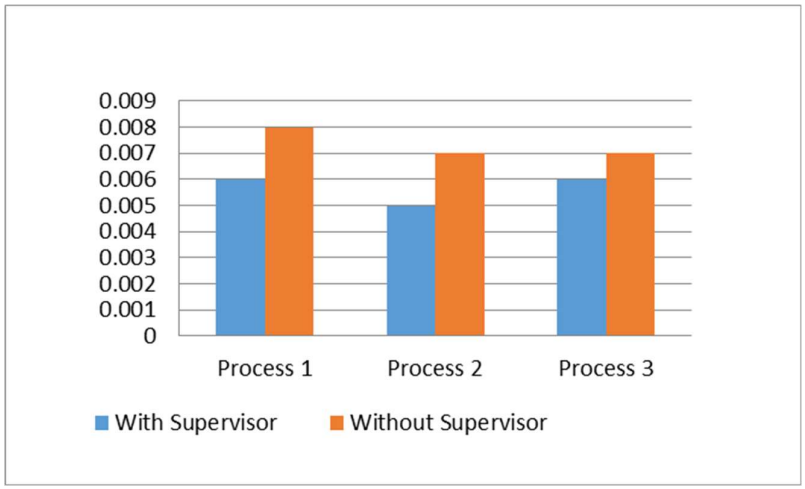

Fig.12 Response time graph with and without supervisor

Conclusion:

Rapidly growing data constitute heavy load on cloud that becomes difficult for service provider to uniformly direct request among various computing nodes. To ensure equal distribution of tasks over distributed cloud environment for an efficient LBa novel LB mechanism based on centralized clustering policy has been proposed. This policy gives an improved cluster oriented LB algorithm which is centralized in nature and balance load dynamically by carrying smooth execution of all incoming service requests. The proposed cluster based model deliberately improves the processing time, resource utilization and reduced response time of the system by using demonstrated the concept of driving a load to suitable computing nodes. Updating the current status of computation on arrival and completion of the task and creating backups at leading nodes makes it a fault tolerant system. The performance may be enhanced by applying any optimized algorithm like genetic algorithm, ant colony optimization, swarm optimization etc. This work may be extended for optimization, security and privacy concerns also.

Data availability:

The data are available on request from the corresponding author.

Conflicts of Interest:

The Authors declares that they have no conflict of interest.

RFERENCES:

Ragmani, Awatif, Amina El Omri, Nouredine Abghour, Khalid Moussaid, and Mohammed Rida. "A performed load balancing algorithm for public Cloud computing using ant colony optimization." Recent Patents on Computer Science 11, no. 3 (2018): 179-195.

Ghomi, Einollah Jafarnejad, Amir Masoud Rahmani, and Nooruldeen Nasih Qader. "Load-balancing algorithms in cloud computing: A survey." Journal of Network and Computer Applications 88 (2017): 50-71.

Polepally, Vijayakumar, and K. Shahu Chatrapati. "Dragonfly optimization and constraint measure-based load balancing in cloud computing." Cluster Computing 22, no. 1 (2019): 1099-1111.

Dhari, Atyaf, and Khaldun I. Arif. "An efficient load balancing scheme for cloud computing." Indian Journal of Science and Technology 10, no. 11 (2017): 1-8..

Singh, Niharika, Upasana Lakhina, Ajay Jangra, and Priyanka Jangra. "Verification and identification approach to maintain MVCC in cloud computing." International Journal of Cloud Applications and Computing (IJCAC) 7, no. 4 (2017): 41-59..

Lakhina, Upasana, Niharika Singh, I. Elamvazuthi, F. Meriaudeau, P. Nallagownden, G. Ramasamy, and Ajay Jangra. "Threshold based load handling mechanism for multi-agent micro grid using cloud computing." In 2018 International Conference on Intelligent and Advanced System (ICIAS), pp. 1-6. IEEE, 2018.

Priya, V., C. Sathiya Kumar, and Ramani Kannan. "Resource scheduling algorithm with load balancing for cloud service provisioning." Applied Soft Computing 76 (2019): 416-424..

Lakhina, Upasana, Niharika Singh, and Ajay Jangra. "An efficient load balancing algorithm for cloud computing using dynamic cluster mechanism." In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), pp. 1799-1804. IEEE, 2016.

M. Rahman, S. Iqbal and J. Gao, "load balancer as a servicein cloud computing," IEEE 8th International Symposium onService Oriented System Engineering (SOSE), 2014 , pp.204-211, 7-11 april 2014.

M. Simjanoska, S. Ristov, G. Velkoski and M. Gusev,"L3B: Low Level Load Balancer in the Cloud," IEEEEUROCON, pp. 250-257, 1-4 july 2013.

S. P. L. ,. S. U. R. K. Ehab NabielAlkhanak, "Cost-awarechallenges for workflow scheduling approaches in cloudcomputing environments: Taxonomy and opportunities,"Future Generation Computer Systems, vol. 50, pp. 3-21,seepteber 2015.

G. K. V. J. R. Antony Thomasa, "Credit Based SchedulingAlgorithm in Cloud Computing Environment," Proceedingsof the International Conference on Information andCommunication Technologies, vol. 46, pp. 913-920, 2015.

M. O. Monir Abdullaha, "Cost-based Multi-QoS JobScheduling Using Divisible Load Theory in CloudComputing," 2013 International Conference onComputational Science, vol. 18, pp. 928-935, june 2013.

N. H. Shahapure and J. P, "Load Balancing with OptimalCost Scheduling Algorithm," 2014 InternationalConference on Computation of Power, Energy, Informationand Communication (ICCPEIC), ,pp. 24-31, april 2014.

S. K. Dhurandher, M. S. Obaidat, I. Woungang and P.Agarwal, "A cluster-based load balancing algorithm incloud computing," 2014 IEEE International Conference onCommunications (ICC), pp. 2921 - 2925, june 2014.

H.-S. Wu, C.-J. Wang and J.-Y. Xie, "TeraScaler ELB-anAlgorithm of Prediction-Based Elastic Load BalancingResource Management in Cloud Computing," 27thInternational Conference on Advanced InformationNetworking and Applications Workshops (WAINA), 2013 ,pp. 649-654, march 2013.

H. Shoja, H. Nahid and R. Azizi, "A comparative survey onload balancing algorithms in cloud computing,"International Conference on Computing, Communicationand Networking Technologies (ICCCNT), 2014, pp. 1-5, 11-13 july 2014.

A. Hans and S. Kalra, "Comparative Study of DifferentCloud Computing Load Balancing Techniques,"International Conference on Medical Imaging, m-Healthand Emerging Communication Systems (MedCom), pp. 395-397, 7-8 november 2014.

Z. Zhang, L. Xiao, Y. Tao and J. Tian, "A Model BasedLoad-Balancing Method in IaaS Cloud," 2013 42ndInternational Conference on Parallel Processing, pp. 808-816, october 2013.

b. G. Z. b. D. T. J. Y. Wei Wanga, "Cloud-DLS: Dynamictrusted scheduling for Cloud computing," Expert Systemswith Applications, vol. 39, no. 3, pp. 2321-2329, feb 2012.

Suresh, M., and S. Karthik. "A load balancing model in public cloud using ANFIS and GSO." In 2014 International Conference on Intelligent Computing Applications, pp. 85-89. IEEE, 2014.

M. M. A. ,. D. Q. Shahin Vakilinia, "Modeling of theresource allocation in cloud computing centers," ComputerNetworks, vol. 91, pp. 453-470, november 2015.

A. K. S. Ritu Garg, "Adaptive workflow scheduling ingrid computing based on dynamic resource availability,"Engineering Science and Technology, an InternationalJournal, vol. 18, no. 2, pp. 256-269, june 2015.

R. kumar and G. sahoo, "a multi resource load balancingalgorithm for cloud - cache system," International Journalof Information Technology Convergence and Services(IJITCS), vol. 3, no. 5, october 2013.

Afzal, Shahbaz, and G. Kavitha. "Load balancing in cloud computing–A hierarchical taxonomical classification." Journal of Cloud Computing 8, no. 1 (2019): 1-24.

Jangra, Ajay, and Renu Bala. "A Survey on various possible vulnerabilities and attacks in cloud computing environment." International Journal of Computing and Business Research 3, no. 1 (2012): 1-13.

Ankur Lohchab, Anu Lohchab and Ajay jangra" A comprehensive survey of prominent cryptographic aspects for securing communication in post-quantum IoT networks" Internet of Things 9 (2020) Elsevier 2542-6605/© 2020 Elsevier B.V. All rights reserve https://doi.org/10.1016/j.iot.2020.100174

Lohachab, A., & Jangra, A. (2019, March). Opportunistic Internet of Things (IoT): Demystifying the Effective Possibilities of Opportunisitc Networks Towards IoT. In 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN) (pp. 1100-1105). IEEE.

Singh, N., Elamvazuthi, I., Nallagownden, P., Ramasamy, G., Jangra, A. "Assessment of microgrid communication network performance for medium-scale ieee bus systems using multi-agent system" Springer Lecture Notes in Networks and Systems, 2021, 140, pp. 377–387.

Jangra, Ajay, and Neeraj Mangla. "Cloud Load Balancing Using Optimization Techniques." In Mobile Radio Communications and 5G Networks, pp. 735-744. Springer, Singapore, 2021.