Machine Learning Based Classification of Multitenant Configurations in the Cloud

Monika Simjanoska[1], Goran Velkoski[2], Sasko Ristov[3] and Marjan Gusev[4]

Abstract – **Cloud computing is a new archetype where the hardware resources offered as services are intended to be excessively scalable. The cloud model's ability of sharing hardware resources and services among multiple tenants sets new challenges and issues. In this paper we consider the performance as an issue of the multi-tenant cloud model, since multiple tenants share same hardware resources, applications and database instances. In order to measure the performance, we performed response time analysis of a memory demanding web service, hosted in two distinct experimental multitenant cloud environments. Furthermore, we used the performance data to build robust machine learning based classifier, capable of distinguishing between two multi-tenant configurations. The significance of this approach is in its ability to learn and adapt to various workloads and to prevent the initiation of new virtual machine instances when not necessary.**

Keywords – **Cloud Computing, Machine Learning, Performance**

I. INTRODUCTION

Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data-centers that provide those services [1]. Cloud's resource sharing feature describes the ability of sharing the hardware resources and the services among multiple tenants and makes this new computing paradigm different from the traditional service computing. Bezemer at al. [2] give an overview to some of the key features of multi-tenancy as hardware resource sharing, high degree of reconfigurability, shared application and database instances, and list a number of benefits for companies to achieve higher utilization of hardware resources, easier and cheaper maintenance, lower overall costs, etc.

Virtualization enables isolation of the tenants in multitenant cloud computing environment. However, the tenants are not quite isolated since they share the same cloud resources, such as CPU cache, network interfaces, main memory etc. Therefore, the multitenancy confronts some inevitable issues. Disrupted and non-consistent performance is one of the consequences which results from the sharing of the same infrastructure between multiple users.

In this paper we analyse the performance of a memory demanding web service hosted in different multitenant configurations. We realize a series of experiments with various server loads by changing the message size and the number of concurrent messages to analyse the response time in each of the multitenant environments. Our idea is to find out whether spreading the resources among several smaller virtual machine instances in a different manner, produces two different multitenant configurations in terms of the performance. In this research we propose machine learning modelling of the two multitenant environments performance, in order to build a classifier capable of discrimination between the two classes. The contribution of an approach like this is in its ability to learn and adapt to new server loads, ignoring the accidental peaks and thus avoiding the unnecessary initiation of new virtual machine instances.

The rest of the paper is organized as follows. In Section 2 we briefly present the latest work related to our problem. The methodology for the machine learning approach is presented in Section 3. In Section 4 we present the experiments and the results, and we derive our conclusions in Section 5.

II. RELATED WORK

In this section we present some of the latest work related to machine learning analysis in cloud computing. Even though machine learning techniques can be implemented for solving many cloud computing issues, considering the cloud computing literature, its application is still not widespread. Mostly, the machine learning is used for job scheduling, workload management, energy saving, etc.

Ganapathi et al. [3] present statistical driven modeling and its application to data intensive workloads. The authors used statistics to predict resource requirements for cloud computing applications, which can be used for making decisions including job scheduling, resource allocation, and workload management. Bodik et al. [4] proposed a machine learning approach to predict system performance for future configurations and workloads and find a control policy that minimizes resource usage while maintaining performance. Cloud environments benefit from intelligent virtualized resources. Xiong et al. [5] proposed an intelligent virtualized resources management solution in their machine learning powered SmartSLA. Cloud computing is often hosted on energy consuming data centers. Therefore, energy consumption optimization is another cloud computing and data center issue. Chen et al. [6] adopted machine learning methods for data center workload, thermal distribution and cooling facilities management. The performance prediction

[1]Monika Simjanoska with the University Sts Cyril and Methodius, Faculty of Computer Sciences and Engineering, Skopje, Rugjer Boshkovikj 16, 1000 Skopje, Macedonia

[2]Goran Velkoski is with the University Sts Cyril and Methodius, Faculty of Computer Sciences and Engineering, Skopje, Rugjer Boshkovikj 16, 1000 Skopje, Macedonia

[3]Sasko Ristov is with the University Sts Cyril and Methodius, Faculty of Computer Sciences and Engineering, Skopje, Rugjer Boshkovikj 16, 1000 Skopje, Macedonia

[4]Marjan Gusev is with the University Sts Cyril and Methodius, Faculty of Computer Sciences and Engineering, Skopje, Rugjer Boshkovikj 16, 1000 Skopje, Macedonia

has always been researchers challenge. Li et al. [7] present a CloudProphet that can accurately predict the response time of an on-premise web application that migrates to a cloud. CloudGuide explores which cloud configurations meet performance requirements and cost constraints and also can find new configuration when workload changes [8].

III. The Methodology

In this section we present the methodology for the machine learning approach, the preprocessing and the classification process itself.

A. The Testing Environment

As a testing environment we used client-server architecture deployed in OpenStack [9] open source cloud platform with Kernel-based Virtual Machine (KVM) hypervisor. The client and server nodes are installed with Linux Ubuntu Server 11.04 operating system on a machine using Intel(R) Xeon(R) CPU X5647 @ 2.93GHz with 4 CPU cores and 8GB RAM installed. The server platform in cloud realized by virtual machine instances consists of Linux Ubuntu Server 11.04 operating system and Apache Tomcat 6 as application server. The client and the cloud are placed in the same LAN segment to minimize network latency [10]. The client uses SoapUI [11] to test web services performance varying the server loads. While testing, two experimental multitenant configurations, that host a memory demanding web service, are taken into account:

1. Multitenant cloud environment with 2 VM instances, each with 2 CPU cores (*2x2*), and
2. Multitenant cloud environment with 4 VM instances, each with 1 CPU core (*4x1*).

B. Server Load Simulation

Generation of various server loads was performed using SoapUI tool. Each VM instance is loaded with N messages with parameter size of M kilobytes each, with variance 0.5. The range of parameters M and N is selected such that web servers in VM instances work in normal mode without replying error messages and avoiding saturation. The web service is loaded with N = 12, 100, 500, 752, 1000, 1252, 1500, 1752 and 2000 requests per second for each message parameter size M from 0K to 9K. In order to simulate different connections per core we divide the N concurrent messages in four groups of N = 4 messages each. In order to achieve realistic performance results, we measured various parameters as minimum and maximum response time, the average response time, transactions per second (tps), bytes and bytes per second (bps).

C. Data Preprocessing

According to the multitenant configurations, we separated the data from the testing into two distinct classes, as specified in Section 3.2. The results for each class are organized as follows. For each value of the message size M and number of concurrent messages N, SoapUI generates approximately 200 tests. We organized the data as vectors containing the minimum response time, maximum response time, average response time, tps, bytes and bps. Considering the cardinality of the Cartesian product of the sets M and N is 81 and the number of tests is nearly 200, there are approximately 16,000 vectors for one thread for a single class. However, there are a total of four threads and nearly 64,000 vectors in each of the classes. In order to organize all data into the total of four threads, we calculated the median of the tests that belong to the particular thread, instead of their mean, in order to eliminate the possibility of unreal average driven by any unexpected peaks.

In order to prepare the data for the classification process, we transformed the values calculating their z-scores using (1).

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

The z-score, or, the standardized score, is a measure of how many standard deviations the observation is above or below the mean. This normalization is useful when there are outliers that would dominate if we map the values so that they fall within a small specified range, such as 0.0 to 1.0, i.e., if we normalize the values using the min-max normalization [12].

D. Classification

In this paper we assume that classifying the two classes measurements, is a fitting problem. Therefore, we aim to find a model which fits the observations. Since for every input we know the desired output, we can use supervised learning technique as neural networks, to map between the observations and the set of numeric targets. In this paper we used a two-layer feed-forward network with sigmoid hidden neurons and linear output neurons as depicted in Fig. 1. Given consistent data and enough neurons in its hidden layer, this method can fit multi-dimensional mapping problems. For our problem we used 20 hidden layer neurons. As a training method we used the Levenberg-Marquardt backpropagation algorithm. The results from the classification are given in Section 4.

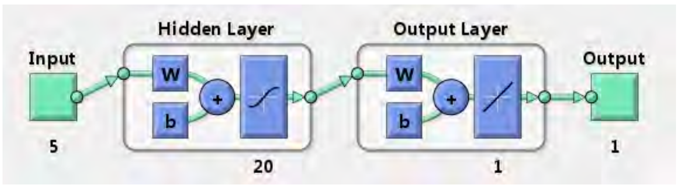

Fig. 1 Two-layer feed-forward neural network [13]

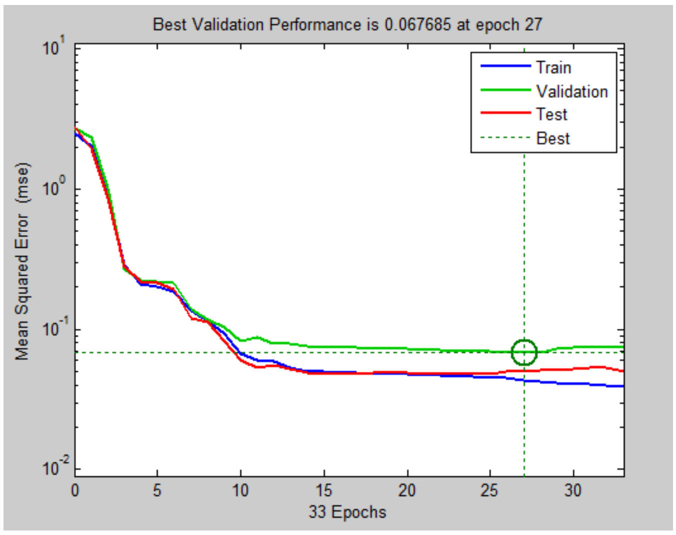

Fig. 2. Classification performance results

IV. Experiments and Results

In this section we present the experiments from the classification and the obtained results.

Organizing the measurements into four threads produced 4 x 81, or, 324 vectors in each class, containing the minimum response time, the maximum response time, the average response time, tps, bytes and bps. However, after few classification experiments, we concluded that the classifier works better if we eliminate the minimum response time as parameter and all further analysis are done without the minimum response time metric.

As we normalized the observations calculating their z-scores, we used the MatLab's neural fitting tool [13] to perform classification analysis as specified in Section 3.4. In our classification we used 70%, or, 454 samples to belong to the training set, 15%, or, 98 samples to the validation set, and also 15% to the testing set. All samples were randomly chosen to belong to each of the sets. Hereupon, we used a neural network with 20 hidden layer neurons and trained it using the Levenberg-Marquardt backpropagation algorithm. The training process finished in 33 iterations when the validation error increased for 6 iterations. The performance of the classifier is measured according to the Mean Squared Error (MSE) algorithm. MSE is the average squared difference between outputs and targets. Fig. 2 depicts the training, validation and the testing errors. Since the best validation performance value is ~0.068, we consider it is low enough to conclude the result is reasonable. Further indicators of promising results are the similarity between the testing and the validation error, and also the insignificant overfitting that occurs after the epoch 27 where the best validation performance occurs.

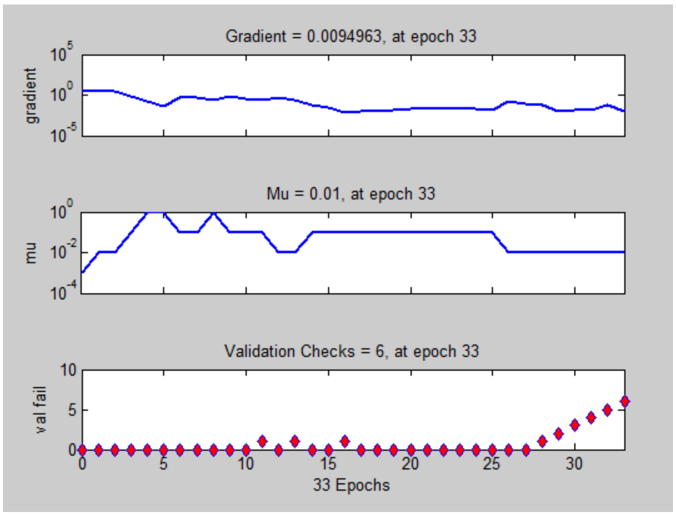

Fig. 3. Training algorithm state

The state of the training algorithm with respect to epochs is depicted in Fig. 3.

A linear regression between the network outputs and the corresponding targets is depicted in Fig. 4. Regression R values measure the correlation between outputs and targets. An R value of 1 means a close relationship, whereas 0 means a random relationship. According to the R values, the output corresponds to the targets good enough for the training, the validation and the testing. The total network response is ~0.904 which satisfies our expectations. All the results are presented in Table I.

TABLE I
CLASSIFICATION RESULTS

Results	Samples	MSE	R
Training	454	~0.043	0.91
Validation	98	~0.067	0.86
Testing	98	~0.049	0.89

V. Conclusion and Future Work

In this paper we perform series of experiments to simulate different multitenant cloud environments. Our aim is to measure the performance impact when sharing the same infrastructure resources among multiple tenants and to perform classification analysis of discrimination between the two types of multitenant configurations. For that purpose we inspect the performance behavior when a memory demanding web service is hosted in multitenant cloud environment with 2 VM instances, each with 2 CPU cores, and a multitenant cloud environment with 4 VM instances, each with 1 CPU core. While evaluating the performance we take into account the minimum response time, the maximum response time, the average response time, tps, bytes and bps, for various numbers of messages different in size. Once we obtained the results we normalized the values and used a two-layer feed-forward network with sigmoid hidden neurons and linear output neurons to perform classification analysis. Considering the high level of similarity between the two classes, the classification results showed satisfying overall response of ~0.904 value of correlation between the outputs and the targets.

We believe that the contribution of the machine learning approach for our problem is in its ability to learn and adapt to new server loads, ignoring any accidental peaks. Moreover,

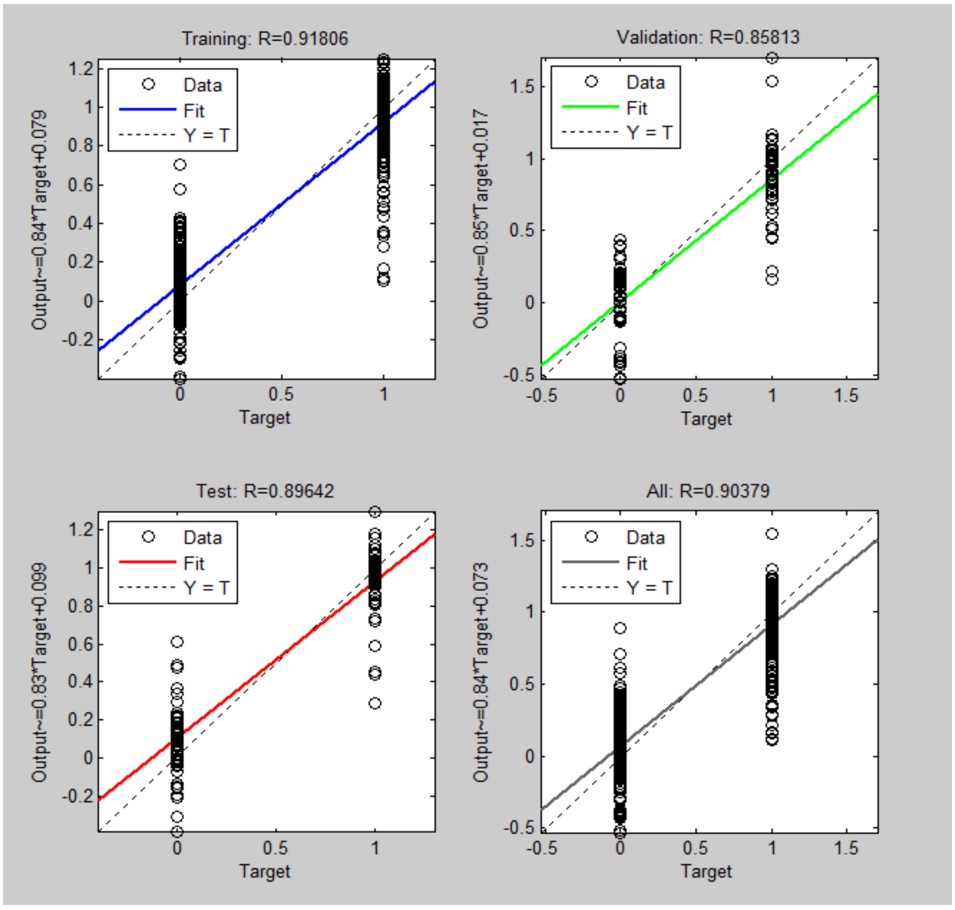

Fig. 4. Regression Analysis

this research aims towards real time decision of new virtual machine instances initiation.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “A view of cloud computing,” Commun. ACM, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[2] C.P. Bezemer and A. Zaidman, “Multi-tenant saas applications: maintenance dream or nightmare?” in Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE), ser. IWPSE-EVOL ’10.

[3] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, “Statistics-driven workload modeling for the cloud," in Data Engineering Workshops (ICDEW), IEEE 26th Int. Conference on, 2010, pp. 87-92.

[4] P. Bodik, R. Griffith, C. Sutton, A. Fox, M. Jordan and D. Patterson, “Statistical machine learning makes automatic control practical for internet datacenters,” in Proceedings of the 2009 Conference on Hot topics in cloud computing, 2009, pp. 12-12.

[5] P. Xiong, Y. Chi, S. Zhu, H. J. Moon, C. Pu and H. Hacigumus, “Intelligent management of virtualized resources for database systems in cloud environment,” in Data Engineering (ICDE), IEEE 27th International Conference on, 2011, pp. 87-98.

[6] H. Chen, P. Kumar, M. Kesavan, K. Schwan, A. Gavrilovska, and Y. Joshi, “Spatially-aware optimization of energy consumption in consolidated datacenter systems,” Proceedings of InterPACK, Portland, OR, 2011.

[7] A. Li, X. Zong, S. Kandula, X. Yang, and M. Zhang, “Cloudprophet: towards application performance prediction in cloud,” SIGCOMM Comput. Commun. Rev., vol. 41, no. 4, pp. 426–427, Aug. 2011.

[8] S. H. Liew and Y.-Y. Su, “Cloudguide: Helping users estimate cloud deployment cost and performance for legacy web applications,” in Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on, dec. 2012, pp. 90 –98.

[9] OpenStack, “Openstack cloud software,” Jan. 2013. [Online]. Available: http://openstack.org

[10] M. Juric, I. Rozman, B. Brumen, M. Colnaric and M. Hericko, “Comparison of performance of web services, WS-Security, RMI, and RMI-SSL,” J. of Systems and Software, vol. 79, no. 5, pp. 689-700, 2006.

[11] SoapUI, “Functional testing tool for web service testing,” Jan. 2013. [Online]. Available: http://www.soapui.org/

[12] J. Han and M. Kamber, “Data Mining: Concepts and Techniques,” Second edition, Elsevier, 2006.

[13] MATLAB, 2013. Available: http://www.mathworks.com