DOI 10.12694/scpe.v17i2.1157
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 17, Number 2, pp. 79–90. http://www.scpe.org 

# IMPLEMENTATION OF A HORIZONTAL SCALABLE BALANCER FOR DEW COMPUTING SERVICES

SASKO RISTOV*, KIRIL CVETKOV†, AND MARJAN GUSEV‡

**Abstract.** Cloud, fog and dew computing concepts offer elastic resources that can serve scalable services. These resources can be scaled horizontally or vertically. The former is more powerful, which increases the number of same machines (scaled out) to retain the performance of the service. However, this scaling is tightly connected with the existence of a balancer in front of the scaled resources that will balance the load among the end points. In this paper, we present a successful implementation of a scalable low-level load balancer, implemented on the network layer.

The scalability is tested by a series of experiments for a small scale servers providing services in the range of dew computing services. The experiments showed that it adds small latency of several milliseconds and thus it slightly reduces the performance when the distributed system is underutilized. However, the results show that the balancer achieves even a super-linear speedup (speedup greater than the number of scaled resources) for a greater load. The paper discusses also many other benefits that the balancer provides.

**Key words:** Cloud computing, dew computing, distributed computing, large scale, load balancer, performance

**AMS subject classifications.** 68M14

## 1. Introduction.

Information and communication technology offers a huge support for the business today by speeding up the internal business processes. However, a company must be prepared for the business processes that interact with its customers and handle the peaks in the load [12].

The exponential growth of Internet of Things (IoT) and Big Data processing lead to the necessary scalability of resources at multiple levels. Skala et al. [26] present a three-layer scalable platform to facilitate the rapidly developing complex distributed computer systems and meet the performance, availability, reliability, manageability, and cost.

Today's companies either build their own virtualized data centers (usually enterprises) or rent on-demand resources organized in virtual machine (VM) instances (usually small and medium enterprises) from some cloud service provider. In either way, they require achieving maximum performance with minimized costs. However, the companies with their applications and services are challenged to orchestrate resources in real time in order to serve the dynamic load by their customers. One mechanism to instantly orchestrate the resources in the virtualized environment or cloud computing and maximize the performance by minimizing the utilized hardware resources is to introduce and implement an appropriate load balancing strategy [18].

Adding more computing resources and balancing the load can solve the peaks and various load balancers with different approaches for balancing techniques exist. The efficient and effective load balancer should have satisfied several functional requirements: balancing all incoming clients' packets to the available endpoint servers, having a configurable port where the requests should be sent their, having a pool of active endpoint servers with IP addresses that will handle the incoming packets, and the introduced load balancer between the clients and endpoint servers should improve the overall response time of the clients' requests.

Horizontal scaling is impossible without load balancing. Even more, the cloud computing based on advanced applications, high-performance computing, with huge storage and networking demands, can not be imagined without load balancing. It is one of the four layers in the cloud application architecture [15, 16]. Rimal et al. [19] describe the load balancing techniques that are used by various cloud service providers and solutions, including Amazon, Google, SalesForce, Azure, Eucalyptus, OpenNebula, etc. Most of these techniques are implementations of a conventional Round Robin schema, weighted selection mechanisms, HAProxy, Sticky session, SSL Least Connect, Source address, cluster server load equalization, high-performance protocols over EC2, hardware load balancing, cloud controllers, etc.

*Ss. Cyril and Methodious University, FCSE, Rugjer Boshkovikj 16, 1000 Skopje, Macedonia (sashko.ristov@finki.ukim.mk).
†Ss. Cyril and Methodious University, FCSE, Rugjer Boshkovikj 16, 1000 Skopje, Macedonia (kiril_cvetkov@yahoo.com).
‡Ss. Cyril and Methodious University, FCSE, Rugjer Boshkovikj 16, 1000 Skopje, Macedonia (marjan.gushev@finki.ukim.mk).

Load balancing was a widely used technique prior to the introduction the cloud computing paradigm and the virtualization technique. The performance of load balancing techniques in the cloud is analyzed by several authors [11, 13].

Radojevic and Zagar [17] grouped them into three groups: Session switching at the application layer, Packet-switching mode at the network layer and Processor load balancing mode. On the other hand, Heinzl and Metz [9] classified the load balancers as hardware and software, and commercial and open source.

There are two main techniques in load balancing: centralized on a specific server or distributed on several servers. The former is controlled by a single central node with all the other endpoint servers communicate with it. Bhadani and For example, Chaudhary [3] proposed the Central Load Balancing Policy for Virtual Machines (CLBVM) solution, while the CLBDM solution [17] considers server load and application performance on top of the Round Robin Algorithm. However, the centralized load balancing approaches in cloud computing have design limitation and communication overhead, thus failing to offer full scalability features [1].

Simjanoska et al. [25] proposed an architecture for a Low Level Load Balancer (L3B). Their architecture can distribute server load among several machines that are integrated over the communication link [24]. The proposed architecture is realized on a network level. It dynamically balances the network packets of the clients among all end point servers and after the computation, it forwards the responses to the clients that send the requests. All these transformations are conducted on the network layer, that is, by changing the information in the incoming and outgoing packet's header.

In this paper, we analyze dew computing services build as modular applications, with big data access, including various smart objects and embedded systems, sensors and other rich mobile clients. Wang [28] explains the dew architecture as an extension of the client server concept, which uses cloud server instead of the classical server, but includes a dew server that realizes a lot of processing and works in companion with the cloud server. In this context, a dew server acts as a cloudlet server that includes local data processing [2].

We have developed a naive implementation of the proposed L3B architecture, which unfortunately did not yield the expected results [21]. The successful implementation of redeveloped balancer according to the same L3B architecture was presented in [5]. The results of the conducted experiments not only that confirmed the successfulness of the implementation, but they show a superlinear speedup, that is, we achieved a speedup of up to 6.5 by using two endpoint servers compared to the case when using only one endpoint server. This paper extends the application domain of our centralized L3B balancer, which offers a high level of scalability when the number of end point servers has been increased (scaled). The conducted experiments lead towards very interesting conclusions. The analysis shows the load where the end point services achieve the best performance shown as speed. Again, a superlinear speedup is achieved as load increases. This important result means that the customers will achieve better performance for the same cost, or pay less for scaled resources to achieve the same performance.

The rest of the paper is organized as follows. Section 2 briefly describes the background. The successful implementation of the L3B balancer is presented in Section 3. Section 4 presents the testing methodology that is used in the experiments to prove the L3B scalability when it is used before two different endpoint servers: 1) web servers with static web pages and 2) computation intensive endpoint servers. The results for the former test case are presented and discussed, along with theoretical analysis in Section 5, while the results for the latter test case are presented in Section 6. Section 7 analyzes and discusses the additional benefits that the successful implementation of L3B balancer provides. Finally, the conclusions and plan for future work are presented in Section 8.

**2. Background.** This section briefly describes the architecture that provides dew computing services and the architectural concept with a naive implementation of the L3B balancer.

**2.1. Dew Computing Services.** A typical modern computing scenario realizes the thin client concept with a browser that has access to the Internet and all the processing is realized by cloud servers. This scenario does not use the potential of the processing and storage of the wirelessly connected small personal and mobile devices. The advances in technology can once again return the fat client idea, but now within the cloud computing environment setting the dew computing concept.

Transferring the servers to the edge of the network is not a new concept, it is a basis of the fog computing concept that extends Cloud computing and services to the edge of the network. Stojmenovic and Wen [27]

discuss that the infrastructure of this new distributed computing allows applications to run as close as possible to sensed actionable and massive data, coming out of people, processes, and things. The idea to distribute servers to the edge of the mobile operator eases the transfer of the information from distributed IoT devices to the cloud servers, reducing service latency and improving the overall quality of service (QoS) with location awareness, mobility, and wireless access [4].

There were several other solutions to cope with WAN delays and approach cloud servers. For example, the cloudlet concept uses a smaller data center positioned locally where increased processing demand arises. The main obstacles the personalized mobile devices are facing include poor resources, such as battery life, weight, heat dissipation, etc. These constraints limit the possibilities of efficient use and initiate the need for offload of computationally expensive tasks to nearby servers in cloudlets or cloud servers [2]. The concept of cloudlets, as trusted, resource-rich computers, located near the mobile users was introduced by Satyanarayanan et al. [23].

The fog computing approach actually is an answer of the communication community to the rise of IT business to deliver IT-based services. The mobile operators have found a new way to deliver extended value-added services to their customers by setting computing resources on the edges of their network, not just for communication needs, but also for initial data processing.

The idea behind the dew computing is using the resources as much as possible before the processing is transferred to the cloud server. It uses the dew computing architecture providing micro services in collaboration with macro services, or dew services in collaboration with cloud services.

This is especially important in the IoT era, where a lot of sensors will stream data and require huge data processing with sufficient volume, velocity, veracity and all the characteristics of Big Data.

Dew services usually require smaller resources and, for example, can perform smaller audio, image or data compression tasks.

Another IoT problem concerns the Internet and cloud server availability. In such cases, the user cannot access its data and can hardly synchronize the computing tasks [28]. The cloudlet and dew architecture concepts enable an environment of smaller cloudlet and/or dew servers that can cope with problems of Internet or cloud server availability. Wang and Pan [28] discuss that the dew server and its related databases have two functions to provides the client with the same services as the cloud server and to synchronize corresponding databases.

**2.2. The L3B architectural concept.** The L3B's concept is presented in Fig. 2.1 [21]. The L3B balancer is hosted on a machine in front of a pool of endpoint servers and its main goal is to balance the clients' requests among the available servers. The latency due to balancer's additional layer should be compensated with a faster response time of the endpoint servers.

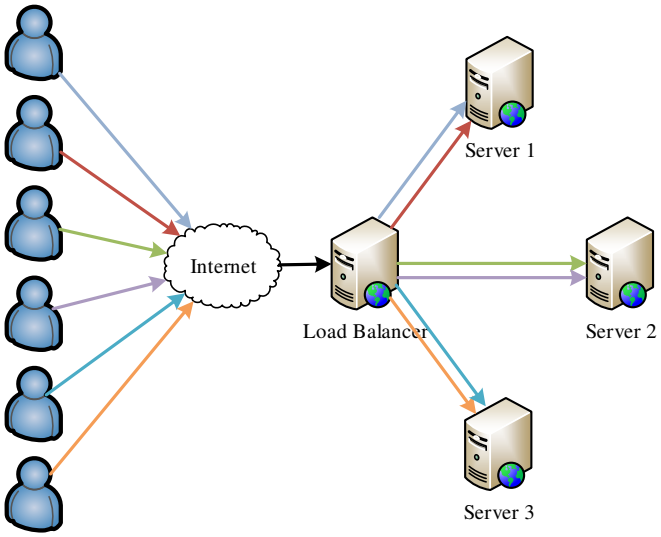


FIG. 2.1. *L3B System architecture*

The L3B balancer is designed with two main modules: *Resource Management Module (RMM)* and *Packet Management Module (PMM)* [25]. The main objective of the former is to provide new available endpoint resources according to the current clients' load and endpoint utilization. PMM balances the incoming packets according to the received information of RMM about available and utilized resources. This paper focuses on the implementation of the PMM module. Its design is presented in Fig. 2.2.

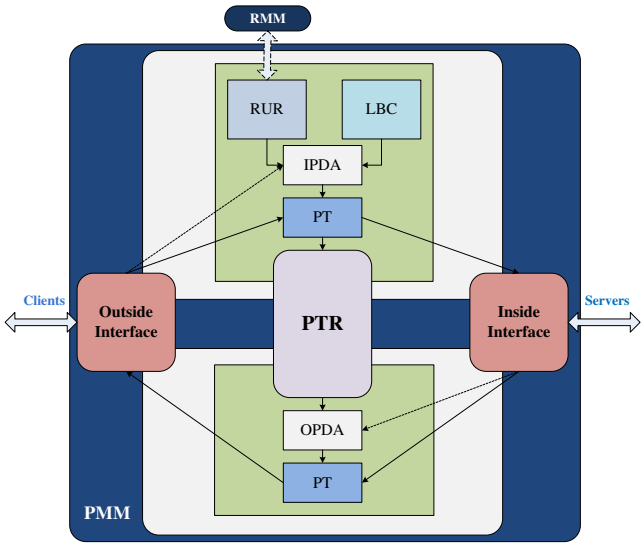


FIG. 2.2. *Design of the PMM module*

PMM manages the client's requests, i.e., it forwards an incoming packet to a particular endpoint by using some balancing technique. After the target endpoint server responds back, PMM forwards again the response to the client that has initiated the request.

When a client request is received at the Outside Interface (OI), it sends information to the Input Packet Decision Agent (IPDA). which decides where to forward the request (which endpoint server), by using relevant information by the Resource Utilisation Repository (RUR) and Load Balancing Configuration (LBC) about current utilisation of the endpoint servers and load balancing configuration, correspondingly. The realized decision is sent to the Packet Translation (PT) agent, which proceed with the IP packet header translation using the NAT/PAT (Network Address Translation / Port Address Translation). These translations are stored in the Packet Translation Repository (PTR) as they can be used to forward the response back to the client. Finally, the transformed packet is forwarded to the Inside Interface (II) in order to be forwarded to the target endpoint server. The response packets are transformed similarly by the PT, by using the decision of the Output Packet Decision Agent (OPDA), according to the information stored in PTR. More details can be found in [25].

The L3B was developed as a Java implementation according to the proposed architecture since modern Java virtual machines' performance is similar to C or C++ [14]. However, the results were not promising [21]. That is, introducing the L3B balancer reduced the performance even worse than the case without balancer. This motivated us to analyze various proposals how to make the L3B more efficient. The next section presents the improved L3B architecture that supports all functional requirements and provides better performance than its predecessor.

**3. The new implementation.** Since the naive implementation of L3B did not yield the expected results, we proceed to improve the L3B implementation. This section presents the development of the new improved implementation.

**3.1. The new implementation's concept.** Fig. 2.3 presents the new improved implementation, based on creation of virtual client socket. When a packet arrives at the L3B server, L3B creates a "Virtual Client",

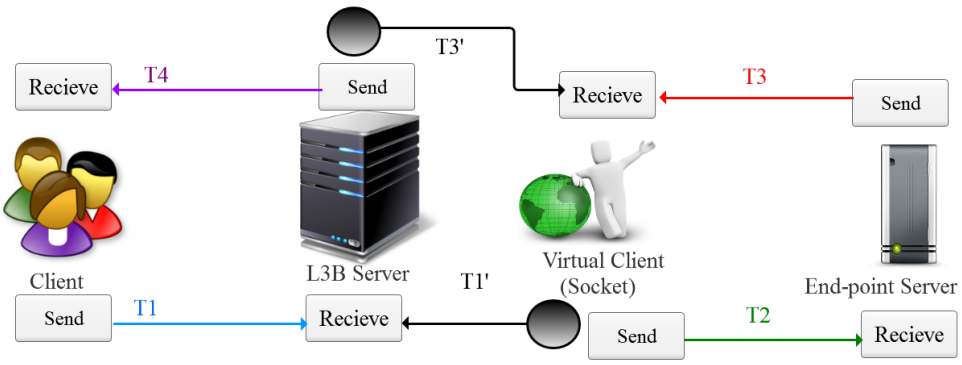


FIG. 2.3. *Improved L3B implementation*

which is a creation of a socket between new client on a certain port, and a port and IP address with the endpoint server, presented as $T_1$ in the figure. Now, the L3B takes the memory (packet) from the received data in the state and prepares for transmission on the server ($T_1'$). The next step is to forward the packet to the endpoint server, shown as $T_2$ in the figure. In the next step, denoted by $T_3$ in the figure, the server processes the request and returns the reply to the virtual client. Then the L3B prepares to send to the client everything that it has previously received and takes its memory ($T_3$ VirtualClient.Send = Balancer.Recieve). Further on, the final activity, denoted by $T_4$ is totally the same as previous case.

**3.2. The implementation details.** A robust, reliable and correct package transmitter is used from source to destination in our concept. Java Network Sockets features are extended in order to realize the load balancing concept.

The implementation is done in Java JDK 7.0, by using network sockets. The algorithm provides package forwarding using Sockets for TCP communication. It will be the same algorithm if a UDP communication is used, but in this case, the Datagram transfer should be used, instead of Sockets.

The details of the implementation are presented in Fig. 3.1. The L3B architecture is implemented with several classes:

- *Client* class, which sends the client's request to Balancer;
- *Balancer* class, which receives request from client;
- *ServerInput* class, which sends client's request to multiple servers;
- *ServerOutput* class, which receives the responses from the servers; and
- *PackageForwarder* class, which creates instances of ServerInput class and ServerOutput class as well as responds with output data to the client.

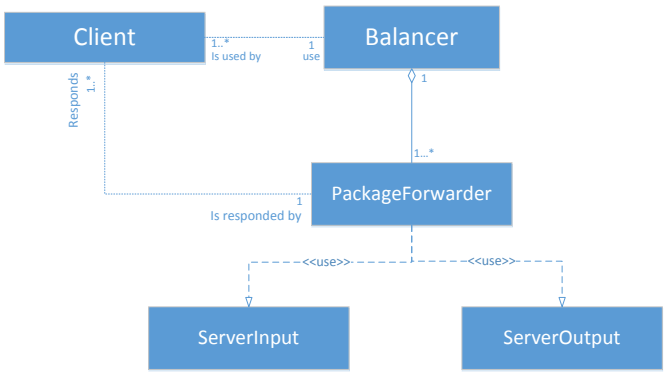


FIG. 3.1. *The details of the implementation*

**4. Testing Methodology.** This section presents the testing methodology that is used in the experiments to prove the L3B scalability. Series of experiments are conducted to determine the performance of the L3B balancer for various load and scaling.

The environment consists of up to 10 same servers, each with Intel(R) Xeon(R) CPU X5647 @ 2.93GHz with 4 cores and 8GB RAM, all connected on 1Gb LAN to exclude the network impact [10]. One server acts as a client, the second as a balancer while others up to eight servers act as endpoint servers. All servers are installed with CentOS 6.5.

Two different type of experiments were conducted:

- Proof of concept for horizontal scaling with an endpoint server that delivers static web page; and
- Proof of concept for horizontal scaling with a computation intensive load as an endpoint server.

**4.1. Testing the L3B balancer for web page end point.** A client requests a packet with constant web page content, which size is $56KB$. The different number of concurrent requests are then initiated to create various loads. We choose this page size in order to be smaller than the IP packet limit of 64KB. Usually, the web requests and responses are smaller than $64KB$.

The concurrent requests are simulated with SOAPUI. Each test case consists of sending $N$ concurrent requests per second, such that $N$ varies in each test case starting from $N = 5$ until $N = 100$ requests, by increasing $N$ with step 5. Each test case lasts 60 seconds.

**4.2. Testing the L3B balancer for computation intensive end point.** The client server is installed with the client benchmark software. It sends an array of 300 integer numbers to the server's listener, which shuffles 10.000 times randomly chosen two elements of the array. The idea of this benchmark tool is to utilize only the resources on the server side, without generating huge network traffic, that is, a situation where the load balancer is applicable.

Total of 9 experiments are conducted with a different number of scaled resources. The first experiment consists of measuring the nominal performance of a single server (an unbalanced endpoint server), while the other 8 experiments are conducted by using $p = 1, 2, \ldots, 8$ servers with one L3B balancer in front of them. Each experiment consists of several test cases, each of which sends different load $N = 25, 50, 75, \ldots, 975, 1000$ (number of concurrent requests) by the client machine.

Several performance parameters are measured and calculated in each test case. *Response Time* $T_iB$ is measured, where index $i$ presents the number of scaled endpoint servers and $B$ denotes that L3B is used in front of those endpoint servers. *Speed* $V_iB = N/T_iB$ expresses the number of handled requests per corresponding response time, while $S = T_1/T_iB$ expresses the *Speedup* achieved in the scaled system compared to the nominal one and $E_i = S_i/i$ expresses the corresponding $Efficiency$. The expected speedup according to the Gustafson's Law is linear ($S_i = i$), while the expected efficiency is 100% ($E_i = 1$).

Several L3B's behaviors will be checked. Firstly, the L3B latency will be checked, that is, the difference between response time $T_1$ of the experiment without L3B and response time $T_1B$ of experiment with L3B that uses also one endpoint server. Secondly, the behavior of the L3B will be examined while loaded with a various number of requests, that is, to check the points where additional scaling should be used. Finally, the regions with the best performance will be analyzed.

**5. Performance analysis of the L3B implementation with web page end point.** This section presents the results of the experiments to confirm the successfulness of the new implementation for web page end point. It also discusses the speedup of the improved L3B implementation. We conduct two experiments based on a configuration using two endpoint servers

**5.1. Results of the experiments for the L3B implementation with web page end point.** Fig. 5.1 shows the results of the experiments that evaluated the performance of the new improved L3B implementation over the system which does not use L3B balancer. The results show that using the L3B is always better for each number of concurrent requests $N > 30$.

Let us explain why the scenario without the L3B balancer (4 CPU cores, 8GB RAM) is better comparing with the scenario with L3B (8 CPU cores and 16GB RAM) for $N \leq 30$, which uses double resources. Introducing L3B adds a constant delay per packet because a packet goes through the additional server (the L3B server) and virtual client (socket). For small server load $N$ this delay is in the range of the server response time. But, when

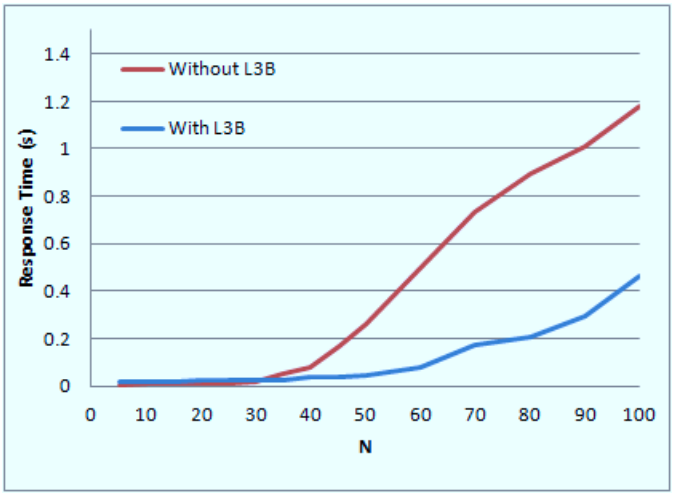


FIG. 5.1. *Results of the experiments for the L3B implementation*

the total load is increased, the single server in the first scenario (without L3B) will be over-utilised, which will decrease its performance while the second scenario (with L3B) will still work in the normal mode with better performance. The next section analyzes the scalability and achieved speedup by using the L3B balancer to balance the load among endpoints in more detail.

**5.2. Theoretical analysis.** According to the Gustafson's law [8], adding more computing and processing resources (CPUs) to some parallel or distributed system will improve its performance limited with linear speedup (the speedup that is equal to the number of scaled resources). The *Speedup* $S(p)$ is defined in (5.1) as a ratio of the response time of the nominal system $T_1$ (without L3B with 1 endpoint server) and the response time of the scaled system $T_p$ ($p$ endpoint servers and the L3B in front of them).

$$S(p) = \frac{T_1}{T_p} \tag{5.1}$$

Our experiment is conducted using L3B with $p = 2$ endpoint servers. That is, $N$ client requests are sent to a single endpoint server (as described in Section 4), while $N$ client requests are sent to two endpoint servers in the second scenario (with L3B), thus balancing the requests by $N/2$ per endpoint server.

**5.3. Case study with scaling factor 2.** Fig. 5.2 presents the speedup that the improved version of L3B achieves over the scenario without L3B. Three regions are observed:

- *Slowdown* region, where the speedup $S(p) = S(2) < 1$;
- *Sublinear* speedup region, where the speedup $S(p) = S(2) < p = 2$;
- *Superlinear* speedup region, where the speedup $S(p) = S(2) > p = 2$;

The slowdown region appears in the range $N < 30$ and the speedup in this region has a constant value. The servers in both scenarios are under-utilised, and the L3B reduces the system performance because it introduces a delay. When a load is increased, the first scenario is over-utilised, while the L3B balances the load and thus keeping both servers still in the normal working mode (going into Sublinear and Superlinear speedup regions). However, when the load reaches $N = 60$, that is, 30 requests per server, both servers become over-utilised and thus the speedup starts to decrease. Increasing the load $N > 100$ started to generate errors in the first scenario.

**5.4. A superlinear speedup phenomenon.** A superlinear speedup region is achieved in distributed environment. The superlinearity is a well-known phenomenon in the cache intensive algorithms [7] because of using more CPU cache memory in parallel implementations compared to the sequential, where the communication among CPUs is low. The superlinear speedup is achieved in the cloud virtual environment, despite the additional virtualisation layer [6, 20].

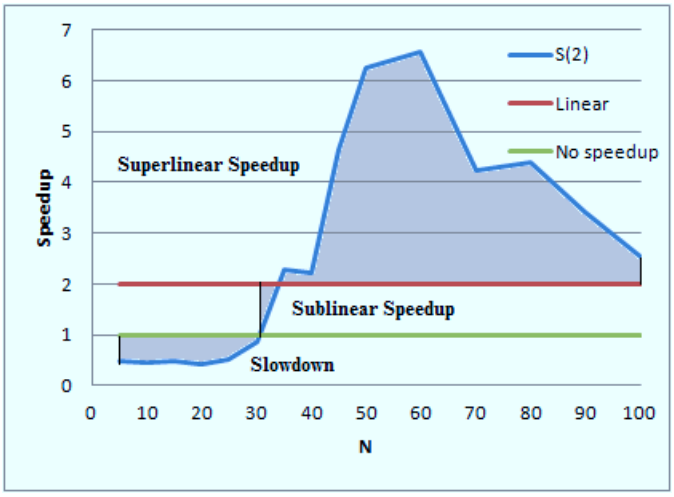


FIG. 5.2. *The achieved speedup of the improved L3B when using two endpoint servers*

However, the reason why the superlinear speedup region appears in this distributed environment is totally different. The nature of web servers is such that increasing the incoming requests dive the server into CPU over-utilisation, while balancing the load among more servers keep the servers in the normal mode, and thus generating the superlinear speedup, despite the delay that the L3B introduces.

**6. Experimental Results for horizontal scaling with a computation intensive load as end point.** This section presents the results of each test case of the experiments for all four test parameters: $T$, $V$, $S$ and $E$.

Fig. 6.1 compares the response time of all experiments. As expected, the response time of scaled systems is lower that the nominal system with only one endpoint and without using load balancing. When comparing the $T_1$ and $T_1B$ curves, one can observe that there is a region ($N < 550$) where $T_1B < T_1$, although it was expected that the L3B server will introduce additional latency. We explain this paradox with the fact that we retain the same session between the balancer and the endpoint, which reduces the utilized RAM memory of the endpoint server.

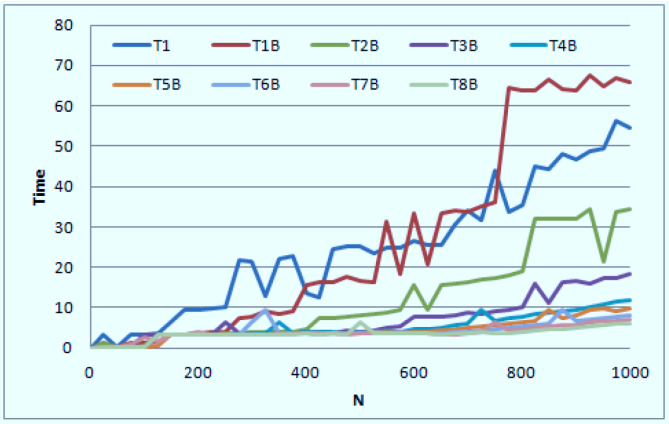


FIG. 6.1. *Execution time for different number of concurrent messages and servers*

The results for the Speed $V_i$ are presented in Fig. 6.2. Three regions are observed for each experiment going from left to the right: A huge peak on the left, a rising region and then a falling region. The first region exists because the fact that was previously explained for the response time. After this peak, the linearly growing of

the speed is observed, which also proves the scalability. The top of the speed is moved to the right for about $N = 100$ for each scaled system. The falling region appears for such load when the existing endpoint servers are not enough to handle the particular load, and in this case, these resources should be scaled.

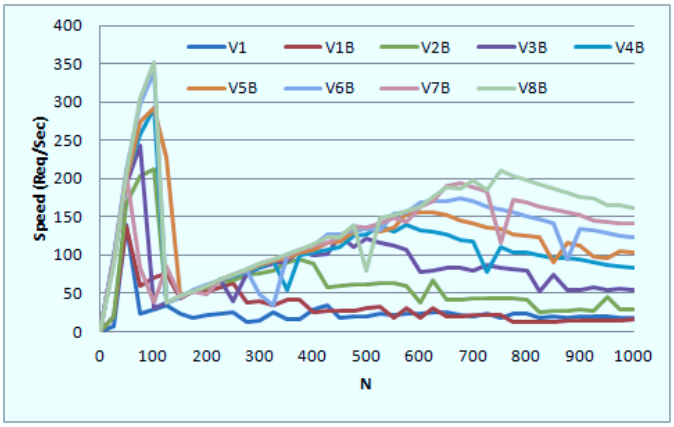


FIG. 6.2. *Speed for different number of concurrent messages and servers*

Fig. 6.3 presents the achieved speedup for scaled systems as a function of the number of messages. Similar regions are observed as those for speed for each scaled experiment compared to the experiment without the balancer. The scalability of the balancer is also observed, that is, the greater speedup is achieved while using greater scaling (more endpoint servers). Even more, the speedup is greater for a greater load. We observe that there is a superlinear speedup region for each experiment, such that each region is moved to the right compared to the experiment with a smaller number of end points.

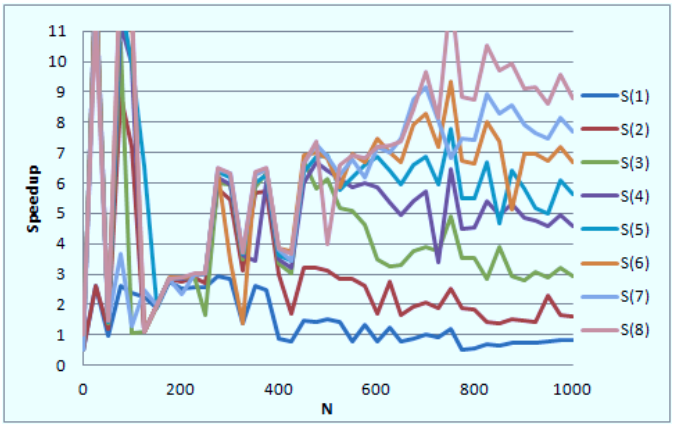


FIG. 6.3. *Speedup for different number of concurrent messages and servers*

The results of the last parameter $E$ is shown in Fig. 6.4. This is the measure of the achieved speedup compared to the number of used scaled resources. That is, Fig. 6.4 shows which experiment achieves the greater performance price trade-off because the price for rented resources (VMs) of the most common cloud service providers is linear.

The Efficiency curves have two different regions: the left region, where mostly the experiments with smaller scaling show superlinear efficiency ($E(i) > 1$ and some region even $E(i) > 2$), and the right region where also superlinear efficiency is achieved, but now those experiments that use greater scaling.

**7. Analysis and Discussion.** This section discusses the benefits that the final implementation of L3B offers. The results that were presented in the previous section show that there are several additional benefits,

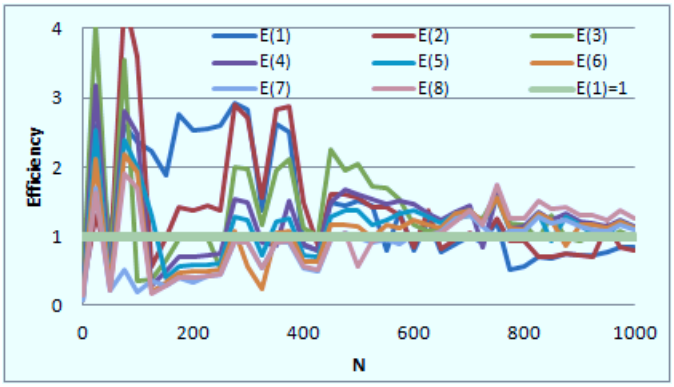


FIG. 6.4. *Efficiency for different number of concurrent messages and servers*

apart of its scalability, along with some further challenges.

The L3B latency is even smaller than the expected one, that is, there is a region where the server is even better by using the balancer in front of only one endpoint. The maximum achieved speedup is 2.93 by using the same endpoint server. This paradox exists because the clients open sessions with the balancer, while the L3B balancer keeps the firstly opened session to the endpoint server and sends the requests through it, which reduces the utilized resources at the endpoint server and improves its performance. Normally, this region ends when the endpoint server is over-utilized, when the performance is reduced and is smaller than the case without the balancer.

Another important benefit is the achieved speedup, which not only that proved the L3B's scalability, but there is a region of superlinear speedup in distributed environment. The superlinearity is a well-known phenomenon, such as the one in parallel systems for cache intensive algorithms [7]. These algorithms are executed in parallel systems that use more CPU cache memory compared to the serial hardware systems. Also, the communication among CPUs is low. Apart of the additional virtualisation layer, the superlinear speedup is also achieved in the cloud virtual environment [20].

However, the reason why the superlinear speedup region appears in this distributed environment is totally different. Ristov et al. [22] modeled the scalable web services by determining five regions, and one of them is the superlinear region. It means that increased number of the incoming requests dive the endpoint server into over-utilisation, while balancing the same client load among increased number of endpoint servers keeps them in the normal mode. Thus, it improves the overall performance and provides the superlinear speedup, despite the latency that the L3B introduces, which is also a positive factor in some region (explained previously).

The maximal achieved speedup is sometimes even as twice as the linear speedup. For example, maximal achieved speedup for the experiment with two endpoint servers is 5.80, rising to 12.26 for the scaling factor of eight. However, this rising of the maximal speedup does not follow the trend with the maximal efficiency, and shows a decreasing trend, from 2.9 for the experiment with two endpoint servers to 1.53 in tests with a scaling factor of eight.

The L3B has a limit in the network packet size. That is, the L3B scalability and additional benefits can be achieved only for the client-server systems where the requests do not exceed the maximal network (IP) packet size. Still, most services, for example, web services, do not exceed it. However, this feature will be implemented in the future and the performance will be measured again.

The testing environment (client-server model and the benchmark tool) is chosen such that L3B balancer will be applicable. Otherwise, for example, if the clients send huge messages with small computation or memory demanding, not only the L3B balancer, but all other centralized balancers would reduce the performance with their latency and bottleneck. However, these systems prefer using a DNS-based (Domain Name Services) balancing the load.

Skala et al. [26] discuss that Dew Computing, which is placed at the lowest level of the distributed computing hierarchy below the fog and cloud computing, offers information-oriented processing rather than data-oriented.

Cloud and fog computing operate on huge amounts of raw data, which is context-free while dew computing is context-aware giving the meaning of data that is processed. They suggest a new context layer in an extended OSI model on top of the application layer to cope with the necessities of the IoT. Our solution for a balancer works on a network level of the OSI model and fits in this scenario.

**8. Conclusion and Future Work.** The presented load balancing technique is intended for horizontal scaling of small scale servers, such as cloudlets and dew servers. Since the naive implementation of the L3B balancer did not yield the expected results, we have improved the implementation and redeveloped the improved L3B, which proved to be scalable with improved performance. Even more, a superlinear speedup region is achieved with the superlinear speedup up to 6.5 when scaling the resources by 2.

This paper presented the case study with introducing L3B in front of only two endpoint web servers, loading them with a single static web page. To test a wider domain of provision of dew services we conducted experiments with a larger number of endpoint servers with computationally intensive tasks. This case study also achieved superlinear speedup, with efficiency greater than two for a smaller load, and greater than 1.5 for those experiments that use greater scaling, which even more improve the contribution of our L3B balancer. We can conclude that the balancer is much better for computation intensive algorithms with a small latency, such as servers providing dew services.

This implementation of L3B promises a lot and thus it will be upgraded even more. An additional plan is to upgrade the intelligent balancer to balance the load keeping the endpoint web servers in the superlinear speedup region where the maximum speedup and the best performance are achieved. Therefore, this solution could be used by the cloud service providers to improve the price - performance trade-off for the scaled resources.

Also, this version of L3B works well only for requests (messages) that do not exceed the size of IP packet. We will upgrade the L3B balancer to support an arbitrary message size, and measure its scalability. In this direction, the RMM module should be implemented to become an elastic load balancer hosted in a cloud due to its scalability.

Additionally, further research will be towards using the L3B in the heterogeneous environment, that is, using the vertical and diagonal scaling, as well, since sometimes they achieve greater speedup than horizontal.

**Acknowledgment.** This work was partially financed by the Faculty of Computer Science and Engineering at the "Ss. Cyril and Methodius University", Skopje, Macedonia.

REFERENCES

[1] 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, July 2011, pp. 163-170.
[2] A. Bahtovski and M. Gusev, *Cloudlet challenges*, Procedia Engineering, 69 (2014), pp. 704-711.
[3] 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, COMPUTE '10, ACM, 2010, pp. 16:1-16:4.
[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, *Fog computing and its role in the internet of things*, in Proceedings of the first edition of the MCC workshop on Mobile cloud computing, ACM, 2012, pp. 13-16.
[5] K. Cvetkov, S. Ristov, and M. Gusev, *Successful implementation of l3b: Low level load balancer*, in Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on, May 2015, pp. 199-203.
[6] M. Gusev and S. Ristov, *Superlinear speedup in Windows Azure cloud*, in Cloud Networking (IEEE CLOUDNET), 2012 IEEE 1st International Conference on, Paris, France, 2012, pp. 173-175.
[7] M. Gusev and S. Ristov, *A superlinear speedup region for matrix multiplication*, Concurrency and Computation: Practice and Experience, 26 (2013), pp. 1847-1868.
[8] J. L. Gustafson, *Reevaluating amdahl's law*, Communication of ACM, 31 (1988), pp. 532533.
[9] S. Heinzl and C. Metz, *Toward a cloud-ready dynamic load balancer based on the apache web server*, in Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2013 IEEE 22nd International Workshop on, June 2013, pp. 342-345.
[10] M. B. Juric, I. Rozman, B. Brumen, M. Colnaric, and M. Hericko, *Comparison of perfor- mance of web services, ws-security, rmi, and rmi-ssl*, J. Syst. Softw., 79 (2006), pp. 689-700.
[11] N. J. Kansal and I. Chana, *Cloud load balancing techniques: A step towards green computing*, IJCSI International Journal of Computer Science Issues, 9 (2012), pp. 238-246.
[12] A. Murua, I. Gonzalez, and E. Gomez-Martinez, *Cloud-based assistive technology services*, in Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on, Sept 2011, pp. 985-989.

[13] 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, Dec 2012, pp. 137-142.
[14] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design*, Fourth Edition: The Hardware/Software Interface, Morgan Kaufmann, 2009.
[15] D. Petcu, G. Macariu, S. Panica, and C. Craciun, *Portable cloud applications - from theory to practice*, Future Generation Computer Systems, 29 (2013), pp. 1417-1430.
[16] D. Petcu and A. V. Vasilakos, *Portability in clouds: approaches and research opportunities*, Scalable Computing: Practice and Experience, 15 (2014), pp. 251-270.
[17] B. Radojevic and M. Zaga, *Analysis of issues with load balancing algorithms in hosted (cloud) environments*, in MIPRO, 2011 Proceedings of the 34th International Convention, May 2011, pp. 416-420.
[18] M. Randles, E. Odat, D. Lamb, O. Abu-Rahmeh, and A. Taleb-Bendiab, *A compara- tive experiment in distributed load balancing*, in Developments in eSystems Engineering (DESE), 2009 Second International Conference on, 2009, pp. 258-265.
[19] B. Rimal, E. Choi, and I. Lumb, *A taxonomy and survey of cloud computing systems*, in INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on, Aug 2009, pp. 44-51.
[20] S. Ristov and M. Gusev, *Performance vs cost for Windows and linux platforms in Windows Azure cloud*, in 2013 IEEE 2nd International Conference on Cloud Networking (CloudNet) ( IEEE CloudNet'13), San Francisco, USA, Nov. 2013.
[21] S. Ristov, M. Gusev, K. Cvetkov, and G. Velkoski, *Implementation of a network based cloud load balancer*, in Computer Science and Information Systems (FedCSIS), 2014 Fed- erated Conference on, Sept 2014, pp. 775-780.
[22] S. Ristov, M. Gusev, and G. Velkoski, *Modeling the speedup for scalable web services*, in ICT Innovations 2014, A. M. Bogdanova and D. Gjorgjevikj, eds., vol. 311 of Advances in Intelligent Systems and Computing, Springer International Publishing, 2015, pp. 177-186.
[23] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, *The case for vm-based cloudlets in mobile computing*, Pervasive Computing, IEEE, 8 (2009), pp. 14-23.
[24] L. Schubert, M. Assel, and S. Wesner, *Resource fabrics: The next level of grids and clouds*, in Computer Science and Information Technology (IMCSIT), Proceedings of the 2010 In- ternational Multiconference on, Oct 2010, pp. 677-684.
[25] M. Simjanoska, S. Ristov, G. Velkoski, and M. Gusev, *L3B: Low level load balancer in the cloud*, in EUROCON, IEEE, Zagreb, Croatia, 2013, pp. 250-257.
[26] K. Skala, D. Davidovic, E. Afgan, I. Sovic, and Z. Sojat, *Scalable distributed computing hierarchy: Cloud, fog and dew computing*, Open Journal of Cloud Computing (OJCC), 2 (2015), pp. 16-24.
[27] I. Stojmenovic and S. Wen, *The fog computing paradigm: Scenarios and security issues*, in Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on, IEEE, 2014, pp. 1-8.
[28] Y. Wang, *Cloud-dew architecture*, International Journal of Cloud Computing, 4 (2015), pp. 199-210.

*Edited by:* Karolj Skala
*Received:* December 21, 2015
*Accepted:* March 31, 2016