Investigational Study of 7 Effective Schemes of Load Balancing in Cloud Computing

Suriya Begum
Research Scholar
Visvesvaraya Technical University
Belgaum, India

Dr. Prashanth C.S.R
Prof. and Head of Department
Dept. of Computer Science & Engg.
New Horizon College of Engg.
Bangalore, India

Abstract— **With the exponential increase in demands of online applications and services, cloud computing has evolved as a boon in modern information technology. Built over the base of grid and distributed computing, cloud computing offers services to cater the dynamic needs of massive user base. However, with the novelty associated with the system, cloud computing is also associated with certain issues like availability, cost, load balancing, security and performance. Very recently in last three years there has been abundant set of research work conducted aiming at mitigating the issues connected to load balancing in cloud computing. This paper discusses 7 efficient techniques that has been evolved in the past as a solution for load balancing issues in cloud computing.**

Keywords-component; Cloud Computing, Load balancing, Scheduling, Virtualization

I. INTRODUCTION

As the information technologies are growing day by day, the need of computing and storage are rapidly increasing. To invest more and more equipments is not an economic way for an organization to satisfy the even growing computational and storage need. Cloud computing [1] is a term, which involves virtualization, distributed computing, networking, software and web services. A cloud consists of several elements such as clients, datacenter and distributed servers. It includes fault tolerance, high availability, scalability, flexibility, reduced overhead for users, reduced cost of ownership, on demand services etc. In its most basic form, cloud balancing provides an organization with the ability to distribute application requests across any number of application deployments located in data centers and through cloud-computing providers. Cloud balancing takes a broader view of application delivery and applies specified thresholds and service level agreements (SLAs) [2] to every request. The use of cloud balancing can result in the majority of users being served by application deployments in the cloud providers' environments, even though the local application deployment or internal, private cloud might have more than enough capacity to serve that user. So Cloud Computing has become a widely accepted paradigm for high performance computing, because in Cloud Computing all type of IT facilities are provided to the users as a service. In Cloud Computing the term Cloud is used for the service provider, which holds all types of resources for storage, computing etc. Mainly three types of services models are provided by the cloud. First is Infrastructure as a Service (IaaS), which provides cloud users the infrastructure for various purposes like the storage system and computation resources. Second is Platform as a Service (PaaS), which provides the platform to the clients so that they can develop, and deploy their applications on this platform. Third is Software as a Service (SaaS), which provides the software to the users and hence the users don't need to install the software on their machines and they can use the software directly from the cloud. Cloud Computing provides many benefits: it results in cost savings because there is no need of initial installation of much resource; it provides scalability and flexibility, the users can increase or decrease the number of services as per requirement; maintenance cost is very less because all the resources are managed by the Cloud providers, basically our model is a step towards green computing. As cloud computing is in its evolving stage, so there are many problems prevalent in cloud computing [3]. Such as:

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

The most prevalent problem in Cloud computing is the problem of load balancing. Further, while balancing the load, certain types of information such as the number of jobs waiting in queue, job arrival rate, CPU processing rate, and so forth at each processor, as well as at neighboring processors, may be exchanged among the processors for improving the overall performance. The proposed paper will introduce a thorough analysis of the 7 efficient techniques that has evolved in cloud platform right from the origination of the initial distributed computing system. The paper will mainly focus on the research issues of load balancing and will attempt to analyze the prior work done in this field.

II. LOAD BALANCING IN CLOUD PLATFORM

Load balancing [4][5] 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. A load balancing protocol is dynamic in nature doesn't contemplate the previous state or behavior of the system, that is, it depends on the current behavior of the system. It is common these days in redundant high-availability computer systems that incoming network traffic is distributed on network level by deploying one of the frequently used network load balancing algorithms like:- random-allocation, round-robin allocation, weighted round-robin allocation, etc). These algorithms use solely network parameters of incoming traffic to create selections wherever to forward traffic, with none data from different elements of database system, like current load of application or info servers. Since these days it is extremely common to possess internet servers acting as application servers, it is usual that load balancers use session-switching technique, which suggests that once a user opens website on one server, it will stay on it server whereas the session lasts.

Depending on who initiated the process, load balancing algorithms can be of five categories:

- Sender Initiated: If the load balancing algorithm is initialized by the sender
- Receiver Initiated: If the load balancing algorithm is initiated by the receiver
- Symmetric: It is the combination of both sender initiated and receiver initiated
- Static: It doesn't depend on the current state of the system. Prior knowledge of the system is needed.
- Dynamic: Decisions on load balancing are based on current state of the system. No prior knowledge is needed. So it is better than static approach.

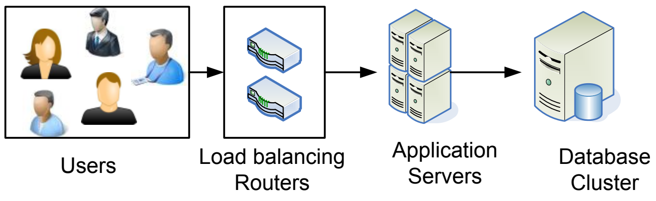


Figure 1 Schematics of typical high-availability computer system with hardware load balancers.

Central to the many other issues likes the establishment of an effective load balancing algorithm. The load can be CPU load, memory capacity, delay or network load. Load balancing is the process of distributing the load among various nodes of a distributed system to improve both resource utilization and job response time while also avoiding a situation where some of the nodes are heavily loaded while other nodes are idle or doing very little work. Load balancing ensures that all the processor in the system or every node in the network does approximately the equal amount of work at any instant of time.

This technique can be sender initiated, receiver initiated or symmetric type (combination of sender initiated and receiver initiated types).

Table 1 Metrics in existing LB techniques in cloud computing

LOAD BALANCING METRICS	
Metric	Illustration
Throughput	It is used to calculate the no. of tasks whose execution has been completed. It should be high to improve the performance of the system
Overhead	It determines the amount of overhead involved while implementing a load-balancing algorithm. It is composed of overhead due to movement of tasks, inter-processor and inter-process communication. This should be minimized so that a load balancing technique can work efficiently.
Fault Tolerance	It is the time to migrate the jobs or resources from one node to other. It should be minimized in order to enhance the performance of the system.
Response Time	It is the amount of time taken to respond by a particular load balancing algorithm in a distributed system. This parameter should be minimized.
Resource Utilization	It is used to check the utilization of re-sources. It should be optimized for an efficient load balancing.
Scalability	It is the ability of an algorithm to perform load balancing for a system with any finite number of nodes. This metric should be improved.
Performance	It is used to check the efficiency of the system. This has to be improved at a reasonable cost, e.g., reduce task response time while keeping acceptable delays

It is important to evaluate solutions for cloud balancing implementations with an eye toward support for the needs of an actual IT department. The global and local application delivery solution chosen to drive a cloud balancing implementation should be extensible, automated, and flexible, and the vendors involved need to look favorably upon standards. Meeting those criteria is paramount to ensuring the long-term success of a cloud balancing strategy. Combining high availability with security is just as important. When the organization is using a network that's not its own for mission-critical application delivery, stability and security become paramount.

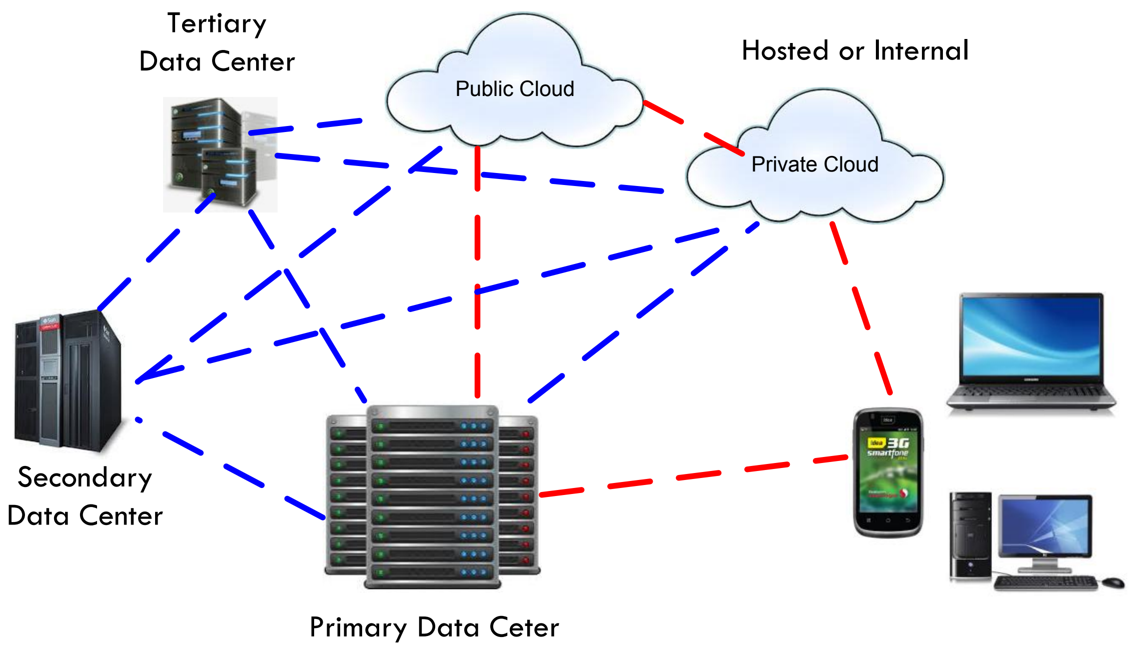


Figure 2: Automated cloud balancing

Cloud balancing is still new, but the technology to add value is available today. The ability to distribute connections across the globe based upon an array of inputs such as geographic location, device type, the state of servers in one location or another, and balanced loads is real. There will be no doubt that more advances in the future as cloud balancing will become more main stream. There are challenges associated with the implementation of such a strategy, some of which might take years to address. But the core capabilities of global and local application delivery solutions today make it possible to build a strong, flexible foundation that will enable organizations to meet current technical and business goals and to extend that foundation to include a more comprehensive cloud balancing strategy in the future. This review aims at summarizing the current state of the art of existing load balancing techniques in cloud computing. Here inspite of quantity of work done, the focus in given to only names of distinctive techniques used to mitigate load balancing issue in cloud computing), (load balancing techniques in cloud computing), (load balancing in clouds) and (load balancing in datacenters). Only papers written in English were included. Following section discusses about 7 load balancing techniques that are investigated in this paper.

III. Event Driven

In the recent the online game playing is much evolved. Day by day the online playing games are increasing like Avatar, Warcraft, and Counter Strike etc. For playing games through online the resource managements take the initiation and provide the game through servers for certain period of time, the time will be for some minutes for shooting game and then online billing occurs. In order to develop this kind of online gaming the dynamic resource management with load balancing is of quite essential.

Nae e.t al [6] evaluated a technique for dynamically resource provisioning in massively multiplayer online games (MMOG) for resource provisioning or for the load managing. The many millions of concurrent players can play the same game at a same time. Hence the dynamic resource management is very much essential for multi-player online games. They investigated the operational centers for provisioning on demand games and operational costs and they evaluated a neural technique for dynamic resource provisioning of MMOG entity distribution for better performance.

Initially the investigated various types of player interaction a source of short-term load variability, which complements the long-term load variability due to the size of the player population and then they introduced a combined MMOG processor, network, and memory load model that takes into account both the player interaction type and the population size. MMOGs are large-scale simulations of persistent game worlds comprising various objects or entities they classify into four categories:

- Avatars are in-game representation of the players.
- Bots or non-player characters (NPCs) are mobile entities that have the ability to act independently.
- Movable objects (such as boxes or guns) are passive entities which can be manipulated but do not initiate interactions.
- Immutable entities or decor.

The most employed model for online gaming is client\server model and it consist of each discrete time unit that to be executed. The clients dynamically connect to a joint game session and interact with each other by sending play actions such as movements, shootings, operations on game objects, or chat. To ensure scalability and real-time response, an MMOG session is distributed on multiple game servers, and each player is mapped to an avatar on one of the servers, usually to one in its closest proximity to minimize latencies. The entities that are hosted in distributed session are called active entities. The game session can be classified as parallelization techniques as

- zoning,
- replication, and
- instancing

Zoning is spatial scaling of game session, it partitions the game world into geographical areas to be handled independently by separate machines, replication targets parallelization of game sessions with a large density of players located and interacting within each other's area of interest, instancing is simplification of replication distributes session for high populated zones.

The MMOG depends on the game design that is on latency and tolerance. The proposed the analytical load model for MMOG by using the type of resource that they use CPU, memory, and network. The load models that they classified as

- CPU Load Model
- Memory Load Model
- Network Load Model
- Complete Load Model

In the CPU model they discussed w.r.t time consuming activities within one game tick, for memory model they formulated a equation w.r.t amount of memory needed to run actual game and game world being played, In network model they focused on outgoing network bandwidth usage for a machine running a server of a distributed game session and last complete model is integrating the all CPU, Memory, Network models.

Based on these models they proposed the neural network based prediction model for better performance and accuracy for provisioning dynamic MMOG. Their main goal is to reduce the prediction error.

To experiment and validate the neural network prediction, they developed a distributed game simulator, which realistically emulates the behavior of game players. The motivation for using an emulator is:

- they do not had available the exact coordinates of entities in the RunScape game and
- through this emulator, they are able to give further evidence that the player interaction determines the server load

The emulator used by them generates eight different data traces for duration of one day each with a sampling rate of two minutes, modeling four parameters: peak hours, peak load, overall dynamics, and instantaneous dynamics. The peak hours correspond to the periods with high player count in online gaming such as late afternoons. The peak load represents the highest load observed in an MMOG, which is a good measure for its relative popularity. The overall dynamic represents the variability of the entity interaction over a period of one day, while the instantaneous dynamic indicates the same variability over a period of two minutes.

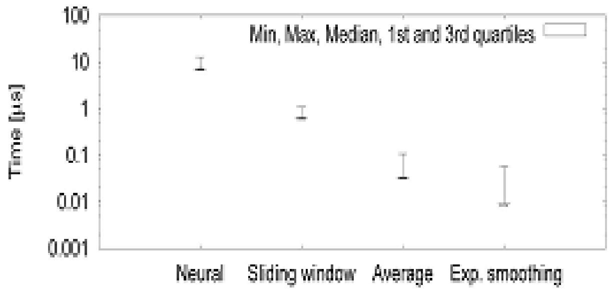


Figure 3 Statistical properties of the duration of one prediction for four prediction algorithms applied to MMOG data

Fig. 3 depicts the duration of one prediction on an Intel Core Duo E6700 (2.66 GHz) processor. Although the neural network predictor is the slowest with average prediction duration of approximately 7 microseconds, it is nevertheless fast enough and suitable to MMOGs. They evaluate each experiment by using three metrics: resource overallocation, resource underallocation, and number of significant underallocation events.

Table.2 Dynamic Resource Allocation Results

Predictor Type	Avg Over-Allocation[%]			Avg Under-Allocation[%]		
	CPU	ExtNet [in]	ExtNet [out]	CPU	ExtNet [in]	ExtNet [out]
Neural Network	25.90	995.27	66.04	-0.09	0	0
Average	32.41	1023.43	69.29	-12.84	0	-2.46
Last Value	25.11	989.10	65.36	-0.16	0	0
Moving Average	24.92	992.06	65.69	-0.33	0	-0.03
Sliding Window	24.97	992.73	65.76	-0.41	0	-0.03
Exponential Smoothing	24.76	977.85	64.11	-0.42	0	-0.03

They proposed a more efficient alternative based on the dynamic resource provisioning and management of data center resources and they made the thorough investigation of an MMOG ecosystem, that is, of a multi-MMOG, multidata center environment.

In this work they considered the number and the type of interactions between players, and between players and the environment is an important contributor to the game load. To address it, they have introduced a new MMOG model that focuses on the interaction count and type between game entities, shown that interaction leads to much more dynamic resource demands than previously believed, and proposed a novel prediction algorithm based on neural networks that is fast yet accurate. Their algorithm performed significantly better than the six-time predictors. They have further investigated the performance of the resource provisioning and management of data center resources with a large variety of scenarios that focus both on MMOG-specific properties and data center hosting policies. Most importantly, they have shown that the static resource provisioning can be, on average, from five upto 10 times more inefficient than dynamic allocation under the same conditions, and that the game operators can penalize the data centers with unsuitable hosting policies, by not using their resources. Finally, they have designed and implemented methods on top of the platform offered by the EUproject that show real time resource provisioning for a real game prototype.

IV. VectorDot

In present era with increasing scale and complexity of modern enterprise data centers, administrators are being forced to rethink the design of their data centers. In a traditional data center, application computation and application data are tied to specific servers and storage subsystems that are often over-provisioned to deal with workload surges and unexpected failures. Such configuration rigidity makes data centers expensive to maintain with wasted energy and floor space, low resource utilizations and significant management overheads.

Today, there is significant interest in developing more *agile* data centers, in which applications are loosely coupled to the underlying infrastructure and can easily share resources among themselves. Also desired is the ability to migrate an application from one set of resources to another in a non-disruptive manner. Such agility becomes key in modern cloud computing infrastructures that aim to efficiently share and manage extremely large data centers. One technology that is set to play an important role in this transformation is virtualization.

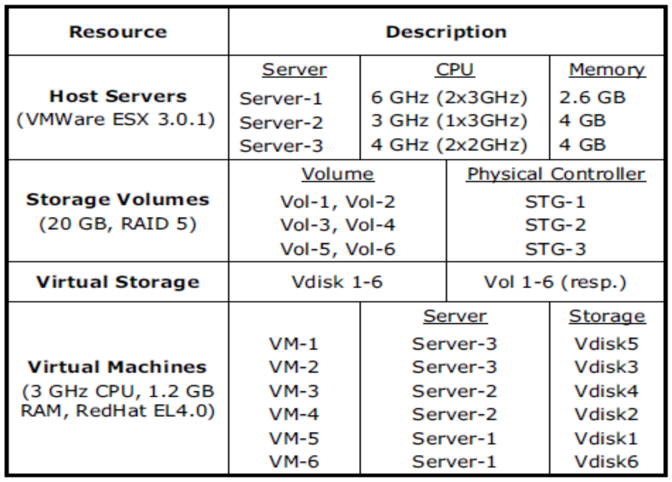
Storage virtualization technologies virtualized physical storage in the enterprise storage area network (SAN) into *virtual* disks that can then be used by applications. This layer of indirection between applications and physical storage allows storage consolidation across heterogeneous vendors and protocols, thus enabling applications to easily share heterogeneous storage resources. Storage virtualization also supports live migration of data in which a virtual disk can be migrated from one physical storage subsystem to another without any downtime.

Singh e.t al. [7] describes the design of an agile data center with integrated server and storage virtualization technologies. Such data centers form a key building block for new cloud computing architectures. They also show how to leverage this integrated agility for non-disruptive load balancing in data centers across multiple resource layers - servers, switches, and storage. They propose a novel load balancing algorithm called VectorDot for handling the hierarchical and multi-dimensional resource constraints in such systems. The algorithm, inspired by the successful Toyoda method for multi-dimensional knapsacks, is the first of its kind. They evaluate system on a range of synthetic and real data center test-beds comprising of VMware ESX servers, IBM SAN Volume Controller, Cisco and Brocade switches. Experiments under varied conditions demonstrate the end-to-end validity of our system and the ability of VectorDot to efficiently remove overloads on server, switch and storage nodes.

They use the Harmony test bed architecture for designing of the agile virtualization vectordot method and their algorithm describes to address hierarchical and multidimensional constraints that arise when deciding what items to move and to where inspired by toyoda heuristic method.The developed two algorithms called Extended vector product (EVP) for handling overloads and selecting destinations. The two algorithms that they used can be shown below.

```
Algorithm1 VectorDot: COMPUTING EVP
Step.1. EV P(Vitem vi, leafNode u) {
Step.2. if (vi already on u) then
    Step31.LV ec ← PathLoadF racV ec(u)
    Step.4.V V ec ← ItemPathLoadF racV ec(vi, u)
    Step.5.TV ec ← PathThresholdV ec(u)
    Step.6.return EV P2(LV ec, V V ec, TV ec)
else
    Step.7.LV ec ← AdjustedP athLoadF racV ec(vi, u)
    Step.8.V V ec ← ItemPathLoadF racV ec(vi, u)
    Step.9.TV ec ← PathThresholdV ec(u)
    Step.10.return EV P2(LV ec, V V ec, TV ec)
end if
}
Algorithm2 VectorDot: EVP2
Step.1.EV P2(LV ec, V V ec, TV ec) {
Step.2.Assert(LV ec.size() = V V ec.size())
Step.3.Assert(LV ec.size() = TV ec.size())
Step.4.val ← 0
Step.5.for i = 1. . . LV ec.size() do
    Step.6.val+ = V V ec[i]  Smooth(LV ec[i], TV ec[i])
end for
}
Step.7.Smooth(frac,T) {
return eα frac−T
}
```

For end-to-end validation of Harmony in a real data center setup, we created four scenarios in our testbed that cause overloads on multiple dimensions of servers, storage and switches. For the testbed experiments, they created six virtual machines (3 GHZ CPU, 1.24 GB RAM running RedHat Enterprise Linux 4.0) and distributed the equally between the three ESX servers. Each ESX Server has 1 HBA with 1 active port of 2GB I/O capacity and gigabit ethernet. They computed for number of scenarios for single server, multiple server and integrated server overloads. The results can be shown below.



Resource	Description		
Host Servers (VMWare ESX 3.0.1)	Server	CPU	Memory
	Server-1	6 GHz (2x3GHz)	2.6 GB
	Server-2	3 GHz (1x3GHz)	4 GB
	Server-3	4 GHz (2x2GHz)	4 GB
Storage Volumes (20 GB, RAID 5)	Volume	Physical Controller	
	Vol-1, Vol-2	STG-1	
	Vol-3, Vol-4	STG-2	
	Vol-5, Vol-6	STG-3	
Virtual Storage	Vdisk 1-6	Vol 1-6 (resp.)	
Virtual Machines (3 GHz CPU, 1.2 GB RAM, RedHat EL4.0)		Server	Storage
	VM-1	Server-3	Vdisk5
	VM-2	Server-3	Vdisk3
	VM-3	Server-2	Vdisk4
	VM-4	Server-2	Vdisk2
	VM-5	Server-1	Vdisk1
	VM-6	Server-1	Vdisk6

Figure 4 Test bed Resource Description

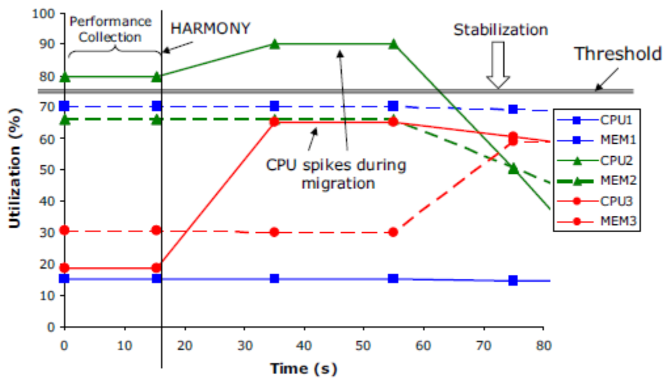


Figure 5 Single Server Overload Resolutions. Solid and dashed lines represent CPU and memory utilizations resp.

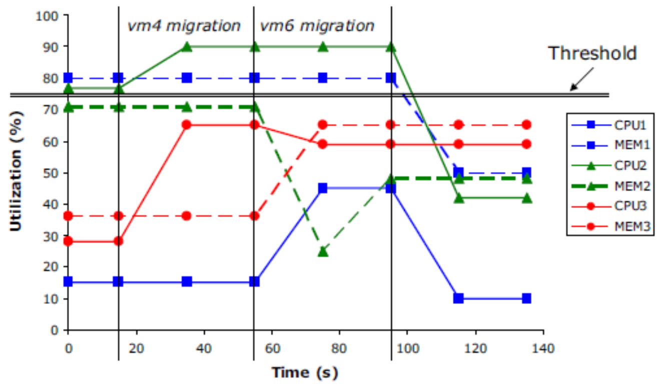


Figure 6. Multiple Server Overload Resolution. Solid and dashed lines represent CPU and memory utilizations resp.

	Servers (%) CPU, Mem	Storage % I/O
Initial Configuration	Server-1: 49.3, 82.4 Server-2: 62.7, 58.9 Server-3: 39.9, 47.5	STG-1: 77.8 STG-2: 9.8 STG-3: 54.9
After VM-5 Migration	Server-1: 33.3, 54.8 Server-2: 59.1, 58.3 Server-3: 67.3, 71.8	〃
After Vol-5 migration	〃	STG-1: 59.1 STG-2: 26.5 STG-3: 56.9

Figure 7 Integrated Servers and Storage Overload Resolution

Their evaluations show [Fig.5-6] on a range of synthetic and real data center testbeds demonstrate the validity of our system and the ability of *VectorDot* to effectively address the overloads on servers, switches, and storage nodes.

V. LBVS Technique

The author [8] has discussed about Storage Virtualization Model (SVM) that is proposed firstly to introduce the abstract storage virtualization model. In this model, virtualization layers are the key point. After that, Virtual Storage Architecture (VSA) is proposed to introduce the specific virtual storage architecture. This architecture is based on SVM, and Virtual Storage Management Layer achieves this abstract model.

The Fig. 8 shows the Storage Virtualization Model (SVM). Firstly, Storage Virtualization (SV) screens the differences of physical storage devices, supplies uniform admin interface and user interface, distributes and maps physical devices. During storage access, SV will route logical address to physical address. All access operations will be completed in a transparent mode. As illustrated, SVM contains three virtualization layers: resource virtualization, logical space virtualization, storage network virtualization.

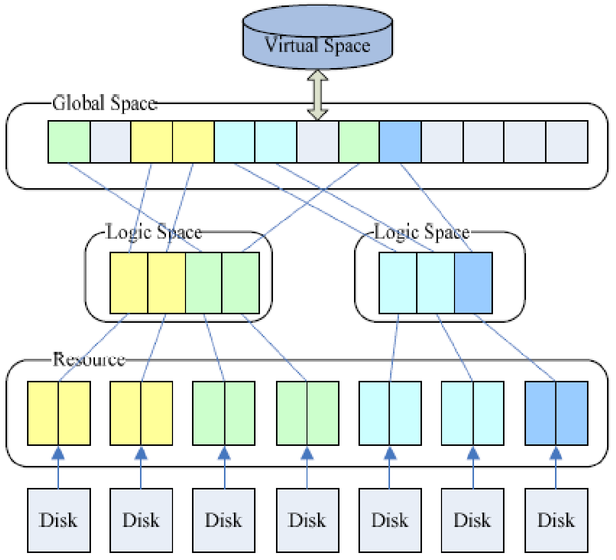


Figure 8 Storage Virtualization Model

In this work, LBVS uses the integrated Rule Oriented Data System technology (iRODS) that is from Data Intensive Cyber Environments (DICE) as the middleware to achieve virtual storage. The iRODS (integrated Rule Oriented Data System) technology is developed by the Data Intensive Cyber Environments (DICE) group, which is distributed between the University of North Carolina at Chapel Hill (UNC) and the University of California, San Diego (UCSD). It is software middleware that organizes distributed data into a shared collection. When data sets are distributed across multiple types of storage systems, across multiple administrative domains, across multiple institutions, and across multiple countries, data grid technology is needed to enforce uniform management properties on the assembled collection. The iRODS Data Grid expresses management policies as computer actionable Rules and management procedures as sets of remotely executable Micro-services. It contains three Logical Name Spaces (LNS) which are from the original SRB Data Grid: Logical names for users, Logical names for files and collections and Logical names for storage resources. The iRODS supports four types of virtualization.

- Workflow virtualization. This is the ability to manage the execution of a distributed workflow independently of the compute resources where the workflow components are executed. iRODS implements the concept of workflows through chaining of Microservices within nested Rule sets and using shared logical variables that control the workflow.
- Management Policy virtualization. This is the expression of Management Policies as Rules that can be implemented independently of the remote storage system. iRODS implements traditional ACID database properties (Atomicity, Consistency, Isolation and Durability).
- Service virtualization. The operations that are performed by Rule-based data management systems can be encapsulated in Micro-services. The iRODS Micro-services provide a compositional framework realized at run-time.
- Rule virtualization. This is a Logical Name Space that allows the Rules to be named, organized in sets, and versioned. A Logical Name Space for Rules enables the evolution of the Rules themselves.

iRODS has been used, replica balancing and writing balancing algorithms to build the virtual storage architecture, and the SVM model.

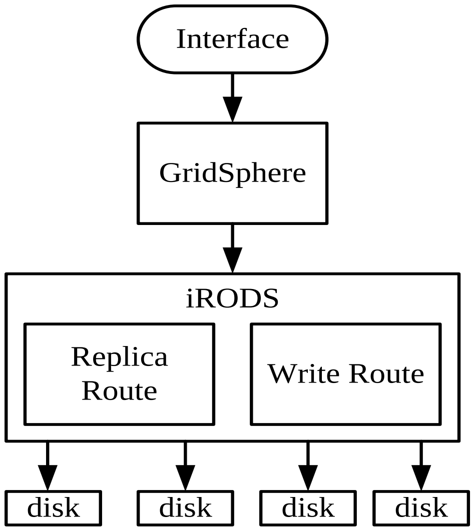


Figure 9 Implementation Model

Compared with the strategy of iRODS, LBVS can use the attribute of architecture to provide the best real time storage scheme. In iRODS, users need to choose the storage space that data write in manually. And in LBVS, system uses the specified parameters to decide the trend automatically. To deal with pressure of concurrent access, LBVS uses replica balancing. After that, the pressure of concurrent access and the response time decrease by a wide margin, and the capacity of disaster recover are enhanced. Compared with managing replica and migrating replica manually, this balancing strategy enhances the flexibility and robustness of system, and makes LBVS provide storage service much better.

VI. SERVER-BASED LB FOR INTERNET DISTRIBUTED SERVICES

In this work, the authors [9] have simulated scenarios where several clients, generating different workloads, access replicas of a web service distributed worldwide. In these simulations, the authors have assessed server selection policies that are representative examples of the two groups of solutions. With respect to this problem, an approach is presented for client-based server selection that adaptively assigns different selection probabilities to each server regarding network latencies and end-to-end response times.

In order to evaluate the solution, a simulator is designed using the CSIM for Java, a discrete event simulator framework. The author has used the PackMime Internet traffic model [10] to generate HTTP traffic in the simulations. PackMime allows the generation of both HTTP/1.0 and HTTP/1.1 traffic. PackMime has been obtained from a large-scale empirical study of real web traffic and has been implemented in the ns-2, a well known network simulator. In order to use the model in our simulations, a Java version of the PackMime is designed and implemented.

It is assumed that each geographically distributed replica of the web server is composed of a cluster of servers. Each server is simulated as a queueing system with fixed service time of 10ms. A scenario is considered with six replicas of the web server that are worldwide distributed: one in South America (S1), one in North America (N1), two in Europe (E1 and E2), and two in Asia (A1 and A2). The average of the latencies (ping RTT/2) measured on real hosts of PlanetLab3 in Brazil, USA, Belgium, Austria, Japan, and China is used to simulate the latencies among the replicated web servers. It is also considered that each replica serves a region and that the latency between a replica and a client of its region is 10ms.

In order to consider the latency of the TCP protocol, the analytic model proposed by Cardwell et al. [11] is adopted. This work extended previous models for TCP steady-state by deriving models for two other aspects that can dominate TCP latency: the connection establishment three-way handshake and the TCP slow start [RFC793 1981]. Therefore, the model proposed by Cardwell et al. can predict the performance of both short and long TCP flows under varying rates of packet loss. The accomplished solution (AD) is compared with two other server selection policies:

- Round Robin (RR): Each client sends requests to all servers in a rotative way;
- Best Server (BS): Each client uses RR to probe all servers. The server that presents the best mean response time is selected. Next, the client keeps sending all requests to the selected server until its mean response time exceeds the mean response time of other server. In this case, the client starts probing again, in order to avoid using out-of-date mean response times.

In order to present the flexibility of solution, the authors performed the experiments considering two scenarios, one that favors BS and another that favors RR. In the first, the total capacity of the servers was set to 1200 requests per second (rps) divided among the servers as follows: S1 = 100 rps, N1 = 300 rps, E1 = 200 rps, E2 = 300 rps, A1 = 200 rps, and A2 = 100 rps. The clients were configured to generate approximately 72% of the total capacity. In the second scenario, the total capacity was divided equitably among the servers and the aggregated load was set to approximately 90% of the total capacity. The parameters used in the heuristic are shown in Table 3.

Table 3 Parameters used in design

Parameter	Value	Description
INC	0.01	Probability increment
DEC	0.3Pi	Probability decrement.
t_UPDATE	1s	Time between probability updates
WSIZE	30 requests	Window size for response time slide mean.

While the first scenario is characterized by a lightly loaded system with heterogeneous servers, the second presents an almost saturated system with homogeneous servers. It is clear that, in the first case, due to its adaptability to server state changes, BS performed better than RR. In the second case, the equitable load distribution produced by RR outperformed BS's greedy strategy. Nevertheless, AD produced the best response times in both scenarios. This indicates that our solution successfully adapted to the system states while the other solutions did not. The results suggest that, in the considered scenarios, the considered hypothesis is valid.

A main advantage of client-side server selection policies is that clients can monitor end-to-end response times in a better way than server-side solutions. Besides, sometimes, client-side policies are the only option available. Most of the client-side policies proposed so far select one server to which the client should send all requests or equitably distribute the load among all of them. The simulations have shown that in scenarios where several clients use the same server selection policy, these two types of solution can lead to load-unbalanced states and, consequently, to the worsening of response times. In this work, the authors have argued that if clients collaborate in order to balance server load they can obtain better response times. The solution adaptively changes the fraction of load each client sends to each server giving higher priorities to nearby servers. Although this less greedy strategy of sending fractions of the load to worser servers seems to be counterintuitive, the experiments have shown that the solution overcomes the two types of policies proposed so far, even an in scenario that favors one type or another.

VII. Fuzzy Logic

Fuzzy Logic Approach [12] was considered for deployment over CloudSim with focus on designing a new load balancing algorithm based on round robin in Virtual Machine (VM) to achieve better response time and processing time. The load balancing algorithm is done before it reaches the processing servers the job is scheduled based on various parameters like processor speed and assigned load of Virtual Machine (VM) and etc. It maintains the information in each VM and numbers of request currently allocated to VM of the system. It identify the least loaded machine, when a request come to allocate and it identified the first one if there are more than one least loaded machine. Here, implementing the new load balancing technique based on Fuzzy logic is tried. Where the fuzzy logic is natural like language through which one can formulate their problem.

The advantages of fuzzy logic are easy to understand, flexible, tolerant of imprecise data and can model nonlinear functions of arbitrary complexity, and is used to approximate functions and can be used to model any continuous function or system. Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic and the mapping provides a basis from which decisions can be made, or patterns recognized.

In the investigation, the fuzzifier performs the fuzzification process that converts two types of input data like processor speed and assigned load of Virtual Machine (VM) and one output like balanced load which are needed in the inference system. In this work, considering the processor speed and load in virtual machine as two input parameters to make the better value to balance the load in cloud using fuzzy logic. These parameters are taking as inputs to the fuzzifier, which are used to measure the balanced load as the output.

The Defuzzification is the process of conversion of fuzzy output set into a single number and the method used for the defuzzification is smallest of minimum (SOM). The aggregate of a fuzzy set includes a range of output values and be defuzzified in order to resolve a single output value from the fuzzy set. Defuzzifier adopts the aggregated linguistic values from the inferred fuzzy control action and generates a non-fuzzy control output, which represents the balanced load adapted to load conditions. The defuzzification method is employed to compute the membership function for the aggregated output.

The algorithm is described below to maintain the load in VM of cloud computing as follows:

```
1 Begin
2       Connect_to_resource ()
3        L1
4        If (resource found)
5          Begin
6             Calculate connection_string ()
7             Select fuzzy_connection ()
8             Return resource to requester
9          End
10      Else
```

```
11          Begin
12              If (Anymore resource available)
13                  Choose_next_resource ()
14                  Go to L1
15              Else
16                    Exit
17          End
18 End
```

The proposed algorithm starts with request a connection to resource. It tests for availability of resource. It Calculate the connection strength if the resource found. Then select the connection, which is used to access the resource as per processor speed and load in virtual machine using fuzzy logic.

As the proposed load balancer Fuzzy based Round Robin (FRR) performs well, when comparing to the Round Robin Load Balancer by considering all instruction length per request.

Some experimental results of the performance increase in the implemented service by minimize the data centre processing time and overall response time is presented. Benefits of using fuzzy logic on round robin policy of load balancing are shown.

The network structure or topology also required to take into consideration, when creating the logical rules for the load balancer. Two parameters named as the processor speed and assigned load of Virtual Machine (VM) of the system are jointly used to evaluate the balanced load on data centers of cloud computing environment through fuzzy logic. The results obtained with performance evaluation can balance the load with decreases the processing time as well as improvement of overall response time, which are leads to maximum use of resources. So, the obtained result shows the proposed Load Balancing algorithms (FRR) perform better than Round Robin (RR) Load balancer and it can be more appropriate in real life application efficient and effectively.

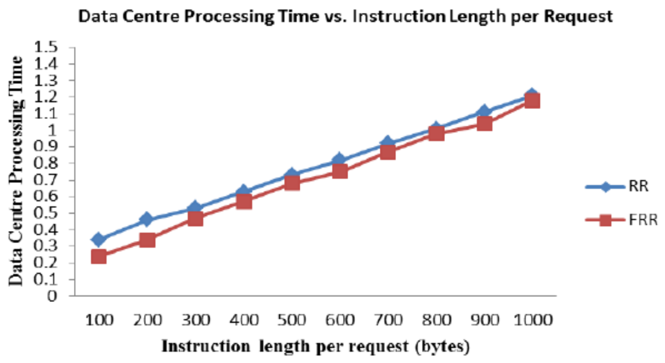


Fig-10: Data Centre Processing Time vs. Instruction Length per Request

In the result phase the main focus is to show the result, as the proposed load balancer Fuzzy based Round Robin (FRR) performs well, when comparing to the Round Robin Load Balancer by considering all instruction length per request. We have simulated the result by exploiting 25 machines, 5 numbers of processors per machine, and hundreds of jobs with the parameters mentioned as table-1. Some experimental results of the performance increase in the implemented service by minimizing the data centre processing time and overall response time. Benefits of using fuzzy logic on round robin policy of load balancing are shown.

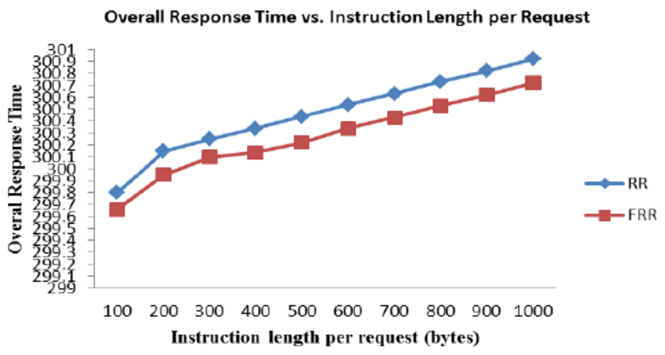


Figure 11 Overall Response Time vs. Instruction Length per Request

Fig.10 shows the data centre processing times are minimized with respect to all instruction length per request for Fuzzy based Round Robin (FRR) load balancer as compared to conventional Round Robin (RR) load balancer. We observed the efficiency of proposed load balancer FRR in terms of overall response time from Fig.11. It decreases the overall response time in all respect of data centre processing times as compared to RR. From these figure we observed that the FRR is better than RR, which is our objective.

VIII. Task Scheduling

Tayal [13] has proposed an optimized protocol based on Fuzzy-GA improvement that makes a programming call by evaluating the whole cluster of task within the job queue. The inspiration of our work is to give the centralized scheduler (master node) a choice by referring to a global view of the whole system. The framework of proposed model is shown in Figure 12. System Model describes the information related to processors which includes slot information, data replication information and workload information of processors. Task Model includes the job and asks information to be processed in the queue. Predicted Execution Time Model is a base for later schedule optimization.

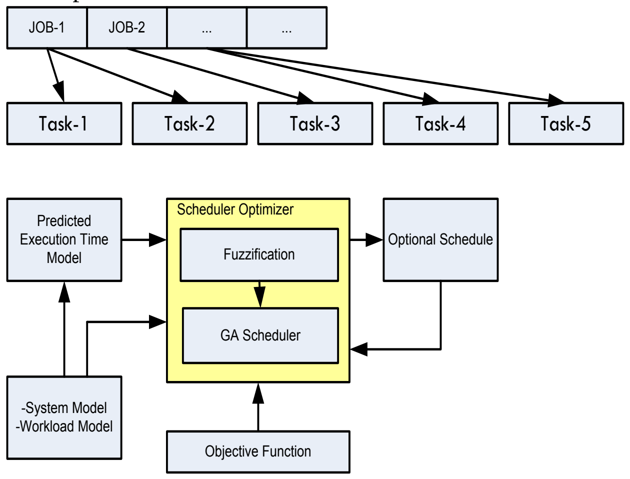


Figure 12 Schematic Diagram of Tayal [13]

It could be got by statistics techniques with tolerable deviation. Using the information of Task Model, System Model, Predicted Execution Time Model, Objective Function as input, to Fuzzification of parameter is implemented and gone through GA algorithm and generates an optimal schedule. When new jobs arrive or rescheduling condition is met, such as processor failure, Reschedule needs to be done.The system model describes the data store and computing cluster that jobs could be assigned to the cluster includes machines arranged in a general tree-shaped switched network as in Figure1. The nodes are commodity PCs. Data are distributed through these nodes. There are several replicas for each data block in the distributed file system. By default, the number of replicas is set as three in Hadoop. Map tasks generate the intermediate data stored the same node. We assume the communication overhead exits when the data does not locate in the same node as the computing node. The network rate between two nodes in the same rack is faster than the communication between nodes in different racks when network traffic on the main backbone network is big. Usually each rack contains 30-40 nodes. The links between racks are 1 Gbps while rack internal is 1 Gbps and local disk read is 2 Gbps. Each node can contain several processors. For each node, there are several map slots and reduce slots. Usually there is per slot for one processor.

The whole algorithm, considering the aspects discussed above, is shown below:

1. Get new tasks to be scheduled. The tasks to be scheduled include the uncompleted task and new jobs. But if jobs arrive in dynamically and make too many jobs waiting to be assigned at one time, the sliding-window technique [3] could be used as an option. The window size is fixed. Tasks fall into the sliding window will get scheduled.
2. Generating E matrix for the job Using KCCA technique to predict the execution time of any individual task assigned to every node.
3. Get the current state of the system.
4. Fuzzification of all above parameter to get optimized task schedule.
5. The Fuzzify parameter Map in GA to get optimized.
   5.1. Generate an initial population of chromosomes randomly.
   5.2. Evaluate the fitness of each chromosome in the population. Evaluate P according to information in E;
   5.3. Create a new population by repeating the following steps until the new population is complete, Selection Select two parent chromosomes from a population according to their fitness. (The better the fitness, the higher is the chance for getting selected). Crossover With a crossover probability, do cross over operations on the parents to form a new offspring. If no crossover is performed, offspring is the exact copy of the parents. Mutation With a mutation probability, mutate new offspring at each locus (Position in chromosome) Acceptance Place the new offspring in the new population.
   5.4. Using the newly generated population for a further sum of the algorithm.
   5.5 If the test condition is satisfied, stop and return the best solution in the current population.
   5.6. Repeat Step c until the target is met.
6. Finally obtain the optimal solution.

The task scheduling using Genetic Algorithm is done. The objective function for our algorithm is the latest completion time of the task schedule, referred as Makespan. The Makespan is calculated in objective function. Where represents the time that processor i will have finished the previously assigned jobs and E[t][i] is the predicted execution time that task t is processed on processor i. This paper also assume centralized scheduling scheme; i.e., a master processor unit in cloud, collecting all tasks, will take charge of dispatching them to other process units. Each process unit has its own dispatch queue (DQ). The master unit communicates with other process units through these dispatch queues. This organization ensures that processor units always find some tasks in the dispatch queue when the finish the execution of their current task. The master unit works in parallel with other units, scheduling the newly arrived tasks, and periodically updating the dispatch queues. Tasks are sorted ascending by the value of deadlines. Reasons to choose GA as an optimization algorithm is simplicity of operation and power of effect. It is suitable to some NP-hard problems

IX. Particle Swarm Optimization

Wu et al. [14] have experimented with a set of workflow applications by varying their data communication costs and computation costs according to a cloud price model. First, the algorithm starts with swarm initialization using greedy randomized adaptive search procedure to guarantee each particle in the initial swarm is a feasible and efficient solution. Then, compute the potential exemplars, pbest and gbest, for particles to learn from while they are moving. The stop condition is considered as the user's QoS requirements, such as deadline, the budget for computation cost or data transfer cost. The particle's new position generation procedure has three steps: 1) select elements from the promising set of pairs with larger probability, that is, the particle learns from gbest and pbest; 2) due to the discrete property of scheduling, there are usually not enough feasible pairs in gbest to generate new position, so the particle will learn from its previous position; 3) all the unmapped tasks should choose resources from other feasible pairs. Finally, gbest will be return as optimal solution.
Assume all tasks are executed on the Amazon Elastic

Compute Cloud (http://aws.amazon.com), all the data are stored in Amazon Simple Storage Service and data transmissions are fulfilled through the Amazon Cloud Front. And assume that Service 1 and 2 to be in US, Service 3 in Euro and Service 4 in APAC. Due to the varying price of service, in the following simulation, the price at this moment is adopted. Cost of execution of Ti on Service$_j$ is \$0.17 per hour (resources for high-CPU, on-demand instance medium instances, Linux Usage). *Taskcost* = *Tasktime* * *Price*. Data communication unit cost matrix is shown in Table 4. Each task has own input/output data and the sum of all data in the matrix varies according to the data size we test (64-2048M).

Table 4 Cost Matrix Used

	S_1	S_2	S_3	S_4
S_1	0	0.01	0.15	0.19
S_2	0.01	0	0.15	0.19
S_3	0.15	0.15	0	0.20
S_4	0.19	0.19	0.20	0

As for workflow, the number of total tasks ranges from 50 to 300 including both workflow and non-workflow activities. The number of workflow segments increase accordingly from 5 to 50. The number of resources is constrained in the range of 3 to 20. QoS constraints including time constraint and cost constraint for each task are defined as follows: time constraint is defined as the mean duration plus 1.28* *variance* and cost constraint is defined as the triple of the corresponding time constraint. The makespan of a workflow is defined as the latest finished time on all the virtual machines and the total cost of a workflow is defined as the sum of task durations multiply the prices of their allocated virtual machines..

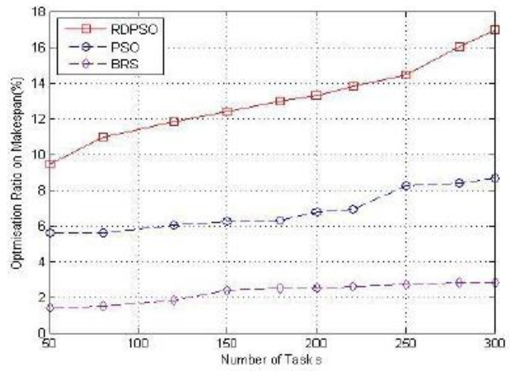


Figure 13 total makespan optimization ratio

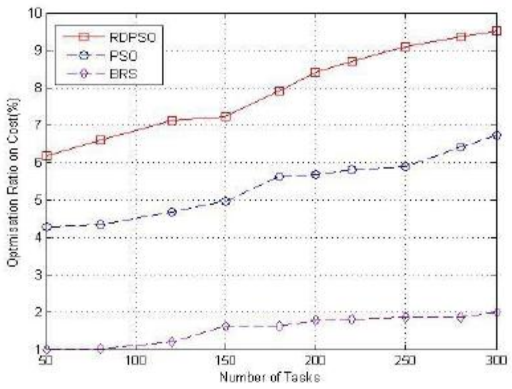


Figure 14. The total computation cost optimization ratio

From Fig 13, we can see that BRS can get around 2% optimization ratio, PSO can achieve from 6% to 8% optimization ratio, RDPSO can get from 10% to 17% optimization ratio on the whole makespan. PSO does not take makespan into account when it evolves; RDPSO takes not only computation cost but also whole makespan into account when it evolves. When user's requirement is specified, complete the workflow application with the requirement constraint is very important, so RDPSO is more applicable in cloud environment than PSO. From Fig 4, we can see that both PSO and RDPSO can achieve relatively large optimization ratio. These two algorithms take cost into account while they are searching the optimal solutions. BRS only blindly choose the best service. The authors have also compared the total computation cost optimization ratio by varying the tasks number. The result shows that when the task number of the workflow becomes large, their technique optimization ratio increases relatively dramatic. It means the technique can actually achieve lower cost for executing the workflow. Experimental results show that the proposed algorithm can achieve much more cost savings and better performance on makes pan and cost optimization. Result could be better if SLA was considered. The goal of this study was to determine whether the literature on load balancing techniques in cloud computing provides a uniform and rigorous base. The papers were initially obtained in a broad search in four databases covering relevant journals, conference and workshop proceedings. Then an extensive systematic selection process was carried out to identify papers describing load balancing techniques in cloud computing. The results presented here thus give a good picture of the existing load balancing techniques in cloud computing.

X. Conclusion

Load balancing is one of the main challenges in cloud computing [15]. It is required to distribute the dynamic local workload evenly across all the nodes to achieve a high user satisfaction and resource utilization ratio by making sure that every computing resource is distributed efficiently and fairly. So in this paper we have compared various algorithms of load balancing in Cloud Computing. And we have concluded that we can use a particular algorithm according to our requirement/need. But as we know that the Cloud Computing covers a very vast area, it is applicable to both small and large scale area but as we have concluded that none of the above algorithms satisfies the criteria. So there is a need to develop an adaptive algorithm which is suitable for heterogeneous environment and should also reduce the cost.

REFERENCES

[1] Peter Mell, Timothy Grance, “The NIST Definition of Cloud Computing”, *Special Publication 800-145, 2011.*

[2] Rajkumar Buyya1, Saurabh Kumar Garg, and Rodrigo N. Calheiros, “SLA-Oriented Resource Provisioning for Cloud Computing: Challenges, Architecture, and Solutions”, *International Conference on Cloud and Service Computing, IEEE, 2011*

[3] Nayandeep Sran, Navdeep Kaur, “Comparative Analysis of Existing Load Balancing Techniques in Cloud Computing”, *International Journal of Engineering Science Invention, ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726, Vol. 2 Issue 1, pp.60-63, 2013*

[4] Jaspreet kaur, “Comparison of load balancing algorithms in a Cloud”, *International Journal of Engineering Research and Applications, Vol. 2, Issue 3, pp.1169-1173, 2012*

[5] K. Ramana, A. Subramanyam and A. Ananda Rao, “Comparative Analysis of Distributed Web Server System Load Balancing Algorithms Using Qualitative Parameters”, *VSRD-IJCSIT, Vol. 1 (8), pp.592-600, 2011*

[6] Vlad Nae, Alexandru Iosup, Member, Radu Prodan, “Dynamic Resource Provisioning in Massively Multiplayer Online Games”, *Parallel and Distributed Systems, IEEE Transactions on* , Vol.22, No.3, pp..380,395, 2011

[7] Aameek Singh, Madhukar Korupolu, Dushmanta Mohapatra, “Server-Storage Virtualization: Integration and Load Balancing in Data Centers, *High Performance Computing, Networking, Storage and Analysis, SC. International Conference for* , Vol., pp.1-12, , 2008

[8] Hao Liu, Shijun Liu, Xiangxu Meng, Chengwei Yang, Yong Zhang, “LBVS:A Load Balancing Strategy for Virtual Storage”, International Conference on Service Sciences, *Service Sciences (ICSS), International Conference on* , pp.257,262, 2010

[9] Alan Massaru Nakai1, Edmundo Madeira1, and Luiz E. Buzato, “Improving the QoS ofWeb Services via Client-Based Load Distribution”, 2011

[10] Jin Cao, William S. Cleveland, Yuan Gao, Kevin Jeffay, F. Donelson Smith, Michele Weigle, “Stochastic Models for Generating Synthetic HTTP Source Traffic”, *INFOCOM, Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, Vol.3, pp.1546-1557 vol.3, 2004

[11] Cardwell, N., Savage, S., Anderson, T. “Modeling tcp latency. *In INFOCOM,. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Vol. 3, pp. 1742 – 1751, 2000*

[12] Srinivas Sethi, Anupama Sahu, Suvendu Kumar Jena, “Efficient load Balancing in Cloud Computing using Fuzzy Logic”, *IOSR Journal of Engineering (IOSRJEN) ISSN: 2250-3021 Vol. 2, Issue 7, pp. 65-71, 2012*

[13] Tayal, S., “Task Scheduling Optimization for the Cloud Computing System”, *International journal of advanced engineering sciences and technologies, Vol. 5, No. 2, pp.111 – 115, 2011*

[14] Zhangjun Wu1, Zhiwei Ni, Lichuan Gu, Xiao Liu, “A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling”, *Computational Intelligence and Security (CIS), 2010 International Conference on* , pp.184-188, 2010

[15] Begum, Suriya, and C. S. R. Prashanth. "Review of Load Balancing in Cloud Computing", *IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 1, No 2, 2013.*

Suriya Begum has completed her Bachelor of Engineering in Computer Science and Engg in 1995 from Bangalore University, India. She completed her Master of Technology in Computer Science in 2007 from Allahabad University, India and currently a research scholar in Visveswaraya Technical University, Belgaum, India. She is having almost 19 years of experience as an academician. Her research interest is cloud computing, networking and load balancing.

Dr. Prashanth C.S.R has completed Bachelor of Engineering in Computer Science and Engg from Bangalore University, India. M.S in Computer Science from University of Texas at Dallas and Completed his PhD in Computer Science, from Auburn University in 2006.His research interest is high performance computing, cloud computing, networks and Operating Systems. He is presently working as Professor and Head of Department of Computer Science and Engg, New Horizon College of Engineering, Bangalore. He has published many national and international papers.