Implementation of a Scalable L3B Balancer

Sasko Ristov, Kiril Cvetkov, and Marjan Gusev

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

Abstract. Cloud computing paradigm offers elastic resources that can serve the 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 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 scalable low level load balancer, implemented on the network layer. The scalability is proved with series of experiments, and the results show that the balancer achieves even a super-linear speedup (speedup greater than the number of scaled resources). The paper discusses also about many other benefits that the balancer provides.

Keywords: Cloud computing; distributed computing; large scale; load balancer; performance.

1 Introduction

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 [7].

Load balancing is a technique that balances the clients' requests between two or more end point services hosted in VM instances. The best strategy is to enable the resources to be provisioned automatically without requiring network changes or its configuration [1]. Load balancing allows the service providers to increase the service level agreement (SLA) and also improves the resources usage [4]. Putting a load balancer in front of "tightly" VM instances can improve their overall performance, while the availability will be optimized if the load balancer is implemented in front of "loosely" VM instances [12].

There are many load balancing techniques and they can be classified as centralized or distributed. The former consists of a single central node that communicates with all other nodes. The latter consists of distributed nodes that share the same load balancing algorithm and the generated load by the customers. The performance of several load balancing algorithms is analyzed by Nuaimi [5].

This paper presents a centralized low level load balancer (L3B), which offers a high level of scalability. The conducted experiments lead towards very interesting conclusions. The analysis show the load where the end point services achieve the best performance shown as speed. Additionally, 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 in several sections. The architecture and implementation of the scalable L3B balancer is elaborated in detail in Section 2. Section 3 presents the testing methodology that is used in the experiments to prove the L3B scalability. The results of the experiments are presented and discussed in Section 4. Section 5 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 6.

2 L3B Architecture and Scalable Implementation

This section presents several phases of development of our scalable L3B balancer, presented in developing an architecture and its Java implementation.

2.1 L3B development phases

The L3B's concept relies on a centralized balancer, implemented in front of endpoint computing resources. The goal of the L3B balancer is to balance the load generated of the clients among the available endpoint servers. Although the L3B adds latency due to additional layer, it should be compensated with smaller response of the endpoint servers.

The L3B balancer consists of two main modules: *Resource Management Module (RMM)* and *Packet Management Module (PMM)* [11]. The former provides new available endpoint resources (or reduces them) according to the current clients' load and endpoint utilization. This module could be considered as the configuration module. The latter does the load balancing of the incoming packets among available endpoint resources. This paper focuses to successful implementation of the PMM module, which design is presented in Fig. 1 [11].

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

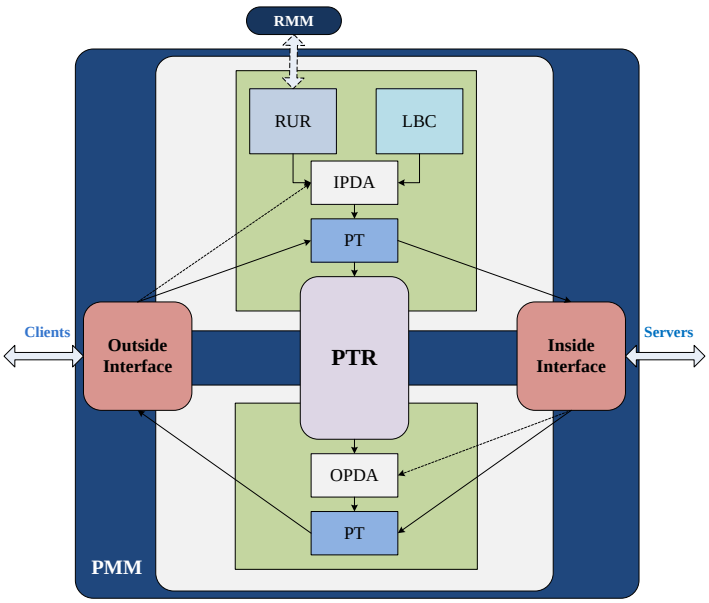

Fig. 1. Design of the PMM module

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 [11].

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++ [6]. However, the results were not promising [8]. 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.

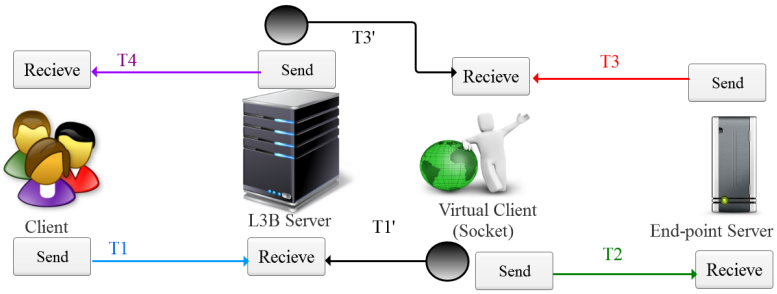

Fig. 2. Improved L3B implementation

2.2 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.

Fig. 2 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", which is a creation of a socket between new client on a certain port, and a port and IP address with the end-point 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 end-point 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.

Robust, reliable and correct package transmitter from source to destination is used for complete conduction of 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. 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 the outputed data to the client.

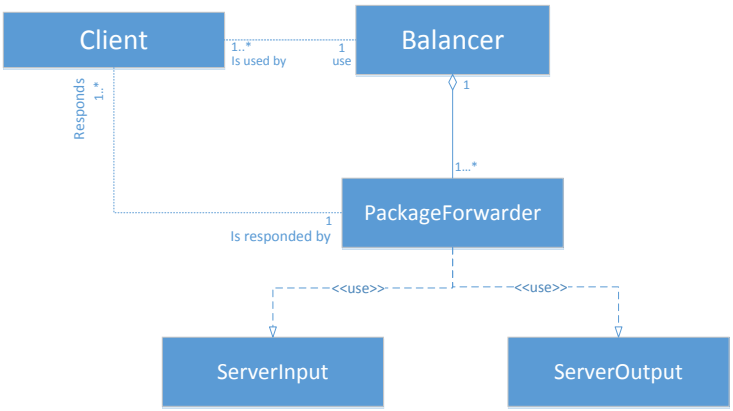

Fig. 3. The details of the implementation

3 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 [3]. 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.

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 at 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 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 experiments 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 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 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 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.

4 Experimental Results

This section presents the results of each test case of the experiments for all four test parameters: T, V, S and E.

Fig. 4 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.

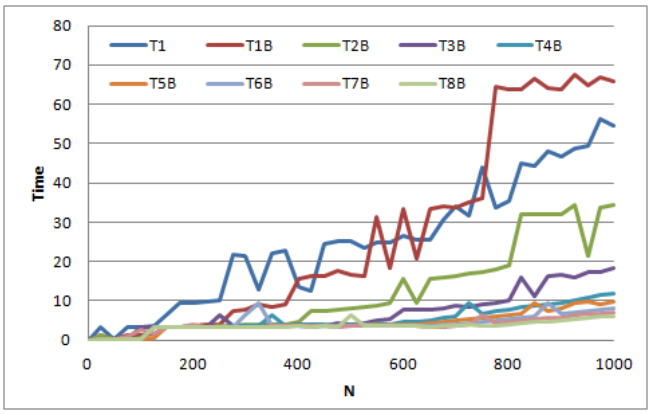

Fig. 4. Execution time for different number of concurrent messages and servers

The results for the Speed V_i are presented in Fig. 5. 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.

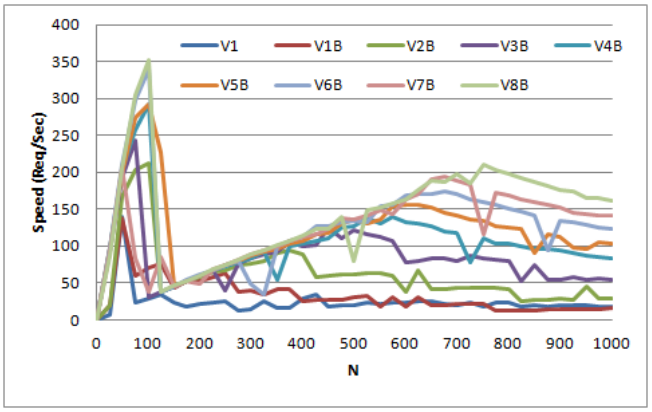

Fig. 5. Speed for different number of concurrent messages and servers

Fig. 6 presents the achieved speedup for scaled systems as a function of number of messages. Similar regions are observed as those for speed for each scaled experiment compared to the experiment without 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 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 smaller number of end points.

The results of the last parameter E is shown in Fig. 7. This is the measure of the achieved speedup compared to the number of used scaled resources. That is, Fig. 7 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.

5 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

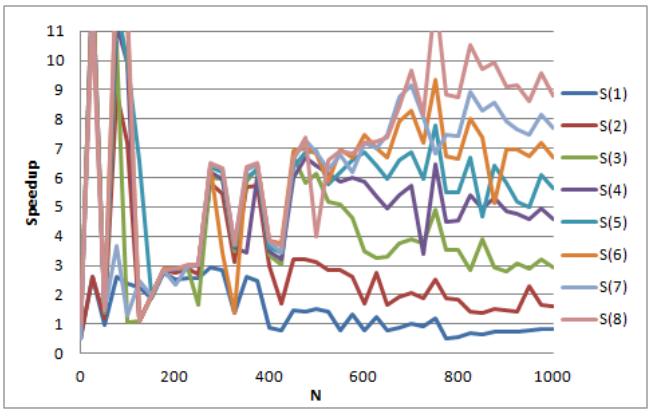

Fig. 6. Speedup for different number of concurrent messages and servers

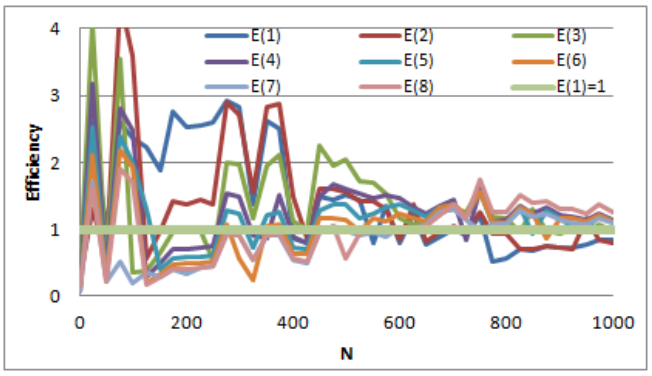

Fig. 7. Efficiency for different number of concurrent messages and servers

several additional benefits, 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 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 [2]. 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 [9].

However, the reason why the superlinear speedup region appears in this distributed environment is totally different. Ristov et al. [10] 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 decreasing trend, from 2.9 for the experiment with two endpoint servers to 1.53 in tests with 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.

6 Conclusion and Future Work

The load balancing is a technique to realize horizontal scaling. This paper presents one successful implementation of a load balancer realized on a low network layer.

The presented solution proved to be scalable when the resources are scaled. The scalability (achieved speedup) increases when the number of resources increases, even to superlinear speedup. Therefore, this solution could be used by the cloud service providers to improve the price - performance trade-off for the scaled resources.

Our future work will be towards implementing additional feature for arbitrary packet size and analyzing the performance for other end point services,

for example, computation intensive, where we expect even better results. Additionally, further research will be towards using the L3B in the heterogeneous environment, that is, using the vertical scaling as well, since sometimes it achieves greater speedup than horizontal.

References

1. Gasior, J., Seredynski, F.: 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. pp. 201–205 (2012)
2. Gusev, M., Ristov, S.: A superlinear speedup region for matrix multiplication. Concurrency and Computation: Practice and Experience 26(11), 1847–1868 (2013)
3. Juric, M.B., Rozman, I., Brumen, B., Colnaric, M., Hericko, M.: Comparison of performance of web services, ws-security, rmi, and rmi-ssl. J. Syst. Softw. 79(5), 689–700 (May 2006)
4. Lee, R., Jeng, B.: Load-balancing tactics in cloud. In: Proceedings of the 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. pp. 447–454. CYBERC '11, IEEE Computer Society, Washington, DC, USA (2011)
5. Nuaimi, K., Mohamed, N., Nuaimi, M., Al-Jaroodi, J.: A survey of load balancing in cloud computing: Challenges and algorithms. In: Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on. pp. 137–142 (2012)
6. Patterson, D.A., Hennessy, J.L.: Computer Organization and Design, Fourth Edition: The Hardware/Software Interface. Morgan Kaufmann (2009)
7. Randles, M., Odat, E., Lamb, D., Abu-Rahmeh, O., Taleb-Bendiab, A.: A comparative experiment in distributed load balancing. In: Developments in eSystems Engineering (DESE), 2009 2nd Int. Conf. on. pp. 258–265 (2009)
8. Ristov, S., Gusev, M., Cvetkov, K., Velkoski, G.: Implementation of a network based cloud load balancer. In: Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on. pp. 775–780 (Sept 2014)
9. Ristov, S., Gusev, M.: 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). USA (Nov 2013)
10. Ristov, S., Gusev, M., Velkoski, G.: Modeling the speedup for scalable web services. In: Bogdanova, A.M., Gjorgjevikj, D. (eds.) ICT Innovations 2014, Advances in Intelligent Systems and Computing, vol. 311, pp. 177–186. Springer International Publishing (2015)
11. Simjanoska, M., Ristov, S., Velkoski, G., Gusev, M.: L3b: Low level load balancer in the cloud. In: EUROCON, 2013 IEEE. pp. 250–257. Zagreb, Croatia (2013)
12. Zhao, Y., Huang, W.: Adaptive distributed load balancing algorithm based on live migration of virtual machines in cloud. In: Proc. of the 2009 5th Int. Joint Conf. on INC, IMS and IDC. pp. 170–175. NCM '09 (2009)