Load Balancer as a Service in Cloud Computing

Mazedur Rahman
Member IEEE
mazedur.rahman.liton@gmail.com

Samira Iqbal
Member IEEE
erasamira10@gmail.com

Jerry Gao
San Jose State University, USA
jerry.gao@sjsu.edu

Abstract – The explosive growth of cloud computing in recent years has led to a massive increase in both the amount of traffic and the number of service requests to cloud servers. This growth trend of load poses serious challenges to the cloud load balancer in efficient balancing of the load, already a daunting job. The cloud load balancing is a highly researched field where numerous solutions to balance load have been proposed. Unfortunately, no research papers provided a comprehensive review focusing Load Balancer as a Service (LBaaS) model. In this paper, we first understand the concepts of load balancing, its importance and desired characteristics in cloud. Then we provide complete review on the existing load balancing strategies, their strength, shortcomings and a comparative study. Finally, we presented load balancer as a service model adopted by the major market players, and our observation, future needs and challenges.

Keywords – **load balancer as a service; cloud computing; load balancing; network traffic manager;**

1. Introduction

In cloud ecosystem, application service providers or software vendors can focus solely on building their software, instead of managing server infrastructure and platform themselves. Cloud computing takes away this burden from software vendors by providing convenient and cost effective mechanism of deploying the software. Modern software vendors develop their software on cloud environment, deploy them on a number of rented server instances and deliver the software as a service. While these rented server instances give them instant delivery mechanism for their software, inefficient load distribution among these server instances may result in poor performance for their delivered software. Hence the load balancer comes into the picture.

Cloud computing is a federation of many resources interacting together by sharing and pooling resources efficiently. As the demand of cloud computing increases, the complexity of the distributed systems involved keeps increasing. As this deals with large number of heterogeneous systems, effective load balancing techniques becomes imperative and important. Sometimes many systems can flood the resources of a target web server with many requests all at once. When a server is overloaded with connection, new connections cannot be handled. This is a case for the Distributed Denial of service attack. This is one scenario where efficient load balancing techniques can be useful. In order to minimize downtime and delays for the requests coming to a server or many servers instances, cloud providers also provide load balancer as a service (LBaaS) to their clients. These load balancers greatly increase performance with optimal utilization of the server instances, and no server in the cluster is overwhelmed.

This paper is organized as follows. Section 2 provides the concept of load balancing, its importance, common load balancing strategies, solutions and algorithms in cloud. Section 3 presents the concept of LBaaS model, its main characteristics and a comparative study among the major LBaaS providers. Section 4 identifies the needs and challenges for LBaaS. Finally, the conclusion remarks are given in Section 5.

2. Load Balancing in Cloud Computing

2.1. Basic Concept of Cloud Load Balancing

Load balancing is a technique used to distribute the incoming traffic among available servers so that the requests can be handled and the response given, at a faster rate. Fig. 1 depicts a classic load balancing architecture used in cloud environment where load balancer balances the load using the following typical steps:

- Receives incoming service requests from various clients
- Calculates requested load size of the incoming load request from clients and builds a request queue
- Checks the current load status of the servers in the server pool periodically using a server monitor daemon
- Uses a load balancing strategy/algorithm/heuristic to select appropriate server

The complex computing network systems does the routing of millions of data packets every second. Such large data traffic should be distributed efficiently among the available servers so that the servers can handle it gracefully without any impact on end user experience. There are dedicated servers allocated to handle this load balancing across the servers. Providing high availability is one of the driving factors of load balancing. Major advantages of load balancing are listed below:

- Helps in controlling and tracking traffic.
- Increases resource utilization
- Balances the network load based on node capability
- Improves resource availability
- Reduces overprovision of infrastructure

The objective of cloud computing architectures is to provide on demand services or rapid elasticity or in other words expand or contract as per demand. This means whatever be the application they will have to scale at one point to meet the demands of the user. There has to be a device that distributes the requests across multiple instances of the application. This is served with the help of a load balancer.

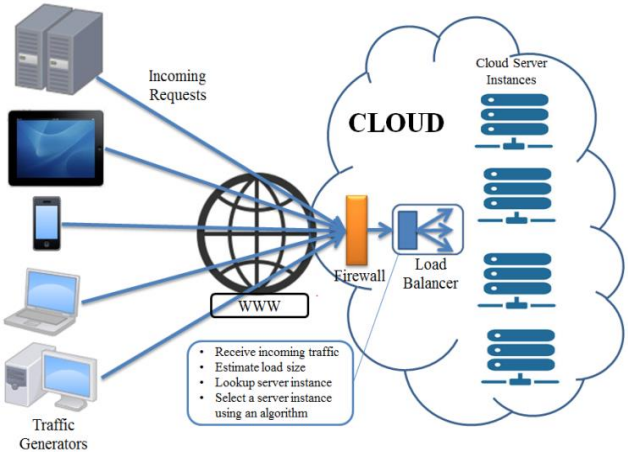

Figure 1. Load Balancing in Cloud

Suppose we consider load in terms of CPU time and assume that we have server A that is not able to allocate CPU time due to other virtual machines that are executing in parallel. Now if there is a Server B sitting idle, we can redistribute the load from A to B by migrating some guests from A to server B, thus reducing server response time and providing high throughput. Therefore load balancing on the cloud should possess the following characteristics:

- No traffic overhead at the load balancer side and maintain low overhead on the resources
- Up-to-date load information of systems that are participating

978-1-4799-3616-8/14 $31.00 © 2014 IEEE
DOI 10.1109/SOSE.2014.31

- Maintains balance of the system and takes action periodically
- Can be centralized or decentralized
- The server that currently serving has sufficient resources
- Migration should take minimal time
- Reliable network communication

2.2. Common Load Balancing Strategies

The most popular and primitive strategies, used by many load balancing algorithms in Section 2.3, are presented here. A comparison among these common strategies is given in Table I.

- *Least-Connection Load Balancing*

Least-connection strategy focuses on balancing number of active requests for the servers in a cloud computing environment. In this method, whenever new service requests arrive, load balancer selects the server with least number of active connections. It is a dynamic scheduling algorithm as it counts the number of active connections for each server to find its load. Fig. 2 shows 3 servers with active requests where server-3 will be selected by the load balancer for the next three service requests. The fourth service request will be served by server-1, fifth by server-3, sixth by server-1, and so on with a goal to balance the number of HTTP connections to the available server pool [1, 3].

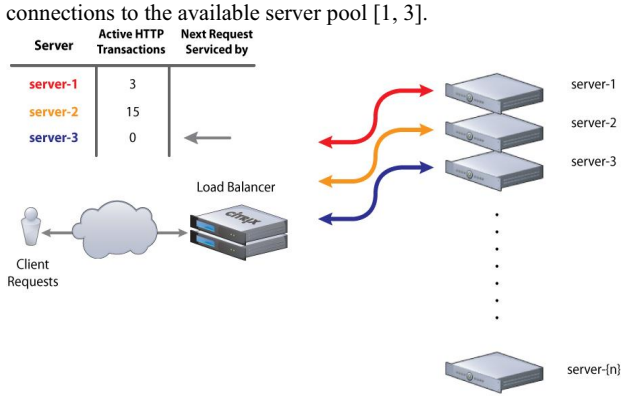

Figure 2. Least-connection load balancing strategy [1]

- *Weighted Least-Connection Load Balancing*

Weighted least-connection is an extension of the least-connection approach where every server is given a weight. The default weight of a server is 1 and a server with weight 0 will not receive any active connection. Likewise least-connections, server with least active connections will receive the next request [2].

- *Round-Robin Load Balancing*

In round-robin strategy, as the name suggests, the servers receives service requests in a round-robin or circular fashion. If there are 7 servers in a server-pool, server-1 will serve first, then server-2, then server-3, and so on until server-7, shown in Fig. 3. Server-1 will again receive service requests after server-7's turn is over. It doesn't consider various factors such as – number of incoming connections, response time, request processing capability, etc., instead treats all the servers as equal. It is simple and easy to implement [4, 5].

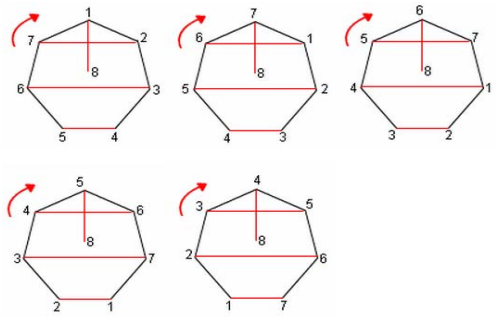

Figure 3. Round-robin scheduling strategy

- *Weighted Round-Robin Load Balancing*

Weighted round-robin is an extension of round-robin strategy where servers are given different weight numbers according to their capability. The servers receive service requests in the order of higher weighted server to lower weighted server. As the capability of the servers is considered in this strategy, it will provide better performance than the round-robin strategy [6].

TABLE I. **A COMPARATIVE STUDY AMONG COMMON LOAD BALANCING STRATEGIES**

Technique	Strategy	Nature	Property consideration			
			Server load info	Server capability	Response time	Active connections
Least-connection	The server with the least number of active connections receives service request	Dynamic	No	No	No	Yes
Weighted least-connection	In addition to active connections, each server is given a weight either 0 or 1 and server with weight 0 won't receive connection	Dynamic	No	Yes	No	Yes
Round-robin	Servers receives service request in a circular fashion irrelevant of the server capability	Static	No	No	No	No
Weighted round-robin	Servers are weighted based on their capability and requests are received in the order of higher to lower weighted server	Static	No	Yes	No	No
Dynamic feedback	Uses a monitor daemon to check various parameters such as availability, load and response time to balance load	Dynamic	Yes	Yes	Yes	Yes

- *Dynamic Feedback Load Balancing Scheduling*

In real life servers system, there are some situations when already overloaded servers keep receiving more service requests although some other servers are still available. Dynamic feedback load balancing strategy avoids this situation by considering availability, load information and response time of each server. It uses *monitor daemon* that gathers availability and load information of the servers periodically. This technique provides better performance where the load varies a lot [7].

2.3. Load Balancing Solutions and Classification

The existing load balancing solutions can be grouped into six categories as shown in Fig. 4. These are discussed in more detail in the following sections.

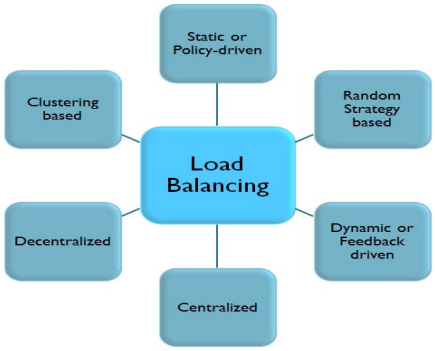

Figure 4. Classification of load balancing solutions

2.3.1. Static Load Balancing

Static load balancing, or policy driven load balancing, uses a predefined set of rules and policies. These rules and policies are defined by various parameters, such as - server capability, availability, response time, resource utilization, and fault tolerance. Usually thresholds, maximum/minimum limit, and other such constraints are predefined to make load balancing decisions.

To optimize throughput, cloud computing environments have three conditions for load balancing: ability to reorganize exchanged information efficiently, usability in heterogeneous environments, and keeping average system response constant. To fulfill these conditions Yao [8] proposed an algorithm that calculates the resources of requests, records the profit into the database and looks for the best server match. This approach can deal with only lightly loaded nodes and has connectivity problems with the server. An improved colony mechanism is used to address these issues by analyzing cloud resources using following steps:

1) If server queue length is longer than the predicted threshold value, the current request (R_1) will contact random unique request (R_2) in an adjacent server.
2) R_2 is put into a new server, and the database updates and removed from the original queue.

The improved algorithm performs well, but fails to utilize increased system resources efficiently. Shu-Ching Wang [9] proposes a two-phase scheduling algorithm using two different algorithms in one model: Opportunistic Load Balancing Algorithm (OLB), and Load Balance Min Min Algorithm (LBMM). The general OLB algorithm assigns each task arbitrarily to the machine expected to be next available, regardless of execution time, and results in poor optimization. LBMM algorithm considers tasks with regard to completion times. The task with the least minimum completion time is selected to the machine. The process is updated and repeats. OLB and LBMM are combined to sort out the most effective service nodes: *request manager* (N_0) collects the information about the tasks that are to be executed; *OLB* distributes the tasks to different service mangers under the request manager; finally after verifying "threshold of service node", *LBMM* algorithm puts a task with minimum load an effective service node.

2.3.2. Dynamic Load Balancing

Unlike static load balancing, dynamic load balancing (also known as feedback-driven load balancing) can scale the number of servers while distributing load based on current traffic [18]. Zhang [10] offered a load balancing mechanism based on ant colony and complex network theory in Open Cloud Computing Federation (OCCF). The proposed theory uses each *control center* to be a Regional Load Balancer Node (RLBN) to direct loads over nodes:

1) If any RLBN has load under or over the threshold then load balancer equilibrates the load over two nodes
2) A table that maintains edges adjacent to current node is then updated
3) If all the edges meet then the table is updated and next edge is traversed from another connected RLBN

In the last six years the number of subscribers to Massively Multiplayer Online Games (MMOG) grew from 10 thousand to 6.7 million. Millions of players are playing together, making it a computationally intensive client-server application with strict Quality of Service (QoS) requirements. To counter overprovision of resources, Nae [11] presents a new solution to hosting MMOG sessions on cloud servers that will allot on-demand provisions based on the number of active players while satisfying all QoS requirements. The following predictive cycle is proposed: *Monitoring,* which collects online metrics; *Load Prediction and Capacity Planning* which predicts the main use of resources and potential hotspots for the game servers; *Resource Allocation* which determines the amount of resources used; and *Load Balancing* which uses a balancing algorithm to adjust the load.

The simulation experiments using bases from real MMOG's show that the hosting costs can be reduced by a factor of 2 and 5 and the algorithm can adjust the number of game servers and distribution while keeping the QoS in 99.34%.

2.3.3. Random Strategy based Load Balancing

For a load balancing system, it is not efficient to overprovision servers for the sake of just meeting all demands in complex computing systems. Any algorithm for assigning demands to servers must incorporate uncertainty of the system, which Randles [12] proposes to do by a biased random sampling algorithm where a network is made with virtual nodes to represent each server, including resource availability per node. Then a number of inward edges are made per node assuming that the request input flow is the same as the output. The allocating node and executing node in-edges create a directed graph. The selected server is chosen via *random sampling*.

2.3.4. Centralized Load Balancing

The need for cloud computing systems that can run multiple instances in different operating systems has greatly increased. However with limited servers and simultaneous jobs, the idea Central Load Balancing for Virtual Machines (CLBVM) comes into play. CLVBM distributes the load uniformly across servers in a distributed environment. Bhandani and Sanjay [13] propose the following approach: Unique identifications for different VMs are assigned while collecting of the CPU load. The gathered data is sorted into Heavy (H), Moderate (M) or Light (L) load categories. Then a master server uses the information and balances the H and L loads first.

Load-balancing services must handle information that needs to be stored in multiple requests in one session. To do this, requests need to be sent repeatedly to the same backend server. Many existing solutions use shared memory and locks but Liu suggested a lock free solution [14]. A part of the solution is "persistence" which uses a session table integrated into the process to map the request to the server within the proposed solution:

1) In the IP component, the data is sent to the INET socket layer where kernel matches the socket structure to the data address

2) Kernel constructs a new connection and sends the requests from the same session to the same load balancing process

TABLE II. A COMPARATIVE STUDY AMONG VARIOUS LOAD BALANCING ALGORITHMS

	Technique	Strength	Limitations
Static	*Artificial Bee Colony Search*	• Keeps response time constant • Improved utilization up to a certain no. of resources • Simple and flexible, uses fewer control parameters • Increases throughput and efficiency of overall system	• Based on the number of requests, different algorithms must be used • Efficiency drops with system resources increase • For a marketing level needs scalability
	Two Phase Scheduling	• Improves resource utilization • Minimizes completion times • Increases overall performance and efficiency	• Fails to improve throughput • Overall response time is not improved
Dynamic	*Artificial Ant Colony Search*	• Greatly reduces overhead • High performance, resource utilization and scalability • Has good fault tolerance • Adaptive in heterogeneous environments • Excellent scalability	• Each iteration has a different probability of load distribution • Uncertainty in both maximum execution time and in idle time • Has very poor response time
	Event Driven	• Excellent scalability for commercial use • Good resource utilization because of component analysis • Real-time resource scale up or scale down • Keeps QoS 99.34%	• Has little to no fault tolerance • Does not improve response time, performance, or throughput
Random	*Biased Random Sampling*	• Improvement in performance with high and similar population of resources • Has excellent scalability for marketing resources • Simple and flexible, can easily be changed to fit certain parameters	• Loses performance ability if populations of resources are varied from each other • Can be more effective by biasing conditions based on predefined conditions • No impact on response time, resource utilization or fault tolerance
Centralized	*Central LB*	• Balances the load evenly • Improves overall performance • Has excellent response time and resource utilization	• Has little to none impact on fault tolerance • Lack of scalability makes it unsuitable for high market applications
	Lock Free	• Improves overall performance • Runs multiple load balancing processes with one load balancer making it simple	• If the one of the levels in the algorithm fails at assigning the requests evenly, the load balancing ability is heavily decreased
De-centralized	*Decentralized Content Aware LB*	• Improves performance by improving the searching time • Reduces idle time at each of the nodes • Good scalability	• No improvement in terms of throughput or fault tolerance
Clustering	*Active clustering*	• Performs well with high amount of system resources, and uses those resources to have increased throughput • Has high scalability	• Requires a high amount of system resources • Efficiency greatly decreases as the population diversity increases

The solution modifies the Linux system by adding a new socket section (sock level hash) for the listening socket, which allows it to be lock free. In the sock level hash, the Toeplitz matrix is used (provided by Microsoft) and shows that it can assign the requests evenly to each process.

2.3.5. Decentralized Load Balancing

Over the past years distributed computing has become much more advanced and there are many types of environments that are available for large scale applications. However scheduling applications in these environments still has a lot of issues, such as the decision process for allocating resources. Mehta [15] introduces content aware load balancing strategy as a solution. Content aware load balancing uses the content requested to schedule a specific request. The authors propose a new content aware load balancing algorithm called workload and client aware policy (WCAP) for a distributed computing environment (DCE).

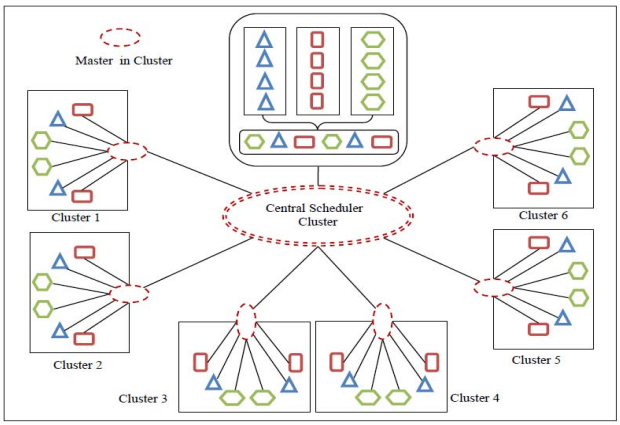

Figure 5. Decentralized content-aware load balancing [15]

For DCE's that require large resources, the algorithm needs to be modified to compensate. The solution is clusters of master servers in the DCE. As in Fig. 5, the master server cluster receives the request. Then the request is sent to the master server with the least load. That master server with the request decides if the request matches the category that its servers handle, if not the request is forwarded to the next master server. This process repeats until the best server is found.

2.3.6. Clustering based Load Balancing

To keep up with load balancing demands with the complexity of newer computing systems a new system must be able to balance nodes that process different jobs and localized knowledge. One of the solutions to this is the usage of autonomic self-aggregation and clustering techniques that change system of heterogeneous nodes in groups of homogeneous that can balance the nodes with previous techniques. Nitto [16] discussed the benefits of active clustering and self-aggregation, where nodes restructure themselves or self-organize to add new links to nodes that are compatible within the system, as in Fig. 6.

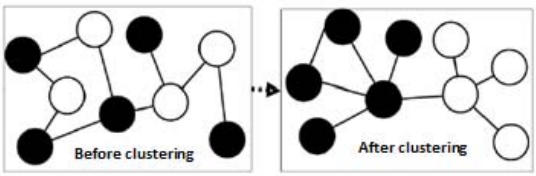

Figure 6. Clustering on a set of nodes [16]

The clustering algorithm is not done by a central entity; rather it is executed by each and every node with its ability to decide on "maintain/disconnect the link" from its neighbors. The nodes perform iterative execution of the following steps. A random node elects itself as an initiator node and then elects a matchmaker node. The matchmaker node selects a neighbor node that is compatible with the initiator node and enables the two similar nodes to establish a link, and then disconnects itself. Here we see that the group of nodes that are homogenous rewired in a heterogeneous system which is then able to distribute the load. Adaptive Clustering algorithm is used along with Load balancing algorithm and the network is always active here. This algorithm goes in conjunction with the Dimension Exchange Algorithm.

2.3.7. Comparative Study among Different Solutions

The important features of load balancing solutions are scalability, response time, resource utilization, performance, fault tolerance, and migration time [17]. Based on these features, a comparative study among the popular available approaches is provided in Table II.

3. Load Balancer as a Service in Commercial Cloud

3.1. Load Balancer as a Service

Traditionally load balancers have been delivered to the client as a hardware product for balancing network traffic similar to typical computing hardware devices. The evolution of service oriented models such as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a service has given the software vendors the opportunity to concentrate solely to their core capability i.e. the application development by utilizing the power of cloud. Thus instead of building data centers and infrastructure to deliver their software product, more and more software vendors started moving their development and deployment infrastructure to the cloud environment. They rent virtual cloud server instances and deploy their software on these virtual machines (VM). Thus traffic received by these requires efficient management so that some server instances won't be overwhelmed while some others will be sitting idle. In order to facilitate the load distribution among the rented server instances by a cloud tenant, the cloud provider has developed a business model where a load balancer an also be rented to the tenants similar to SaaS, PaaS and IaaS.

The typical load balancer as a service (LBaaS) business model uses the following steps to provide their LBaaS to their clients, as shown in Fig. 7:

1. *Create an account with SLA*: In this step, customers create an account with their preferred LBaaS provider. They provide all the necessary account information required for billing and also signs the service level agreement (SLA).
2. *Select preferred billing package*: Customers need to select a preferred billing option. Usually billing model is pay as you go where bill is calculated based on per minute or per hour LB usage and amount of data processed by the load balancer.

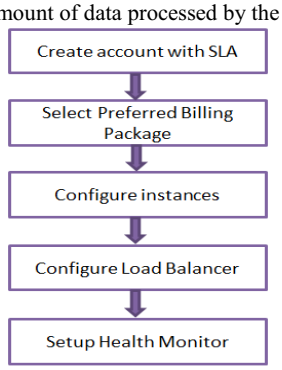

Figure 7. Typical LBaaS business model

3. *Configure server instances*: In this step, customers create and configure appropriate number of server instances needed for their business.
4. *Configure load balancers*: In this step, customers create and configure appropriate number of load balancers needed to balance the traffic for the server instances.
5. *Setup health monitoring*: Finally, customers set up the conditions for monitoring the health of their load balancer periodically.

3.2. Main Features of Load Balancer as a Service

Conventional load balancers focuses on core networking solutions and balance the incoming network traffic based on the knowledge of OSI layer 2 and 3 [27]. But the load balancers available as a software service in modern cloud computing environment are capable of balancing traffic based on the information present in OSI layer 4 – 7, as in Fig. 8-a. The most important features of an LBaaS are listed below (see Fig. 8-b):

- ***Algorithms availability and configurability***: Load balancing algorithms are the brain of load balancers and availability of these algorithms are very important in determining the effectiveness of the load balancers. Higher number of load balancers in LBaaS model also gives flexibility to the customers to be more suitable for their unique need.
- ***Communication interface***: Communication interface is another important feature that provides the LBaaS to be more fitted for diverse application integration. It gives the LBaaS provider the agility to be appropriate for different businesses. Popular communication protocols and APIs such as HTTP, HTTPS, SSL, TCP, UDP, LDAP, SMTP, and REST.

- ***Usage tracking****:* Similar to all other software as a service model, LBaaS also require very accurate and reliable usage tracking mechanism so that clients can analyze their usage and make decision on when to scale up or scale down the number of load balancers to fulfill their need.

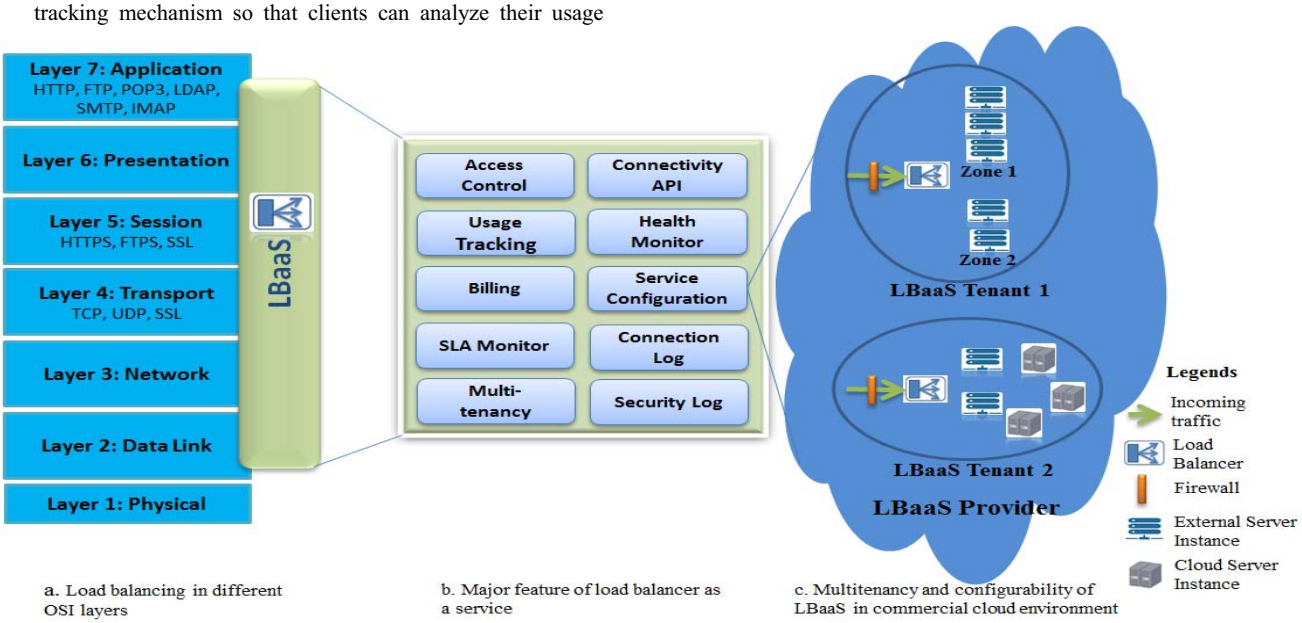

Figure 8. Characterizing load balancer as a service

- ***Built in billing****:* This is a very important feature because this gives LBaaS tenants the capability to get highest return on their investment. The LBaaS clients need the capability to control billing in a self-service manner to help them stay within their budget.
- ***Pricing model***: Competitive and rational pricing model play a huge role in becoming an LBaaS provider to be successful. In traditional pricing model where yearly license or subscription fee is necessary, the customers have higher risk of upfront investment. On the other hand, the pay as you go model reduces this risk and allows small to medium size businesses to take advantage of this service through smaller investment risk.
- ***Low vendor-lock-in risk****:* In order to minimize the vendor-lock-in risk and hence popular, the LBaaS providers need to follow common specification so that the tenants can opt in or opt out of their service on will. LBaaS model needs universal standard and application programming interface (API) so that clients can easily migrate among different providers.
- ***Service Level Agreement (SLA)****:* SLA, the formal contract between load balancer providers and the clients, plays a very important role in determining the reliability and availability of LBaaS. Today's cloud service providers are always in the risk of distributed denial of service (DDoS), network intrusion, eavesdropping, man-in-the-middle and other attacks from hackers and rogue community. These pose an enormous threat to the businesses running on cloud. Thus LBaaS providers need to demonstrate serious commitments towards their clients by guaranteeing their service with compensation, if they fail to meet their service agreement.
- ***Source code availability****:* Making source code available allows the LBaaS customer to enhance, customize and tailor the service to achieve competitive edge. Since the clients have the full access to the source code, it reduces the business risk and gives them additional security in case their service providers fail to meet their need. It also minimizes the risk of vendor-lock-in.
- ***Multitenancy****:* Multitenancy is a core feature of every service orientation. Multitenant architecture allows serving multiple tenants in an isolated and non-overlapping fashion, see Fig. 8-c. This is a crucial feature for the LBaaS service provider to become successful.
- ***Load balancer (LB) health monitoring****:* Health check capability of LBaaS gives the load balancer to recover from failure without interrupting the service. General health check mechanisms are Ping, TCP, HTTP Get and UDP.
- ***Internal and external server load balancing****:* The capability of LBaaS provider to allow their load balancer to balance traffic for both internal and external server instances give more agility and adaptability for diverse businesses. The clients with enormous legacy applications running in-house can take advantage of leveraging their data centers with cloud servers and load balancers.
- ***Dedicated or Static IP address****:* Allowing dedicated IP address assignment for the load balancer gives the opportunity to share it other load balancers for a certain customer.
- ***Connection Logging****:* This feature allows the tenant to avoid malicious traffic by setting limits on the number of connections per IP.
- ***Security Logging****:* Security logging allows the tenants to perform security auditing in case of network intrusion or hacking activity.

3.3. A Comparative Study among Major Market Players

Load balancing as a service is provided by some popular cloud service providers. We performed a comparative study among Amazon Elastic Load Balancing (ELB) [19, 20], Windows Azure [21, 22], Rackspace [23, 24], HP Cloud Load Balancing (CLB) [25] and GoGrid [26] (see Table III).

TABLE III. **A COMPARATIVE STUDY OF MAJOR LOAD BALANCER AS A SERVICE PROVIDERS**

Feature	Amazon ELB	Windows Azure	Rackspace	HP CLB	GoGrid
Load balancing algorithms	• Round robin LB • Sticky-session	• Performance LB • Failover LB • Round Robin LB	• Random LB • Round robin, weighted round robin • Least connections, weighted least connections	• Round robin • Least connection	• Round Robin, weighted round robin • Least connection (LC), weighted LC • Source Address Hashing
Communication interface	HTTP, HTTPS, TCP, SSL, or Custom CLI	HTTP, HTTPS, TCP, UDP, SSL	HTTP, HTTPS, SSL, REST, TCP, UDP, LDAP, LDAPS, FTP, SFTP, POP3, SMTP	HTTP, HTTPS, TCP, REST, Python CLI, or HP Console UI	HTTP, TCP, SSL
Pricing model	• Per load balancer – hour, • Per GB of processed data	• Pay As You Go • 6-months with pay monthly or pre-pay • 12-months with pay monthly or pre-pay	• Per instance • Per instance with SSL • Concurrent connections	Free for private beta as of November 2013 for HP Public Cloud users	• Pay as you go per hour, per server • Monthly prepaid by RAM usage • Monthly and Annual Pre-pay
Free trial available	750 hours/month up to 15 GB for one year	1 month	No	For HP Public Cloud users	No
Service Level Agreement (SLA)	99.99% monthly uptime	99.99% uptime	99.99% uptime	99.95% monthly uptime	100% uptime
Load balancer source code availability	Proprietary	Proprietary	Open source	Open source via OpenStack	Proprietary
Vendor-lock-in risk	Medium	Medium	Low	Low	Medium
Connection logging	No	Unknown	Yes, via Apache access logs.	Yes, via REST API	Unknown
Support internal and external server LB	No. Only for Amazon EC2 instances	Yes	Yes	Yes	Yes
Health monitoring	HTTP and TCP based	HTTP and HTTPS based	HTTP response code	Unknown	HTTP, Connect, SSL
Dedicated public IPs	Yes	Yes	Yes	Yes	Yes

4. FUTURE NEEDS AND CHALLENGES

A. Needs

Some of the major needs for cloud load balancers are listed below:

- ***Need #1 – Exploit platform heterogeneity****:* The need to understand the complex networks and the ways in which we can exploit the system by understanding the trade-offs and interplay is important.
- ***Need #2 – Zero Vendor-Lock-In risk through standardization****:* Interoperability among load balancing solutions that are currently available in the market is extremely crucial. LBaaS providers complying with the standards and guidelines of OpenStack technology can help to address this issue [28].
- ***Need #3 – Energy efficient load balancer****:* The need to come up with energy efficient techniques for load balancing, that will increase the cloud computing performance, is very important.
- ***Need#4 – Fault tolerance capability****:* Failover management in cloud environment is extremely important as failure can be triggered by many factors due to the diverse nature of cloud network. Hence load balancer needs to have high fault tolerance capability.
- ***Need #5 – Flexible communication API***: Current software services allow integration of diverse businesses through many different back-end systems. Thus availability of most communication interfaces such as HTTP, HTTPS, TCP, UDP, SSL, REST API, and Web Services are very essential for LBaaS to be successful.

B. Challenges

While other service oriented models such as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) are quickly moving towards maturity and standardization, LBaaS is still in its infancy and many problems still remain to be solved. Here is a list of major challenges:

- ***Challenge #1 - Elastic scalability****:* Elasticity is the most important feature for provided services through cloud computing; and the automatic allocation and freeing up of resources by maintaining the same performance and optimal usage of the resources is crucial and challenging. While moving the virtual machines from one physical server to the other, completely avoiding encountering bottlenecks is challenging.
- ***Challenge #2 – Network topology independence****:* The algorithm to be used is highly dependent on the composition and the topology of the network of servers.
- ***Challenge #3 – Communication interface****:* The arrangement and the provisioning of the diverse and large scale systems that are underlying in a network for virtualization must be understood clearly.
- ***Challenge #4 – Algorithm configurability****:* Effective load balancing techniques requires combination of algorithms, sensing of trade-offs and automated switching.
- ***Challenge #5 – Service Level Agreement (SLA)****:* Achieving load balancing while maintaining user satisfaction and resource utilization ratio is very challenging.
- ***Challenge #6 – Environment friendliness****:* Maintaining acceptable performance from the datacenters, while making sure that energy consumption is reduced, is challenging.

- ***Challenge #7 – Extensibility to both cloud server instances and internal servers**:* Most available LBaaS providers only allow their load balancers to manage load for the server instances rented from themselves only. This limits the extensibility of load balancers for existing enterprises that already have their internal servers and willing to manage their servers through load balancers from LBaaS provider.

5. CONCLUSIONS

We have seen a rapid growth of cloud computing in recent years. The business model of load balancer as a service (LBaaS) in cloud computing environment is really impressive and attractive. But the technologies involved are still in their infancy and need to mature for the companies to believe and trust their business with cloud computing. There are many challenges; the major ones are listed in this paper that needs to be addressed by the research community. In this paper we discussed about load balancing, its service orientated business model and its relevance in the cloud environment. We discussed algorithms, approaches and strategies based on research conducted and analyzed. We have also provided a big picture on how load balancing is being provided as a service by major cloud players in the industry today. We conclude that this is a field that needs a lot of research and each cloud vendor will have to adopt the right technique to achieve optimal performance and business endurance.

REFERENCES

[1] Ellrod, C. Load Balancing – Least Connections Windows Azure Traffic Manager [Online]. Available: http://blogs.citrix.com/ 2010/09/02/ load-balancing-least-connections. Published Sep. 2, 2010. Accessed on Feb. 18, 2012.

[2] Weighted Least-Connection Scheduling [Online]. Available: http://kb.linuxvirtualserver.org/wiki/Weighted_Least-Connection_ Scheduling. Accessed on Feb. 18, 2012.

[3] Least-Connection Scheduling [Online]. Available: http://kb.linuxvirtualserver.org/wiki/Least-Connection_Scheduling. Accessed on Feb. 18, 2012.

[4] Round-Robin Scheduling [Online]. Available: http://kb.linuxvirtualserver.org/wiki/Round-Robin_Scheduling. Accessed on Feb. 18, 2012.

[5] Round-robin Scheduling [Online]. Available: http://en.wikipedia.org/ wiki/Round-robin_scheduling. Accessed on Feb. 18, 2012.

[6] Weighted Round-robin Scheduling [Online]. Available: http://kb.linuxvirtualserver.org/wiki/Weighted_Round-Robin_Scheduling. Accessed on Feb. 18, 2012.

[7] Dynamic Feedback Load Balancing Scheduling [Online]. Available: http://kb.linuxvirtualserver.org/wiki/Dynamic_Feedback_Load_ Balancing_Scheduling. Accessed on Feb. 18, 2012.

[8] Yao, J. and He, J., "Load Balancing Strategy of Cloud Computing based on Artificial Bee Algorithm", *Proc. of the 2012 8th Intl Conf. of Computing Technology and Information Management (ICCM)*, 24-26 Apr. 2012, Seoul, Korea, pages 185 – 189.

[9] S. Wang, K. Yan, et al., "Towards a Load Balancing in a three-level cloud computing network," *in 3rd IEEE Intl. Conf. on Computer Science and Information Tech.*, 2010, vol. 1,2010, pp. 108-113.

[10] Z. Zhang, and X. Zhang, "A Load Balancing Mechanism Based on Ant Colony and Complex Network Theory in Open Cloud Computing Federation", *Proc. of 2nd Intl. Conf. on Industrial Mechatronics and Automation (ICIMA)*, Wuhan, China, May 2010, pages 240-243.

[11] V. Nae, R. Prodan, and T. Fahringer, "Cost-Efficient Hosting and Load Balancing of Massively Multiplayer Online Games", *Proceedings of the 11th IEEE/ACM International Conference on Grid Computing (Grid)*, IEEE Computer Society, Oct. 2010, pages 9-17.

[12] M. Randles, D. Lamb, and A. Taleb-Bendiab, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing," *in WAINA'10: Proc. of the 2010 IEEE 24th Int. Conference on Advanced Information Networking and Applications Workshops*, Perth, Australia, Apr. 2010.

[13] A. Bhadani and S. Chaudhary, "Performance Evaluation of Web Servers using Central Load Balancing Policy for Virtual Machines on Cloud" , *in Proceedings of the Third Annual ACM Bangalore Conference*, Article No 16, New York, 2010.

[14] Xi. Liu, Lei. Pan, Chong-Jun. Wang, and Jun-Yuan. Xie, "A Lock-Free Solution for Load Balancing in Multi-Core Environment", *3rd IEEE International Workshop on Intelligent Systems and Applications (ISA)*, 2011, pages 1-4.

[15] H. Mehta, P. Kanungo, M. Chandwani, "Decentralized content aware load balancing algorithm for distributed computing environment," *in Intl. Conf. and Workshop on Emerging Trends and Technology, Mumbai*, Feb. 2011.

[16] E. Di Nitto, D.J. Dubois, R. Mirandola, F. Saffre and R. Tateson, Applying Self-Aggregation to Load Balancing: Experimental Results. *in Proc. of the 3rd Intl. Conf. on Bioinspired Models of Network, Information and Computing Systems (Bionetics 2008)*, Article 14, 25 – 28 Nov. 2008.

[17] Kaur, J. "Comparison of load balancing algorithms in a Cloud", *International Journal of Engineering Research and Applications(IJERA)*, Vol. 2, Issue 3, May-Jun. 2012, pp.1169-1173.

[18] Khiyaita, A., Zbakh, M., El Bakkali, H. and El Kettani, D. "Load balancing cloud computing: state of art", *in Proc. of the 2012 National Days of Network Security and Systems*, JNS2, 2012, pp. 106–109.

[19] Amazon White Paper, Elastic Load Balancing Developer Guide [Online]. Available: http://awsdocs.s3.amazonaws.com/ ElasticLoadBalancing/latest/elb-dg.pdf. Accessed on Aug. 21, 2013.

[20] Elastic Load Balancing. Available [Online]: http://aws.amazon.com/elasticloadbalancing/. Accessed on Aug. 21, 2013.

[21] About Load Balancing Methods [Online]. Available: http://msdn.microsoft.com/en-us/library/windowsazure/ dn339010.aspx. Accessed on Feb. 22, 2013.

[22] Windows Azure Traffic Manager [Online], Available: http://msdn.microsoft.com/en-us/library/windowsazure/ hh744833.aspx. Accessed on Sep. 2, 2013.

[23] Rackspace Cloud Load Balancers. Available [Online]: http://www.rackspace.com/cloud/load-balancing/. Accessed on Sep. 22, 2013.

[24] The Technologies Behind Cloud Load Balancers. Available [Online]: http://www.rackspace.com/cloud/load-balancing/technology/. Accessed on Sep. Sep. 22, 2013.

[25] HP Cloud Load Balancer. Available [Online]: http://www.hpcloud.com/products-services/load-balancer?t=features. Accessed on Oct. 1, 2013.

[26] GoGrid Load Balancer. Available [Online]: http://www.gogrid.com /products/load-balancers. Accessed on Oct. 20, 2013.

[27] Load Balancing. Available [Online]: http://www.citrix.com/ glossary/load-balancing.html. Accessed on Oct. 20, 2013.

[28] Neutron/LBaaS. Available [Online]: https://wiki.openstack.org/wiki/Neutron/LBaaS