Computationally Intensive Web Service Hosing in the Commercial Virtual Environment

Goran Velkoski[1], Sasko Ristov[2] and Marjan Gusev[3]
Faculty of Computer Sciences and Engineering
16 Rugjer Boshkovikj, Skopje
Macedonia
{velkoski.goran@gmail.com} {sashko.ristov@finki.ukim.mk} {marjan.gushev@finki.ukim.mk}

ABSTRACT: *The web service functions largely rely on virtualization techniques and we conducted experiments in this paper. We initially took two samples for testing the commercial virtual environment. The first one consists memory demand and the second is computationally intensive. We employ the testing system with many compartments which consists of host and guest used to transfer web resources among themselves. The purpose of this effort is to document the drawbacks so as to ensure less virtualization.*

Keywords: Virtualization, Apache Tomcat, Windows, JAVA

Received: 3 March 2022, Revised 7 June 2022, Accepted 19 June 2022

DOI: 10.6025/jisr/2022/13/3/76-83

1. Introduction

The purpose of a virtualization environment is to improve resource utilization by providing integrated operating platform applications based on heterogeneous and autonomous resources aggregation [1]. Virtualization is a popular technique especially by being the baseline for cloud computing [2]. Most cloud service providers use machine virtualization to provide flexible and cost-effective resource sharing [3]. Additionally, the multifarious resource demands imposed virtualization usage on data centers (DCs) as an infrastructure for data storage and deployment platform [4][5].

There are different levels of virtualization: Full Virtualization, Paravirtualization, Operating System-level Virtualization, and Native Virtualization [6].

Commercial and open source virtualization software solutions are owned by some of the most popular ICT companies such as VMware, Citrix, Microsoft etc. [7-9]. Each virtualization software (VMware Infrastructure, Amazon Elastic Compute Cloud – EC2 etc.) operates on top of a layer of system software, called hypervisor or VMM (virtual machine monitor), inserted between the guest operating system and the underlying hardware [10]. Several hypervisors occupy all datacenters and cloud computing solutions. Most popular are VMware ESXi, KVM, Xen, Microsoft Hyper-V etc. [11-15].

The additional virtualization layer degrades the performance compared to the base system especially for HPC clusters [6][16]. In this paper, we analyze the performance of two web services hosted in VMware virtual environment and measure the performance drawback generated by the additional virtualization layer. We conduct series of experiments for compute and memory web services (WSs) on the same hardware infrastructure hosted on bare metal and virtualized environment. We have set the hypothesis that the virtualized environment will degrade the WSs performance.

The rest of the paper is organized as follows. In Section 2, we describe the methodology used for testing. The experiments and the results are presented and discussed in sections 3 and 4. In Section 5, we derive conclusions from the results and we present our plans for future work.

2. The Testing Methodology

In this section, we present the testing methodology we used to produce reliable testing results. We describe the testing platform along with the infrastructure setup, and the differentiated experiment design and test cases.

2.1. Experiment Environment

The testing environment is based on client-server web service architecture. Figure 1 depicts both bare-metal and virtualized experiment environments. For experimental purposes we setup two distinctive server platforms on the same infrastructure that consist of Intel(R) Xeon(R) CPU X5647@2.93GHz with 4 CPU cores and 8GB RAM. We use an Apache Tomcat 6.0 application server installed on Windows Server 2008 to host our Java based WSs named: Concat and Sort described in more detail in the next Section II.B.

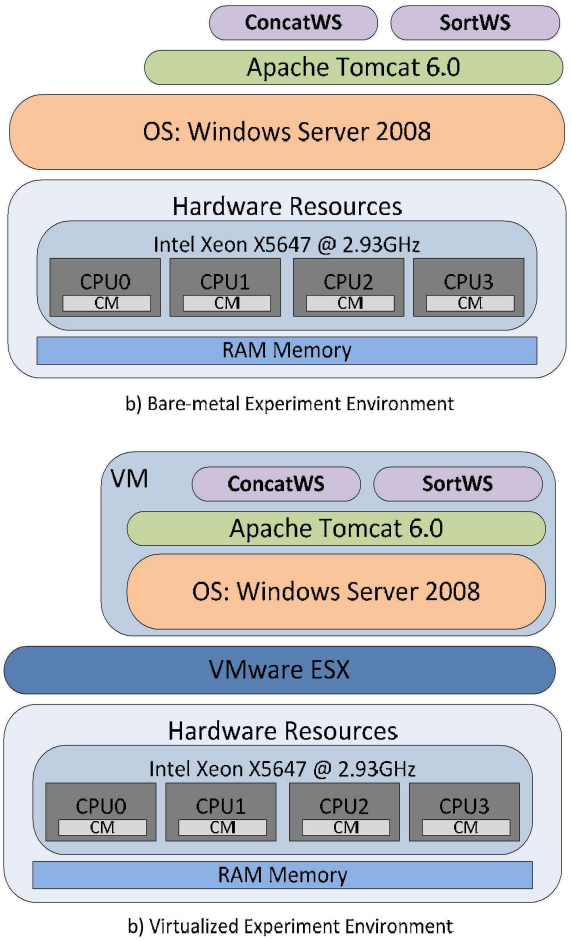

Figure 1. Experiment environments

The difference between the scenarios depicted in Figure 1(a) and Figure 1(b) is the additional virtualization layer which is included only in the virtualization scenario. The servers are installed with VMware ESXi 4.1 and a virtual machine (VM) instance is instantiated and allocated with maximum available resources of the physical machine, i.e., 4 CPU cores and 8GB RAM.

The client uses SoapUI [17] to generate server load with a different number of concurrent messages with various size. The client software is deployed on a different machine consisted of the same Intel(R) Xeon(R) CPU X5647 @2.93GHz with 4 cores and 8GB RAM and placed in the same LAN segment with the servers to minimize network latency and to assume that the measured SoapUI response time is the same as the server response time [18], i.e., the network latency can be neglected.

2.2. Experiments and Test Cases Definition

We use two simple WSs, i.e., Concat and Sort. The former is memory demanding WS which accepts two strings and returns their concatenation. The latter accepts two strings and returns their alphabetically sorted concatenation, which makes it computationally intensive beside its memory demands. We focus on simple WSs since our goal is not to analyze the performance of the real life web services on n-tier application where other factors will impact on the performance and bottlenecks can appear, but focus only on the virtualization impact on the performance.

We define four experiments in order to achieve the reliable comparison, i.e., two web services hosted on two different platforms:

• Experiment 1 – Concat WS on bare metal machine;

• Experiment 2 – Sort WS on bare metal machine;

• Experiment 3 – Concat WS on virtualized machine;

• Experiment 4 – Sort WS on virtualized machine.

Each experiment executes several test cases, such that each test case loads the Apache web server with particular number of concurrent messages and their size.

Each test case runs for 60 seconds. Web servers in VM instances are loaded with N messages with parameters size of PS kilobytes each, with variance of 0.5, that is, the number of messages varies with N/2, starting from N, to 3N/2, and then N/2. The range of parameters PS and N is selected such that web servers in VM instances work in normal mode without replying any error messages.

Parameter size PS is measured in KB for values 0, 1, 2, …, 9KB for Concat WS and 0, 1, …, 6 KB for Sort WS. Both Concat and Sort WSs are loaded with N = 12, 100, 500, 750, 1000, 1250, 1500, 1750 and 2000 requests per second for each parameter size PS.

2.3. Analysis metrics

In order to compare the WS performance based on the response time (RT) measured in milliseconds we introduce the Response Time Relation (RTR). We define RTR as a relation between measured RT for WSs hosted on bare metal (bm) and virtualized platforms (V).

$$RTR = \frac{RT_{bm}}{RT_V} \tag{1}$$

Furthermore, we will use RTR to identify the regions where one or the other environment offers better performance for memory demanding WS (Concat) and both memory demanding and computation intensive WS (Sort).

3. Experiment Results

In this section we present the results of the experiments defined previously in Section 2.2. Additionally, we analyze the results

based on both message sizes, and the number of concurrent messages' impact on the WS performance.

3.1. Experiment 1 - Concat WS hosted on bare Metal Machine

Figure 2 depicts the performance of Concat WS based on the measured response time when hosted on bare metal machine, for different PS with constant number N, and the opposite, constant PS with varying the number of messages N. We observe that both input parameters PS and N negatively impact to the WS performance, i.e. increasing one of the parameter will slightly degrade the Concat WS performance. When messages with size of 9KB are used, the response time fluctuates because we are near to our setup platform limit.

3.2. Experiment 2 - Sort WS hosted on bare Metal Machine

Figure 3 depicts how the Sort WS performance depends on the input parameters PS and N while hosted on bare metal

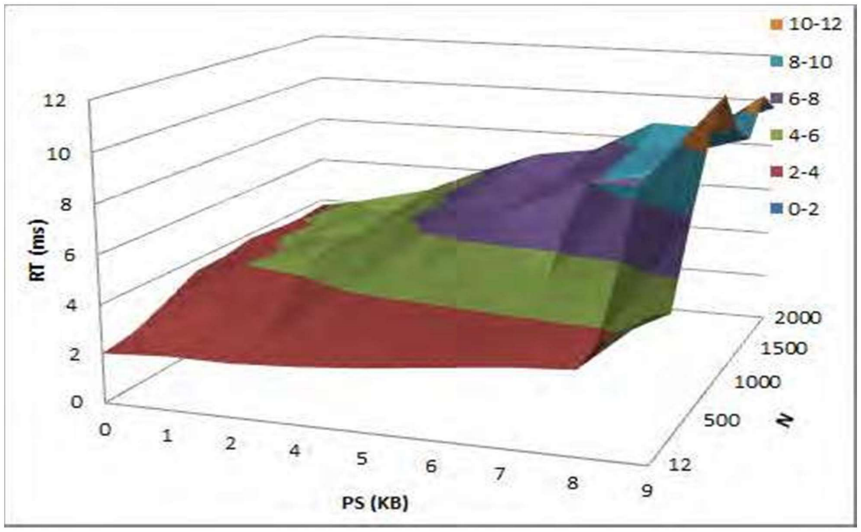

Figure 2. Concat WS response time on bare metal machine

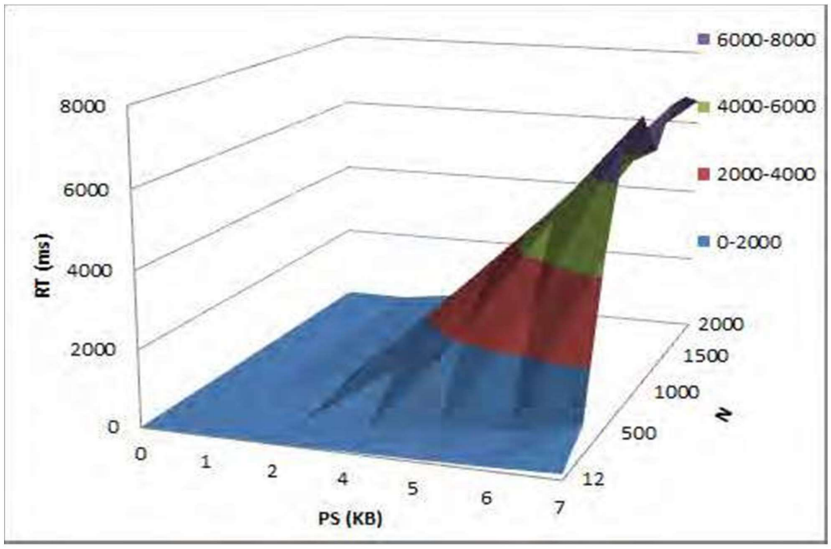

Figure 3. Sort WS response time on bare metal machine

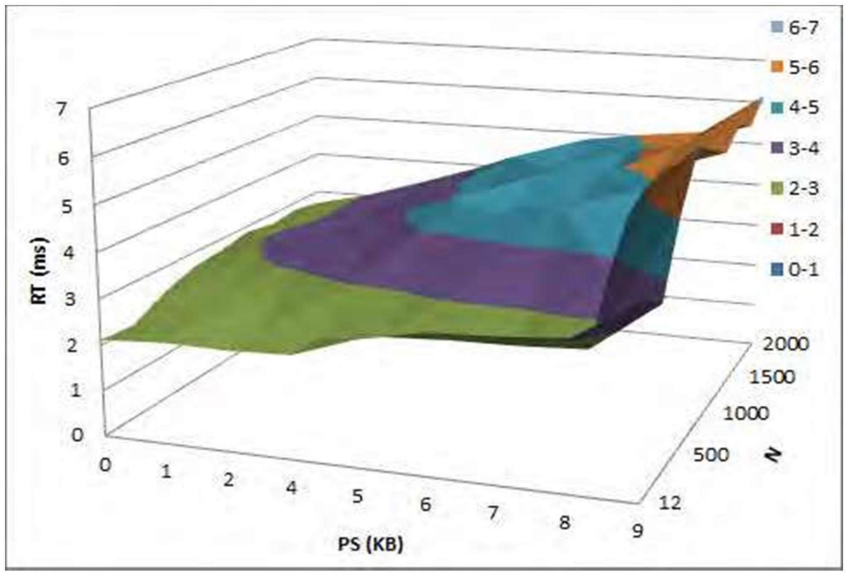

Figure 4. Concat WS response time in virtual environment

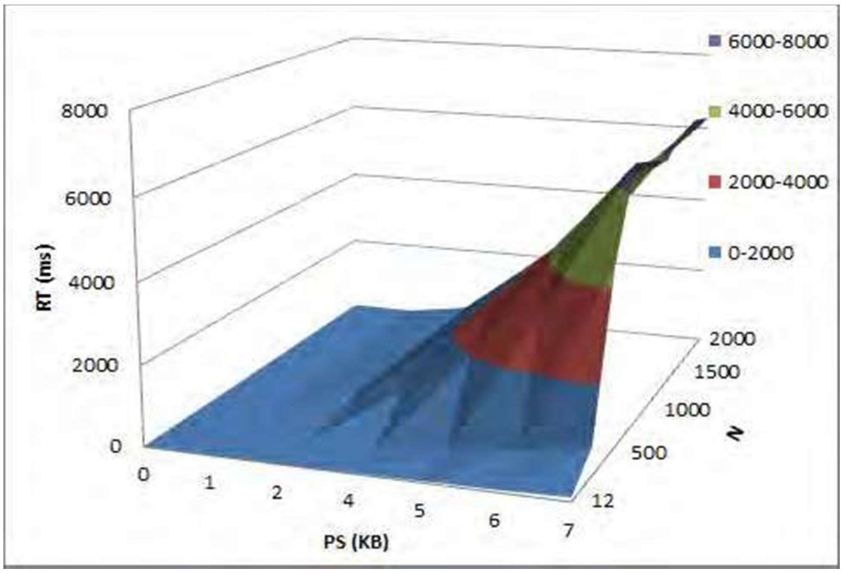

Figure 5. Sort WS response time on virtual machine

machine. The performance is weighted while one of the parameter is constant and the other varies.

In this experiment, the increase of N results with steeper RT growth compared to the increase of PS. This is due to the quick CPU usage increase because of the parallel invocation of the computation and memory demanding web service. We observe that Sort WS has greater response time than Concat WS in range of seconds instead of milliseconds for Concat's response time.

3.3. Experiment 3 - Concat WS hosted on virtual machine Figure 4 depicts the performance of Concat WS based on measured response time when hosted on ESX powered virtual machine. The performance is weighted out based on different payload i.e. progressive parameter PS with constant *N*, and the opposite, i.e., constant parameter PS with progressive *N*.

We observe that both input parameters PS and N slightly impact the Concat WS performance, i.e. increasing one of the parameters will slightly degrade the Concat WS performance in a virtual environment.

3.4. Experiment 4 - Sort WS hosted on bare metal machine Figure 5 depicts the results of the test cases in Experiment 4, i.e., the Sort WS performance while hosted on ESX Figure 4. Concat WS response time in virtual environment Figure 5. Sort WS response time on virtual machine powered virtual machine instance. The performance is weighted out based on different payload i.e. progressive parameter *PS* with constant *N*, and the opposite, constant parameter PS with progressive *N*.

In this experiment, the increase of N results in steeper RT growth than the increase of PS. This is also due to the increased CPU usage because of the parallel invocation of the computation and memory demanding web service.

4. Discussion

In this section we analyze the impact of the virtualization on the performance for a particular WS, by varying the parameters PS and N. We measure the introduced variable RTR for different values of parameters PS and N.

4.1. Virtualization Impact on Concat WS Performance

Figure 6 depicts the comparison of the experiments 1 and 3 by analyzing the parameter RTR for Concat WS.

Despite the hypothesis that bare metal environment has better performance than virtual, we observe the opposite in the most test cases, i.e., the virtualized environment is better for the majority of the tests (RTR > 1). Small accidental regions when executed using small number of WS invokes are slightly below the *RTR* = 1. By increasing the parameters PS or N the value of RTR parameter increases, i.e. the performance of the virtual environment increases compared to bare metal.

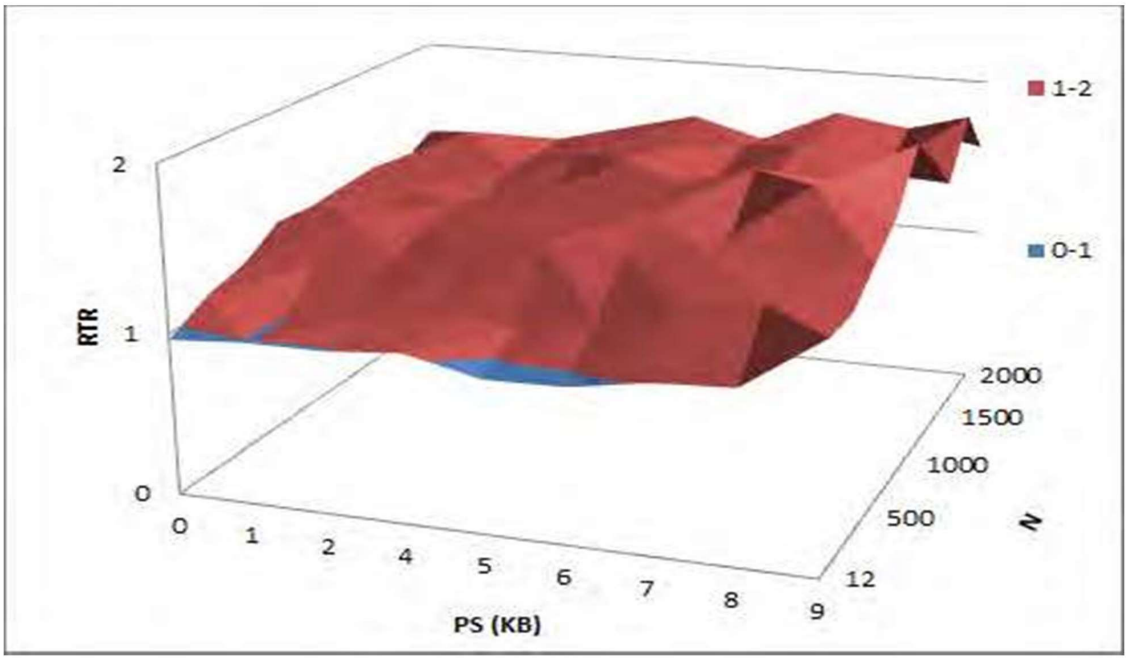

Figure 6. *RTR* for Concat WS

4.2. Virtualization Impact on Sort WS Performance

This section compares the experiments 2 and 4 by analyzing the parameter RTR for Sort WS, as depicted in Figure 7. The results are similar as for Concat WS, i.e. the virtual environment has better performance in most of the test cases. Opposite to Concat WS, the value for RTR for Sort WS is constant regardless of the parameters PS or N. We also observe some test cases where local extremes exist for RTR and RTR < 1.

5. Conclusion and Future Work

In this paper we analyze the performance of memory demanding Concat WS and both memory demanding and computation intensive Sort WS. A correlation is determined between the performance and the input parameters: the number of messages and their size. Both web services are hosted in two different environments, i.e., the bare metal and virtual environment.

The results show that the performance of both web services depends on both of the input parameters. The performance of Concat WS slightly decreases compared to the performance of Sort WS, for both environments.

We observed a phenomenon related to the comparison of the environments. That is, despite the virtualization layer, virtual environment provides better performance for almost each test case (each number of concurrent messages and each size) for both WSs.

We will analyze whether this phenomenon appear for other web services, WSs, and operating systems.

References

[1] Sahoo, J., Mohapatra, S. & Lath, R. (2010) Virtualization: A survey on concepts, taxonomy and associated security issues, computer and network technology (ICCNT) Second International Conference on, pp. 222–226, April 2010 Fig. 7. RTR for Sort WS.

[2] PRNewswire, three key points not to be overlooked in your virtual strategy (2013) [Online]. Available: www.prnewswire.com/news-releases/three-key-pointsnot-to-be-overlooked-in-your-virtual-strategy-119312769.html.

[3] Wang, G. & Ng, T.S.E. (2010) The impact of virtualization on network performance of amazon ec2 data center. Proceedings of the IEEE. IEEE. Infocom.

[4] Bari, M.F., Boutaba, R., Esteves, R., Granville, L.Z., Podlesny, M., Rabbani, M.G., Zhang, Q. & Zhani, M.F. (2012) Data center network virtualization: A survey. *IEEE Communications Surveys and Tutorials*, 15, 909–928 [DOI: 10.1109/SURV.2012.090512.00043].

[5] Goiri, Í., Berral, J.L., Fitó, J.O., Julià, F., Nou, R., Guitart, J., Gavaldà, R. & Torres, J. (2012) *Energy-efficient and multifaceted resource management for profit-driven virtualized data centers. Future Generation Computer Systems*, 28, 718–731 [DOI: 10.1016/j.future.2011.12.002].

[6] Chaudhary, V. et al (2008) A comparison of virtualization technologies for HPC. Advanced Information Networking and Applications. 22nd International Conference on IEEE. Arctic Institute of North America, p. 2008.

[7] VMware (2013) [Online]. Available: www.vmware.com/.

[8] Citrix (2013) [Online]. Available: www.citrix.com/.

[9] Microsoft (2013) [Online]. Microsoft Website. Available: www.microsoft.com/en-us/default.aspx.

[10] Crosby, S. & Brown, D. (2006) The virtualization reality. Queue, 4, 34–41 [DOI: 10.1145/1189276.1189289].

[11] VMware, VMware vSphere Hypervisor™ (2013) [Online]. Available: www.vmware.com/products/vspherehypervisor/overview.html.

[12] KVM, kernel based virtual machine (2013) [Online]. Available: www.linux-kvm.org/page/Main_Page.

[13] Xen, the Xen Project (2013) [Online]. Available: www.xen.org/.

[14] Microsoft & Microsoft (2012) Apr 2013 [Online] Hyper-V server. Microsoft Website. Available: www.microsoft.com/en-us/

servercloud/hyper-v-server/default.aspx.

[15] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. & Warfield, A. (2003) Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 37, 164–177 [DOI: 10.1145/1165389.945462].

[16] Youseff, L. et al Paravirtualization for HPC systems. Frontiers of high-Performance Computing and networking- ISPA (2006). Workshops. Springer: Berlin, Heidelberg.

[17] Soap, U.I. (2013) [Online]. *Functional Testing Tool for Web Service Testing*. Available: www.soapui.org/.

[18] Juric, M.B., Rozman, I., Brumen, B., Colnaric, M. & Hericko, M. (2006) Comparison of performance of web services, ws-security, rmi, and rmi–ssl. *Journal of Systems and Software,* 79, 689–700 [DOI: 10.1016/j.jss.2005.08.006].