L3B: Low Level Load Balancer in the Cloud

Monika Simjanoska, Sasko Ristov, Goran Velkoski, and Marjan Gusev

Ss. Cyril and Methodius University, Faculty of Information Sciences and Computer Engineering,
Rugjer Boshkovik 16, PO Box 393, 1000 Skopje, Macedonia
m.simjanoska@gmail.com
sashko.ristov@finki.ukim.mk
velkoski.goran@gmail.com
marjan.gushev@finki.ukim.mk

***Abstract*—Cloud service providers (CSPs) aim to preserve infinite scalable and elastic computing resources. In addition, user requirements for flexible cloud computing resources impact the overall performance of the cloud. From the other side, virtualization adds additional layer in cloud stack and therefore decreases the service performance compared to on-premise traditional hosting. It seems that the price to pay for scalability and flexibility is not so small. Therefore, in order to improve the overall cloud performance and make it sustainable there is need for real-time resource orchestration. Considering the heterogeneous architecture of the cloud, CSPs must develop appropriate strategy to achieve maximum performance minimizing the utilization of hardware resources and energy consumption. In this paper, we propose a novel Low Level Load balancing (L3B) approach applied to Infrastructure as a Service (IaaS). Our approach preserves the cloud's elasticity by dynamic activation of cloud resources and load balancing the traffic over the resources on low network level. In addition, our L3B balancer tends to load the instances in the region where they provide maximum performance. Introducing L3B improves the overall cloud performance, reduces power consumption and customers' cost for renting cloud resources, etc.**

***Index Terms*—Cloud Computing, IP, Networking, Performance**

I. Introduction

Cloud Computing is a new paradigm of on demand computing resources promised to be continuously scalable and elastic. However, the virtualization in the cloud additionally decreases the performance to approximately 70% of on-premise for both memory demand and compute intensive web services [1]. The performance drawback is emphasized for input / output bound applications (more than 110% overhead) and a little (about 10% overhead) for CPU bound applications [2]. The large amount of inconsistent traffic also drawbacks the performance in the cloud compared to on-premise performance.

Many hypervisors exist to instantiate virtual machine (VM) instances, using either full virtualization or para virtualization. However, maximum resources that can be allocated to a certain VM instance are determined by the resources that the physical server has. This issue limits the maximum load that a VM instance can handle without affecting the performance. One mechanism to instantly orchestrate the cloud resources and maximize the performance using minimum hardware resources is to develop appropriate load balancing strategy [3]. Load balancing is one solution to balance the requests between two or more VM instances of client's applications in a way that they can be provisioned automatically without requiring changes to the network or its configuration [4]. It increases the service level agreement (SLA) and improves the resources usage [5]. Balancing the load among "tightly" VM instances can optimize the performance, while balancing the load among "loosely" VM instances can optimize the availability [6].

In this paper, we propose a novel load balancing approach to IaaS cloud model in order to maintain balance with minimal hardware resource utilization and achieve maximum performance. Our dynamic load balancing approach is supposed to handle the imbalance considering the heterogeneity of the nodes. It is not profitable to have all of the nodes running concurrently. The system should be elastic, i.e., to create new nodes or shut down under-utilized nodes at any time to optimize the customer performance and its own operation costs [7]. Therefore, we introduce intelligence on the internet layer of the TCP/IP model to balance the load among the active instances using minimum hardware resources.

The rest of the paper is organized as follows. State of the art in load balancing is presented in Section II. In Section III we present the L3B system architecture and design. Performance analysis of our proposed L3B balancer in the cloud is realized in Section IV. L3B pros and cons are given in Section V. Conclusion and the future work are specified in final sections VI and VII.

II. Related Work

Different load balancing techniques exist. Their impact to the performance and availability depends on a certain problem and condition. Nuaimi et al. [8] presented a nice survey for the performance of several load balancing algorithms in cloud computing. Kuhn and Sesum-Cavic [9] proposed SILBA (Self Initiative Load Balancing Agents) which exchanges pluggable algorithms to select the best algorithm for a certain problem and condition.

Load balancing techniques can be either centralized or distributed. The centralized load balancing techniques are controlled by a single central node and all the other nodes communicate with this node. In distributed architecture the load balancing algorithm is executed by all nodes present in the system and the task of load balancing is shared among them.

978-1-4673-2232-4/13/$31.00 ©2013 IEEE

Bhadani and Chaudhary [10] proposed Central Load Balancing Policy for Virtual Machines (CLBVM) to balance the load evenly in a distributed VM/cloud computing environment and migrates particular VM to lightly loaded server if its performance is affected by other VMs. Radojevic and Zagar [11] proposed an initial design of a centralized load balancer which takes into consideration other parameters as server load and application performance, rather than relying only on network parameters. Their CLBDM module is an improvement of Round Robin Algorithm [12]. This opposes the thesis which states that using additional metrics will result in more processing and additional communication overhead [13]. Randles et al. [14] evaluate and compare three different distributed load balancing approaches: Honeybee Foraging Behaviour, Biased Random Sampling and Active Clustering. However, centralized approaches for load balancing and resource allocation in cloud systems have design limitations as reduced scalability and communication overhead which can easily result in a bottleneck [15].

Wen et al. [16] stated that most load balancers assumed homogeneous nodes, whereas dynamic and heterogeneous systems are actually necessary to provide on-demand resources or services; therefore, they propose a node interactive approach which allocates and schedules resources in a reasonable way. However, their solution does not increase the throughput enough while utilizing the scaled system resources.

Sharma and Sharma [17] proposed a VM load balancer implemented in the abstract cloud computing environment CloudSim. They confirmed an increased performance and decreased average response time. However, CloudSim is simulating a cloud computing environment and the researcher does not need to get concerned about the low level details related to cloud-based IaaS [18], which would be a disadvantage in our research.

Hu et al. [19] proposed a load balancing strategy based on genetic algorithm, and historical data and current state of the system. However, this solution lacks security and packet routing since the cloud environment is dynamic and one VM instance can have the same IP of the same VLAN with recently turned off VM instance and many packets can be lost or directed to wrong VM instance. Jin et al. [20] proposed BAlance-Reduce (BAR) heuristic task scheduling algorithm which dynamically adjusts the data locality according to network state and cluster workload.

Several load balancers exist on the application layer. Ristov et al. [21] proposed e-Assessment architecture in the cloud creating one VM instance per assessment. Their solution instantiate VM instances only during the assessments and each VM instance works much faster with its own small database. Tsai et al. [22] proposed a two-tier Software as a Service (SaaS) architecture which increases the resources to bottleneck components.

Velkoski et al. [23] have found an existence of superior behavior meaning that cloud service performs much better when executed on parallel resources. There is a particular region which they refer to as superlinear, where web services hosted in the cloud achieve speed up greater than the number of scaled hardware resources. The reason lies in the fact that VM instance with smaller resources (total number of CPUs or amount of memory) saturates, while the VM instance with more resources does not saturate for the same load. Therefore, we aim to load the instances in the superlinear region in order to get maximum performance. Achieving maximum performance with minimum computing resources is profitable both for the customers and the CSPs in terms of additional costs for renting resources and power consumption costs, respectively.

Balancing the load provides not only better performance, but it can reduce the energy overhead, as well [24]. Power aware load balancing algorithms can achieve significant cost savings [25]. Adnan et al. [26] showed that load balancing algorithms that migrate the workload using the future electricity price prediction, can achieve significant cost savings compared to greedy approach algorithms. Reducing the number of active physical machines (nodes) in the cloud using balancing algorithm reduces the overall cost for electricity and cooling [27]. The load balancer proposed by Galloway et al. [28] is an approach to IaaS cloud architecture that maintains the utilization of all compute nodes and distributes virtual machines in a power efficient way. Similarly, Kasae and Oguchi [29] focus on a hybrid cloud in cloud computing, which is a combination of public and private clouds and propose a method implemented as a middleware, that can both process a large amount of data and control monetary costs, including power consumption.

III. L3B System Architecture and Design

In this section, we present the L3B architecture and design. The L3B is a centralized load balancer introduced between the clients and the heterogeneous nodes in an IaaS cloud service layer as depicted in Figure 1.

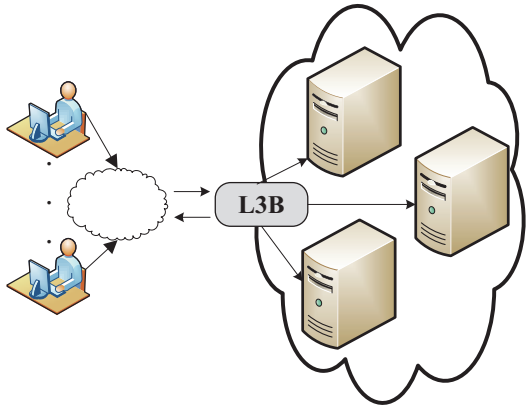

Fig. 1. L3B System architecture

L3B intermediates between the client's requests for services and the VM instances set in the cloud. This solution

is designed to maintain load balance and gain maximum performance using minimum computing resources.

Another important task for L3B is the cloud resource provisioning. If the load increases in such a way that service performance goes below a given threshold, then L3B will automatically instantiate a new VM instance. If the load decreases and service performance goes bellow the minimum threshold, then L3B will automatically shut down one of active VM instances.

In order to account the resource utilization when balancing the load, we introduce two modules in L3B, i.e. *Resource Management Module (RMM)* and *Packet Management Module (PMM)* as depicted in Figure 2.

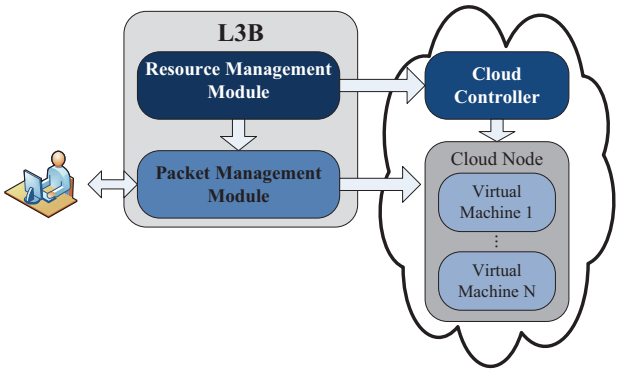

Fig. 2. L3B modules

RMM's main goal is to manage the provision of cloud resources according to current load and active VM instances utilization. It communicates with the cloud controller to create a new VM instance or shut down some of the existing VM instances hosted in cloud computing nodes. Additionally, RMM informs PMM for active VM instances in order to balance the load among them. PMM redirects the input packets to some of the active VM instances and forwards the responses to the exact client that sent the request.

Both modules consist of several internal agents and repositories depicted in Figure 3. The following two sections describe their design in details.

A. Resource Management Module

RMM manages the cloud resources. This module communicates with PMM and the cloud's controller. Therefore, it regularly gathers information of the active resources utilization and takes into consideration the client's needs for more or less resources.

Its goal is twofold:

- To minimize the cloud service provider costs, i.e. minimize power consumption and resource utilization; and
- To maximize the service performance.

In order to obtain minimal resource utilization, we design RMM with *Resource Utilization Manager (RUM)* and *Resource Provisioning Agent (RPA)*.

Particularly, RUM manager is responsible for deriving intelligent decisions whether a new VM instance should be instantiated or whether an existing active VM instance should be released. RUM can observe the VMs and physical machine CPU utilization, service response times, memory usage and other hardware utilization metrics for the running VM instances and the host. Once a decision is derived, RUM commits directive to RPA. RPA only executes RUM orders and does not deduce any intelligent conclusions. It communicates with Cloud controller for resource provisioning. The aftermath of a good operating RMM is maintaining minimal computing utilization that will guarantee the service performance threshold and also achieving the best performance from the same quantity of computing resources, thus reducing the cost for power consumption.

B. Packet Management Module

PMM module is the core of the L3B and intelligently manages all the inconsistent traffic coming from the clients. The design of PMM module is also depicted in Figure 3. It is used to manage the client's requests, i.e. to forward particular incoming packet to particular VM instance in order to balance the load of all VM instances. When the target VM responses back, PMM forwards the response packet to the client that has sent the corresponding request.

PMM has two interfaces to communicate with the clients and with the VM instances (Cloud Nodes), i.e. *L3B Outside Interface (OI)* and *L3B Inside Interface (II)* correspondingly as depicted in Figure 3.

L3B OI is an arbitrator between the clients and the PMM inner agents. When a new packet arrives, OI instantly sends information to the *Input Packet Decision Agent (IPDA)*. IPDA is the most important agent since its intelligent algorithm derives smart decisions in assigning the requests to appropriate resources. To determine which VM can handle the request in order to preserve sustainable performance, IPDA uses information from the *Resource Utilization Repository (RUR)* and the *Load Balancing Configuration (LBC)*. RUR is in direct association with RMM, precisely with RUM. Therefrom, IPDA receives real time information of the hardware utilization actuality. Once IPDA acquires intelligent decision based on previously mentioned metrics, it notifies the *Packet Translation (PT)* agent. In a meantime, PT receives the packet from OI and proceeds with translation of the IP header using NAT/PAT (Network Address Translation / Port Address Translation) functionality to translate between internal and external network addresses. PT's NAT/PAT translations are then stored in the shared *Packet Translation Repository (PTR)* and the transformed packet is forwarded to L3B II in order to be forwarded to target VM instance using load balancing.

L3B II arbitrates between the inner L3B agents and the cloud nodes, i.e. the target VM instances. Applying IPDA's decision, it forwards the IP packet with modified header to a target VM instance in particular cloud node. Additionally, L3B II has the ability to operate in the opposite direction. It receives the responses from the VMs since the modified IP

978-1-4673-2232-4/13/$31.00 ©2013 IEEE

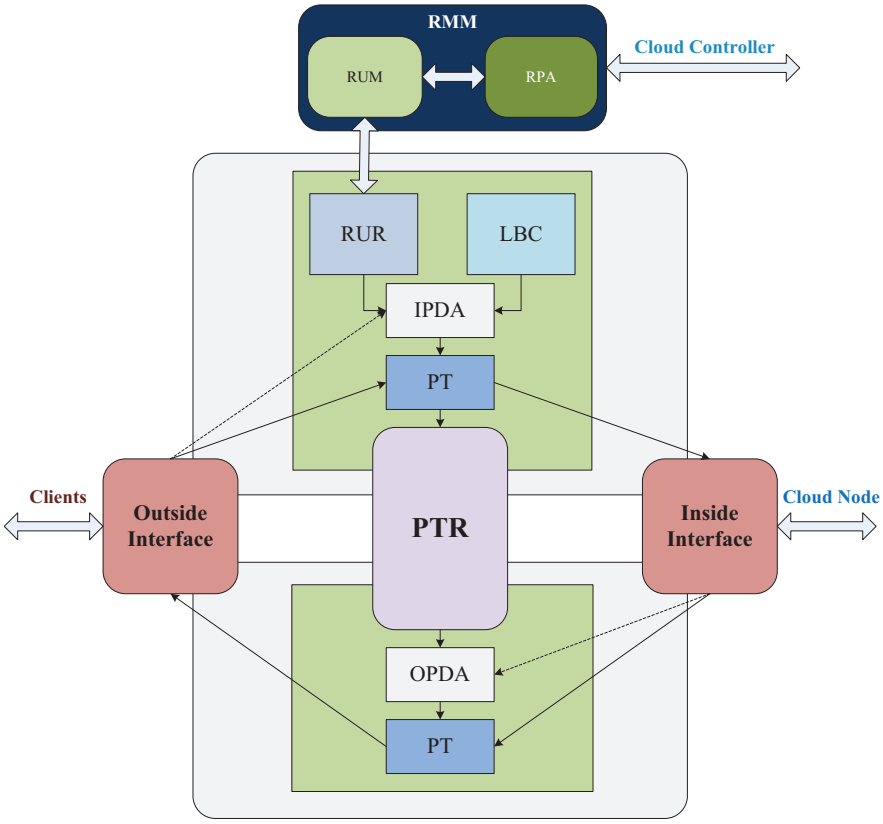

Fig. 3. L3B modules design

packet was previously encapsulated in data link frame with II as source address (MAC Address since ethernet protocol is used) and the target VM as destination MAC address. Thus, the target VM will address the response with the address of II as destination MAC address. Once L3B II receives a reply, it informs the *Output Packet Decision Agent (OPDA)*. OPDA goes through the shared PTR and perceives the packet's PT instructions. As long as OPDA derives a decision based on the obtained information, it informs the inverted PT. At the time the response packet arrives, PT repeats the inverse NAT/PAT procedure of translating the external IP address into the original and the packet is sent to L3B OI. Then OI sends the response to the client that has sent the original request.

IV. L3B Performance Analysis

In this section we perform L3B mathematical performance analysis and discuss the possible performance enhancement in the cloud.

A. Traditional Client-Server Scenario

Figure 4 depicts a traditional client-server scenario, where WS denotes Web server and AS and DS denote application and database server correspondingly.

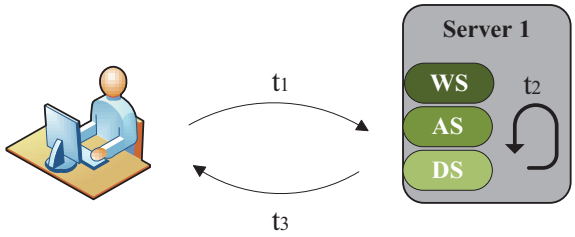

Fig. 4. Traditional client-server design

The client sends a request to the server (Server 1). In a 3-tier web application, the request is sent to the web server, then forwarded to the application server and usually to the database server. The response is forwarded back to the web server and sent to the client.

The total response time t_A of client-server communication is defined in (1) as a sum of the time t_1 necessary for the packet to be delivered from the client to the server, the time t_2 necessary for the server to manage the request and compute the response, and the time t_3 spent for the response to be

delivered from the Server 1 back to the client.

$$t_A = t_1 + t_2 + t_3 \quad (1)$$

Let's analyze the times t_1, t_2 and t_3. Times t_1 and t_3 usually do not depend on service provider (Server 1) and its hardware resources, but from the network path from the client to the server. These times are usually different in range of milliseconds to few seconds, but we can assume that they are constant for each request and response since we are interested to improve time t_2. The time t_2 substantially depends on the size, the amount and the type of the requests, for example, if security is implemented in the requests [30]. Moreover, the requests can be static html, dynamic web application or web service, or even additionally query to some database. All these issues impact to the time t_2. However, we can assume that the time t_2 is constant for the same client-server architecture and client requests and server response similar to explanation presented for the times t_1 and t_3.

B. L3B Scenario

When implementing L3B as an additional layer between the clients and the server, additional latency is added to the original response time. Figure 5 depicts the client-server communication for proposed L3B scenario.

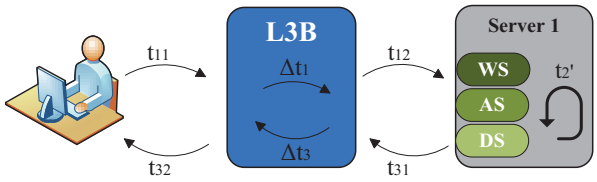

Fig. 5. Client-Server design with added virtualization

The client sends the request to the L3B instead of directly to the end-point VM instance, which forwards the request packet to the server as depicted. Introducing L3B imposes additional delay Δt_1 for the client's request since L3B should determine the target VM instance where the request should be forward and to readdress the request packet to that VM instance. Similarly, L3B imposes Δt_3 delay for the response since L3B should determine the client address where the response should be sent and to readdress the response packet to that client. Both times $\Delta t_1, \Delta t_3 > 0$. Hereupon, the total response time for this L3B scenario client-server communication is defined in (2).

$$t'_A = t_{11} + t_{12} + \Delta t_1 + t_{32} + t_{31} + \Delta t_3 + t'_2 \quad (2)$$

Let's describe all times in (2). Time t_{11} denotes the time needed for the client's request to be delivered to L3B OI in the PMM module. L3B OI receives the packet and notifies the IPDA intelligent agent. Once IPDA derives its assignment decision, it informs the PT agent. Meanwhile, the PT agent receives the packet and does an appropriate NAT / PAT translation to the packet header. Then the translated packet is forwarded to L3B II, and the packet transformation information is saved in the shared PTR. All these processes are finished in time Δt_1.

The translated request packet from L3B II arrives to the server (target VM instance in some cloud node) in time t_{12}. Then the server processes the received packet in time t'_2. This time substantially depends on the size, the amount and the type of the requests, similarly as in traditional client-server scenario without L3B. After the request packet is processed, the server sends the response packet back to L3B in time t_{31}. L3B II receives the response packet and notifies the OPDA intelligent agent. OPDA accesses the shared PTR and reads the information of the packet's transformation previously saved. In a meantime PT receives the packet and once OPDA makes its decision, the packet is forwarded to L3B OI. The time Δt_3 denotes the time for all processes inside L3B, i.e. moving the response packet from II to OI. Finally, the response packet is delivered to the client in time t_{32}.

We can assume that the times t_{11}, t_{12}, t_{31} and t_{32} are constant similar to explanation presented for the times t_1 and t_3 in traditional client-server scenario. Similarly to traditional client-server scenario, we can assume that the time t'_2 is constant for the same client-server architecture and client requests and server response.

C. Response Time Improvements

In this section we analyze all times in (1) and (2) in order to determine how to decrease t'_A compared to t_A, i.e. to speed up the response time in L3B scenario, despite the additional latency when introducing L3B.

In previous sections IV-B and IV-A we assumed that the times t_1, t_3 in traditional client-server scenario, as well as the times t_{11}, t_{12}, t_{31} and t_{32} in L3B scenario are constant for particular request and response.

The relation for request times for traditional client-server and L3B scenarios is given in (3) of Lemma 1.

Lemma 1: The time needed for the request packet to be delivered from the client to the server in traditional client-server scenarios is equal to the sum of times for the request packet to be delivered from the client to L3B OI and from L3B II to the server in L3B scenario, as defined in (3).

$$t_1 = t_{11} + t_{12} \quad (3)$$

Proof: The relation (3) is satisfied since the L3B is introduced between the client and the server for the request delivery and all L3B latency for the request packet is defined as time Δt_1. ■

The relation for response times for traditional client-server and L3B scenarios is given in (4) of Lemma 2.

Lemma 2: The time needed for the response packet to be delivered from the server to the client in traditional client-server scenarios is equal to the sum of times for the response packet to be delivered from the server to L3B II and from L3B OI to the client in L3B scenario, as defined in (4).

$$t_3 = t_{32} + t_{31} \tag{4}$$

Proof: The relation (4) is satisfied since the L3B is introduced between the the server and the client for the response delivery and all L3B latency for the response packet is defined as time Δt_3. ■

Using the equations (3) and (4) from lemmas 1 and 2, the relation (2) can be simplified as shown in (5).

$$t'_A = t_1 + t_3 + \Delta t_1 + \Delta t_3 + t'_2 \tag{5}$$

In order to have a benefit of L3B implementation, we must perform a good trade-off between the increased latency and the time t'_2. That is, balancing the load with L3B among several VM instances and their underutilization will improve the service performance and thus reduce the time for processing the request and generating the response on server side [31].

An enhanced performance can be obtained if the time t'_2 is reduced. The time t'_2 can satisfy the fixed-time speed-up model, known as Gustafson's law [32]. The authors in [33] state that if the problem size is scaled up to maintain a fixed execution time, the fixed-time speed-up is a linear function of p, where p is the number of processors.

This means that using more resources will reduce the time t'_2 and thus the total response time t'_A. Let's analyze the condition when L3B scenario will provide better performance than traditional client-server scenario. Theorem 1 gives the required condition.

Theorem 1: L3B scenario will provide better performance than traditional client-server scenario if (6) is satisfied.

$$t_2 - t'_2 - (\Delta t_1 + \Delta t_3) > 0 \tag{6}$$

Proof: Lets Δt_A denotes the difference between response times of traditional client-server and L3B scenarios, i.e. $\Delta t_A = t_A - t'_A$. Using the values for t_A and t'_A defined in relations (1) and (5) correspondingly yields the condition in (6). ■

Let's explain the condition in Theorem 1. Introducing L3B will impose additional latencies Δt_1 and Δt_3. Since L3B is low level load balancer, i.e. on network layer, these latencies are inconsiderable compared to server response time. On the other side, using more resources in the cloud can reduce the time t'_2 up to p times (or more than p times if server works in superlinear region [23]) where p is the factor of scaling the resources.

V. L3B Pros and Cons

We mentioned some benefits and detriments of introducing L3B between the client and the server. In this section we exhibit all the pros and cons of our proposed L3B approach.

A. L3B Pros

Although introducing L3B generates additional latency in delivering the request and response packets in the direction from client to server and vice versa, it provides several benefits especially if the server is hosted in VM instances, as described in Section III. In this section we will discuss the pros from both customers' and CSPs' aspects.

Considering economic reasons, the customers need cost-aware CSP. Most common CSPs offer charging model based on utility consumption. The price doubles for double rented resources [34], [35], [36]. Our L3B activates and deactivates the computing resources on user demand and thus it prevents additional customers' costs for idle resources. Moreover, this is also a CSP advantage in terms of maximizing hardware resources utilization and therefore reducing the power consumption costs. A simple reserve capacity model for dimensioning the capacity of cloud based system in the presence of time-varying customer demand [37] can be implemented in our L3B in order to decrease even more the latency that L3B imposes. After the research presented in [38] that 53% of the data centers monthly costs go to power and cooling, the authors in [39] present the fact that the IaaS providers are under enormous pressure to reduce energy consumption and even more meet environmental standards.

Another benefit from the proposed L3B approach apart from cost and power awareness is increased number of the CSPs customers. By implementing L3B, CSPs can offer value added services with the facilitation of renting less computing resources and in the same time get maximum performance. L3B will also satisfy their service level agreements. For example, Google and Microsoft guarantee 99.9% availability (uptime) [40], [41] and Amazon guarantees 99.95% [42]. Availability of 99.9% means maximum 8.77 hours of downtime per year and 99.95% means 4.39 hours of downtime per year.

From a performance point of view L3B avoids congestion since it works on a cloud customer service level. Also, failure of one or more cloud nodes could decrease the performance, but the system will remain active and available since the cloud possesses unlimited resources and other VM instances will be instantiated on another health cloud node.

B. L3B Cons

When introducing L3B we may confront the most common issues initiated by the design of the centralized load balancers. The election of a single central node reduces the reliability of the L3B. Hence, if the central node fails, it will result in L3B decline correspondingly.

Another negative issue for our L3B is the fact that all L3B modules, agents and repositories require additional hardware resources, i.e. computing, memory and storage capacity. L3B adds additional latency both for requests and responses in order to decide which VM instance to forward the packet and to translate the request packets to that particular VM instance. Additional latencies are produced when L3B determines which client will receive the response and also for translation of the response packet header for that particular client.

The authors in [43] refer to the desire of handling the client's requests by the same node throughout the client's session, as session affinity, or, session stickiness. They state that while distributing the load evenly among the available nodes, there is no guarantee that all the requests coming from the users will be handled by the same node from the pooled resources. This issue implies another L3B drawback. Since the L3B is implemented on the network level of the TCP/IP model, it is acquiescent to the IP packet length constraint of up to 64KB. Therefore if the request size is greater than IP packet size (64KB), than two or more IP packets with different IDs will be sent to two different nodes with greater probability than to be handled by a same node and thus preserve from infinite appealing of resending the packets.

Dynamic creation of VM instances is not a real time process. Creating a new VM instance sometimes could be several minutes. It is a period when the load can be significantly changed and the VM instance should be deactivated even before it is started completely.

VI. Conclusion

We have developed a new strategy to address the problem of decreased performance of the cloud in comparison to on-premise system due to additional virtualization layer. The new strategy handles the increased service requests and manages the CSPs computing resources. Since the IaaS cloud tends to be heterogeneous environment, we proposed a mechanism to retain optimal hardware utilization and achieve maximum performance utilizing the same cloud computing resources orchestrating differently.

We introduced a novel centralized approach implemented on the network level of the TCP/IP model. Our L3B is an intelligent agent which orchestrates the nodes resources according to service requests. In order to achieve maximum performance, we loaded the instances in the superlinear region, where customers get more performance than expected.

Introducing L3B provides benefits both for cloud customers and CSPs. Customers can achieve better performance for their services hosted on the same resources for the same price. CSPs will reduce the costs for power electricity, increase their availability and reliability into their SLAs increase the number of customers using the same resources etc.

Not only that we proved that introducing L3B in the cloud will improve the performance of the client-server model, but we also determine the condition how it can be achieved.

VII. Future Work

In this paper we have presented an implementation of the proposed L3B architecture. Our further research will be focused on comparing the performances of real implementations of traditional client-server scenario with the implementation of L3B scenario in order to derive conclusions about the overall performance improvements. However, cost analysis is also important when scaling the resources.

We will continue our research in developing an intelligent algorithm and compare it with the existing network load balancing algorithms as round-robin, random-allocation, etc and introduce a simple reserve capacity model for dimensioning the capacity of cloud based system in the presence of time-varying customer demand. Moreover, we will proceed to improve existing modules and will aim to overcome the design obstacles. The goal is to upgrade our L3B to overcome previously mentioned obstacles especially for greater request message size which will produce several IP packets which should be forward to the same target VM instance.

References

[1] S. Ristov, G. Velkoski, M. Gusev, and K. Kjiroski, "Compute and memory intensive web service performance in the cloud," in *ICT Innovations 2012*, S. Markovski and M. Gusev, Eds. Springer Berlin / Berlin Heidelberg, 2013, vol. AISC 257, pp. 215–224.

[2] W. Lloyd, S. Pallickara, O. David, J. Lyon, M. Arabi, and K. Rojas, "Migration of multi-tier applications to infrastructure-as-a-service clouds: An investigation using kernel-based virtual machines," in *Grid Computing (GRID), 2011 12th IEEE/ACM International Conference on*, 2011, pp. 137–144.

[3] M. Randles, E. Odat, D. Lamb, O. Abu-Rahmeh, and A. Taleb-Bendiab, "A comparative experiment in distributed load balancing," in *Developments in eSystems Engineering (DESE), 2009 Second International Conference on*, 2009, pp. 258–265.

[4] J. Gasior and F. Seredynski, "Load balancing in cloud computing systems through formation of coalitions in a spatially generalized prisoner's dilemma game," in *CLOUD COMPUTING 2012, The Third International Conference on Cloud Computing, GRIDs, and Virtualization*, 2012, pp. 201–205.

[5] R. Lee and B. Jeng, "Load-balancing tactics in cloud," in *Proceedings of the 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, ser. CYBERC '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 447–454.

[6] Y. Zhao and W. Huang, "Adaptive distributed load balancing algorithm based on live migration of virtual machines in cloud," in *Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC*, ser. NCM '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 170–175.

[7] S. Loesing, M. Hentschel, T. Kraska, and D. Kossmann, "Stormy: an elastic and highly available streaming service in the cloud," in *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, ser. EDBT-ICDT '12. New York, NY, USA: ACM, 2012, pp. 55–60.

[8] K. Nuaimi, N. Mohamed, M. Nuaimi, and J. Al-Jaroodi, "A survey of load balancing in cloud computing: Challenges and algorithms," in *Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on*, 2012, pp. 137–142.

[9] E. Kühn and V. Sesum-Cavic, "A space-based generic pattern for self-initiative load balancing agents," in *Proceedings of the 10th International Workshop on Engineering Societies in the Agents World X*, ser. ESAW '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 17–32.

[10] A. Bhadani and S. Chaudhary, "Performance evaluation of web servers using central load balancing policy over virtual machines on cloud," in *Proceedings of the Third Annual ACM Bangalore Conference*, ser. COMPUTE '10. New York, NY, USA: ACM, 2010, pp. 16:1–16:4.

[11] B. Radojevic and M. Zagar, "Analysis of issues with load balancing algorithms in hosted (cloud) environments," in *MIPRO, 2011 Proceedings of the 34th International Convention*. IEEE, 2011, pp. 416–420.

[12] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," *IEEE Internet Computing*, vol. 13, no. 5, pp. 14–22, Sep. 2009.

[13] Z. Sui and S. Pallickara, "A survey of load balancing techniques for data intensive computing," in *Handbook of Data Intensive Computing*, B. Furht and A. Escalante, Eds. Springer New York, 2011, pp. 157–168.

[14] M. Randles, D. Lamb, and A. Taleb-Bendiab, "A comparative study into distributed load balancing algorithms for cloud computing," in *Proceedings of the 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, ser. WAINA '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 551–556.

[15] D. Ardagna, S. Casolari, and B. Panicucci, "Flexible distributed capacity allocation and load redirect algorithms for cloud systems," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 163–170.

[16] H. Wen, Z. Hai-ying, L. Chuang, and Y. Yang, "Effective load balancing for cloud-based multimedia system," in *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*, vol. 1. IEEE, 2011, pp. 165–168.

[17] M. Sharma and P. Sharma, "Performance evaluation of adaptive virtual machine load balancing algorithm," *Performance Evaluation*, vol. 3, no. 2, 2012.

[18] R. N. Calheiros, R. Ranjan, C. A. F. D. Rose, and R. Buyya, "CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services," *CoRR*, vol. abs/0903.2525, 2009.

[19] J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," in *Proceedings of the 2010 3rd International Symposium on Parallel Architectures, Algorithms and Programming*, ser. PAAP '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 89–96.

[20] J. Jin, J. Luo, A. Song, F. Dong, and R. Xiong, "Bar: An efficient data locality driven task scheduling algorithm for cloud computing," in *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, ser. CCGRID '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 295–304.

[21] S. Ristov, M. Gusev, G. Armenski, K. Bozinoski, and G. Velkoski, "Architecture and organization of e-assessment cloud solution," in *Global Engineering Education Conference (EDUCON), 2013 IEEE*, March 2013, pp. 736–743, best paper award.

[22] W.-T. Tsai, X. Sun, Q. Shao, and G. Qi, "Two-tier multi-tenancy scaling and load balancing," in *e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on*, 2010, pp. 484–489.

[23] G. Velkoski, S. Ristov, and M. Gusev, "Performance behavior of cloud web services," University Ss Cyril and Methodius, Skopje, Macedonia, Faculty of Information Sciences and Computer Engineering, Tech. Rep. IIT:15-12, Aug. 2012.

[24] O. Sarood, A. Gupta, and L. Kale, "Cloud friendly load balancing for hpc applications: Preliminary work," in *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*, 2012, pp. 200–205.

[25] J. Galloway, K. Smith, and J. Carver, "An empirical study of power aware load balancing in local cloud architectures," in *Proceedings of the 2012 Ninth International Conference on Information Technology - New Generations*, ser. ITNG '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 232–236.

[26] M. A. Adnan, R. Sugihara, and R. K. Gupta, "Energy efficient geographical load balancing via dynamic deferral of workload," in *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, ser. CLOUD '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 188–195.

[27] X. Li, Z. Qian, R. Chi, B. Zhang, and S. Lu, "Balancing resource utilization for continuous virtual machine requests in clouds," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, 2012, pp. 266–273.

[28] J. Galloway, K. Smith, and S. Vrbsky, "Power aware load balancing for cloud computing," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, 2011, pp. 19–21.

[29] Y. Kasae and M. Oguchi, "Proposed of a load balancing method for data intensive applications on a hybrid cloud accounting for cost including power consumption." in *CLOSER*, F. Leymann, I. Ivanov, M. van Sinderen, and T. Shan, Eds. SciTePress, 2012, pp. 407–412.

[30] S. Ristov and A. Tentov, "Performance impact correlation of message size vs. concurrent users implementing web service security on linux platform," in *ICT Innovations 2011*, ser. Advances in Intelligent and Soft Computing, vol. 150. Springer Berlin / Heidelberg, 2012, pp. 367–377.

[31] R. Iakymchuk, J. Napper, and P. Bientinesi, "Improving high-performance computations on clouds through resource underutilization," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, ser. SAC '11. New York, NY, USA: ACM, 2011, pp. 119–126.

[32] J. L. Gustafson, "Reevaluating Amdahl's Law," *Communication of ACM*, vol. 31, no. 5, pp. 532–533, May 1988.

[33] X. Sun, Y. Chen, and S. Byna, "Scalable computing in the multicore era," in *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Programming*, Sep. 2008.

[34] Microsoft, "Windows Azure," [retrieved: March, 2013]. [Online]. Available: http://www.windowsazure.com/pricing/

[35] Google, "Compute engine," [retrieved: March, 2013]. [Online]. Available: http://cloud.google.com/pricing/

[36] Amazon, "EC2," [retrieved: March, 2013]. [Online]. Available: http://aws.amazon.com/ec2/

[37] B. Bouterse and H. Perros, "Scheduling cloud capacity for time-varying customer demand," in *2012 IEEE 1st Int. Conference on Cloud Networking (CLOUDNET) (IEEE CloudNet'12)*, Paris, France, Nov. 2012.

[38] J. Hamilton, "Cooperative expendable micro-slice servers (cems): low cost, low power servers for internet-scale services," in *Conference on Innovative Data Systems Research (CIDR09)(January 2009)*, 2009.

[39] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, pp. 7–18, 2010.

[40] Google. Google apps service level agreement. [retrieved: March, 2013]. [Online]. Available: http://www.google.com/apps/intl/en/terms/sla.html

[41] Microsoft, "About microsoft online services: Security and reliability," [retrieved: March, 2013]. [Online]. Available: http://www.microsoft.com/online/faq.aspx#security_and_reliability

[42] Amazon. Amazon EC2 service level agreement. [retrieved: March, 2013]. [Online]. Available: http://aws.amazon.com/ec2-sla/

[43] E. Caron, F. Desprez, L. Rodero-Merino, and A. Muresan, "Auto-scaling, load balancing and monitoring in commercial and open-source clouds," in *Cloud Computing: Methodology, Systems, and Applications*, L. Wang, R. Ranjan, J. Chen, and B. Benatallah, Eds. Taylor and Francis Group, LLC, September 2011, ch. 14, pp. 301–324, iSBN: 9781439856413.