Non-Invasive Estimation of Cloud Applications Performance via Hypervisor's Operating Systems Counters

Fábio Diniz Rossi, Israel Campos de Oliveira,
César A. F. De Rose
Faculty of Informatics
Pontifical Catholic University of Rio Grande do Sul
Porto Alegre/RS, Brazil
{*fabio.diniz, israel.oliveira*}*@acad.pucrs.br, cesar.derose@pucrs.br*

Rodrigo Neves Calheiros, Rajkumar Buyya
The Cloud Computing and Distributed Systems Lab
Department of Computing and Information Systems
The University of Melbourne, Australia
{*rnc, rbuyya*}*@unimelb.edu.au*

***Abstract*—The adoption of cloud computing environments as the infrastructure of choice for computing services is growing rapidly, due to features such as scalability and pay-per-use. As a result, more pressure is put on cloud providers, which manage the underlying computing platform, to maintain the Quality of Experience of application users within acceptable levels. However, the mapping of high-level application metrics, such as response time, to low-level infrastructure metrics, such as utilization rate of resources, is a non-trivial task. Many works present monitoring of processor, memory, and network utilization. Nevertheless, the monitoring of these resources can be intrusive to the system that provides the service. This paper presents a non-invasive approach for estimating the response time of cloud applications through the mapping of Quality of Service metrics to operating system counters at the hypervisor level. We developed a model that estimates the response time of real-time applications based on Linux Operating Systems counters that presented an accuracy of 94% in our evaluation.**

Keywords-Cloud Computing; IaaS; QoE; Response time; SLA.

I. INTRODUCTION

Cloud computing and cloud storage have become the preferred methods for distributing information and online functionality over the Internet [1]. While some cloud service providers focus on providing customers with a broad range of features and services, including online shopping, search, social network, entertainment consumption, and protecting important documents, other cloud service providers focus on providing services for small businesses, large corporations, governments, and other institutions.

Most of these environments need to improve Service Level Objectives (SLOs) and meet Service Level Agreements (SLAs) in terms of availability, performance, security, and data protection [2]. This is important, as it impacts directly on the Quality of Experience (QoE) of users, who may or may not remain loyal to the offered services [3] [4].

With the aim of offering services that comply with these high level quality metrics established without excessive operational costs, cloud service providers use rapid elasticity provided by cloud computing [5]. Thus, it is possible to dynamically increase or decrease instances of virtual machines and/or compute nodes, as well as the applied quota of CPU, memory, and network bandwidth on a cloud service. Besides the obvious benefits of cost and performance for users, cloud providers can also benefit from a more efficient use of resources.

Elasticity, the capacity to dynamically change the amount of resources dedicated to a service, for more or less, is controlled via pre-defined SLAs. When these SLAs limits are exceeded, new resources are added so that the load returns to an acceptable level. When resources are underutilized, resources can be freed in order to reduce operational costs. Nevertheless, the decision on the amount of resources required to meet high-level metrics defined as SLAs is non-trivial [6], because some high-level metrics can not be easily monitored by the infrastructure. As the SLA [7] involves the definition of minimum acceptable levels of service that are expected by the customer, it is common the use of indicators for the quantitative measurement of the Quality of Service (QoS) received. Some commonly used indicators are availability, response time, and mean time between failure, among others.

As services offered by cloud service providers are accessed over the Internet, it is natural that network QoS metrics are the most important for user experience [8]. Thus, this work focuses on the response time of cloud applications. This is a metric that can influence the decision on the need for more or less resources to maintain an acceptable level of service, and it is measured by the time taken from the client request is received by the service provider until the response by the service provider is sent. However, this can not be monitored by the Infrastructure-as-a-Service (IaaS) without the risk of interference on the communication channel between the client application and the service provider. Furthermore, there are privacy issues that should be taken into account, as most users would not agree with monitoring systems running within their

 ISBN: 978-1-61208-398-8

virtual machines. In addition, there are applications that using non-standard software stacks that do not allow reliable monitoring of the response time.

Therefore, the problem that this paper addresses is how to estimate the response time of cloud applications, based only on information that can be accessed through the infrastructure, and without being intrusive in the communication channel of client applications. Accordingly, the hypothesis tested by this work is that internal operating system counters at the hypervisor level allow estimation of the client application response time based on the history of the load on the physical machine on which the application is allocated.

The aim of this work is not prediction, i.e., the objective of the proposed method is not to infer the response time before it occurs. Rather, our analysis takes place after the event occurred, and it aims at, based on the observed value, to estimate the application performance. This enables perform control actions on scalability, reacting to fluctuations of this metric.

To this end, tests to measure the response time were performed with a real three-tier cloud application. At the same time, system counters were monitored to verify the system load. Finally, we present a model that allows estimating the response time based on historical information about the load on the operating system, and some evaluations of this model.

This paper is organized as follows: Section II introduces a background on response time and performance counters of operating systems; Section III presents related work; Section IV describes preliminary experiments used to fit a new model; Section V presents evaluations; finally, Section VI concludes the paper and addresses future work.

II. BACKGROUND

QoE [8] refers to the user's perception of the quality of services transactions and may be represented by human feelings. On the other hand, QoS refers to a systematic method to evaluate the service, usually via metrics that can be measured and verified and that directly affect the perception of the end user. A QoS metric that directly impacts the QoE is the response time of applications. This section presents a conceptualization of response time and shows some involved algorithms for its calculation.

A. Response Time

Response time can be understood and evaluated in several ways depending on the context. In this section, we will conceptualize and explain the response time in the context of a cloud computing infrastructure. Figure 1 presents a three-tier architecture, in which a customer performs requests to a service in a cloud, and waits for a response.

The time required to complete the entire process between the start of the request from a customer up to all of its

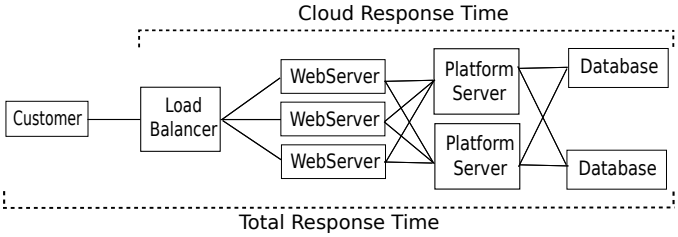

Figure 1. Response Time in Cloud Architectures

response consisting of the total response time as shown in Figure 1. Thus, the total response time includes the time that packets must travel between the customer and the cloud. This means that the cloud manager that is providing the service has no way to measure and ensure the quality of service of this external link, unless there is monitoring from the client. However, a monitoring client-side influences other aspects such as security, privacy, and the actual cost of monitoring on the link.

Therefore, when dealing with response time in cloud infrastructure, we are assuming the Cloud response time. The time spent in establishing the connection operations can be seen in (1), where one-way trip (OTT) time is the difference between the last (ω) synchronization packet (SYN) with the first (α) acknowledgment packet (ACK), divided by the number of participants in the connection.

$$OTT = \frac{\alpha ACK - \omega SYN}{2} \quad (1)$$

Therefore, the Total Real Time (RT) (Figure 2) of each request is the sum of the one-way trip times, and the time that a reply takes to answer a client's request.

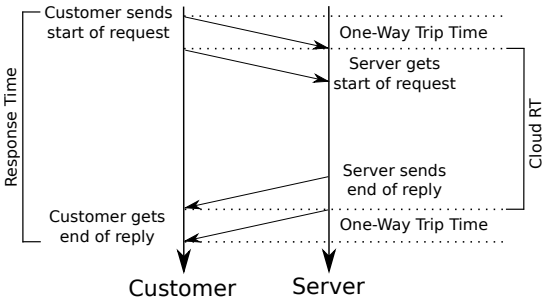

Figure 2. Total Response Time and Cloud Response Time

Figure 2 shows the one-way trip in the beginning and end of the connection establishment, and therefore, the transmission between the client and the server.

In summary, the cloud response time consists of the time that the customer request takes to be processed by the cloud service provider. This is counted from the moment it is received by the cloud application layer, through the business layer, searching and querying data in the database layer, and

 ISBN: 978-1-61208-398-8

returning necessary information for the application layer to be sent to the customer.

B. Operating System Counters

The response time is an ideal metric to verify the quality of a service offered via the network, but its monitoring may cause overhead. Estimate the response time of an application without interference in the channel between the customer and the service can substantially reduce this impact. To this end, we hypothesized that operating system counters can be monitored in order to verify the response time within the cloud data center, and therefore, allow estimating the response time between the customer and the data center is acceptable or not.

In the context of this work, we target real-time Linux counters as the focus of monitoring due to the fact that Linux is widely used in IaaS environments, such as Openstack [9]. Linux stores the counters in a virtual file system referenced in `/proc`. This directory contains, rather than files, a runtime information system in which one can monitor the states and loads of the processor, memory, and devices, in real-time. In particular, we rely on information available at the hypervisor level (rather than virtual machine-level) when the hypervisor is supported by a privileged operating system, which is the case of Xen [10] (on its Dom0).

Because of this, most system management commands search for system information into this directory. Since this information is accessible at the user level, system management tools can display mashups that will support decisions about the use of resources to the system administrator, e.g., system and services [11].

To monitor the load average of I/O in our tests, we used the information provided by `/proc/loadavg`. This file is populated with values collected from the run queue of the operating system. It stores a series of values of load in three intervals representing the average load of the system in the last 1, 5, and 15 minutes. These values are updated by the system every one minute. Since the values are shown in the time period, it provides a good indication on if the workload is increasing or decreasing the use of resources. Moreover, it allows to estimate when the system is overloaded and impacting on QoS metrics.

III. Related Work

Aceto et al. [12] discusses the difficulty of monitoring cloud environments with respect to the mapping of high-level metrics to metrics at the infrastructure level, due to the fact that high-level metrics may include external parameters to the infrastructure, which are not controlled by cloud environment.

Emeakaroha et al. [6] proposed a framework for managing the mapping of low-level resource metrics to high-level SLAs. The scalability of the model was validated using queuing network models, and it was able to detect SLA violations and notify the manager module of the cloud environment.

Dobson and Sanchez-Macian [13] presented a work-in-progress paper proposing a QoS ontology that can be applied on several scenarios of cloud/grid. This model is divided into two parts: QoS monitoring and QoS adaptation.

Rosenberg et al. [14] discussed QoS attributes for web services, identifying the most important attributes and its composition from resource metrics. In addition, the study presented some mapping techniques to compose QoS attributes of resources to generate metrics of SLA parameters for a specific domain.

D'Ambrogio and Bocciarelli [15] present a model-driven approach with the intention to incorporate application performance prediction into a service composition process. The paper shows the composition of SLA parameters, although it does not consider monitoring the SLA.

Comuzzi et al. [16] propose an architecture for monitoring SLAs considering two criteria: the availability of historical data to evaluate the SLA offers and the evaluation of the ability to monitor an offer of SLA.

What differentiates our work from the other ones discussed in this section is the fact that we present a methodology to estimate the response time without impacting on user communication channel, either in terms of performance or in terms of security and privacy. To the best of our knowledge, no other work has mapped information from operating system counters, obtained from the hypervisor, in order to infer the performance of an application running in a virtual machine that directly influences end users' QoE.

IV. Estimating Application Performance

For web applications, particularly in the case of e-commerce, performance tests are essential. A service provider can not define the needed amount of resources required by the application to serve a specific workload based only on average traffic. To ensure that a web application meets certain criteria such as performance, data throughput or response time, test in an environment similar to the production environment is required.

The focus of this paper is on three-tier cloud applications, type of most used application on cloud environments. Therefore, we are targetting a virtualized environment that supports a web interface that accesses, through a business logic layer, a database allocated to another computer system. This architecture is shown in Figure 3.

The first step toward the goal of performance estimation was conducting experiments to enable the modeling of the relationship between response time and the load as indicated by the loadavg counter, which we detail next.

 ISBN: 978-1-61208-398-8

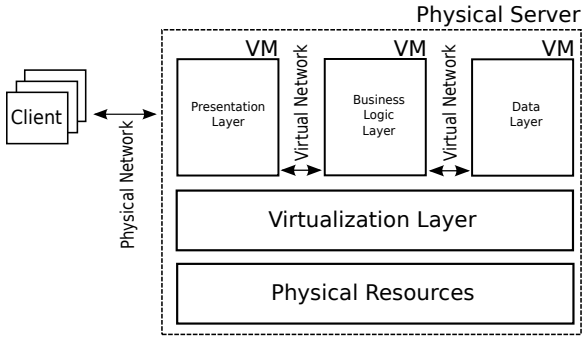

Figure 3. Three-Tier Cloud Architecture

To this end, we deployed a testbed consisting of two physical servers connected by a Gigabit Ethernet network. Each server has two Intel Xeon 2.4 GHz processors (12 cores) and 16 GB RAM. On each server, we deployed three virtual machines, each one supporting the web server (Apache), application server (Tomcat) and database server (MySQL). The CPUs are dedicated for each VM and disks are shared between them. Load average is taken from /proc/loadavg file at the hypervisor level (Dom0).

To represent a multi-layer architecture, the Apache Benchmark was used. This benchmark is widely used to test performance of multi-tier applications. It mimics the users' access to web servers serving as front end for multi-tier applications. Aiming to emulate an elastic cloud environment, we use the HAProxy load balancer that splits the workload between new nodes, when the response time set out in a SLA is exceeded. In our tests, 100 clients should insert 1000 records into a table, query them, and delete them at a maximum of 300 milliseconds using 50% of the network throughput. To control the flow of the network, we use the Linux Traffic Control (TC). This allows adjustment of the network flow and hence impact the response time of applications.

The evaluations were conducted in a total of 45 minutes (2700 seconds). Each test was performed with the size of 15 minutes because this is the maximum time that the loadavg uses to update its values. For the tested architecture, the loadavg values can range from 0 (underused) to 22 (overutilized). We set 300 milliseconds as the target response time of the test application. The values of loadavg and their corresponding timestamps were recorded in intervals of 1 second. The application response time was monitored by a feature of the Apache server, which shows when a request is received from a customer, and when it was responded to the customer, which allows the verification of the infrastructure response time. As Apache requests are also based on timestamps, it was possible to relate the load as measured by loadavg with the request response time as measured by Apache. The experiment was repeated 35 times and the average values obtained on each experiment time are reported. As the results showed little variance, 35 experiment repetitions allowed statistically significant analysis of results. In each repetition, the stochastic process was run with a different seed value. The hypothesis of correlation between the two measurements was tested using the Pearson correlation coefficient

A. Evaluation and Discussion

Figure 4 presents the tests results as the average values collected by loadavg and Apache. The values presented a small standard deviation, and thus the deviation is not shown. During the first 15 minutes (900 seconds), the SLA response time of 300 milliseconds was met in its entirety. In the next 15 minutes, the network latency has been increased to enable us to evaluate the effect of resource underutilization by forcing requests to arrive at a lower rate. Similarly, in the last 15 minutes the network latency is reduced to induce a higher request income on the system, causing a longer response time.

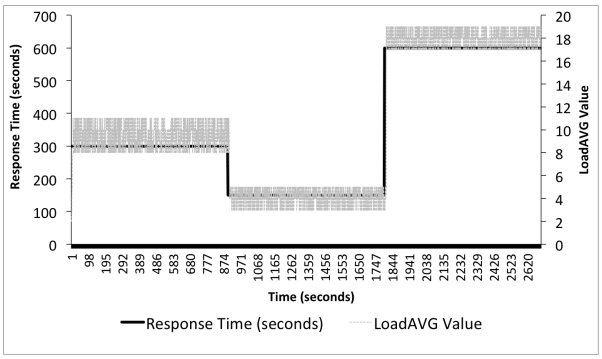

Figure 4. Response Time vs. LoadAVG Evaluation

Figure 5 shows that the load values given by loadavg accompany the application response time behavior. The Pearson correlation coefficient between the behavior of the application response time and load values of the operating system displayed by loadavg was 0.9965. This shows that there is a strong positive correlation between the two behaviors.

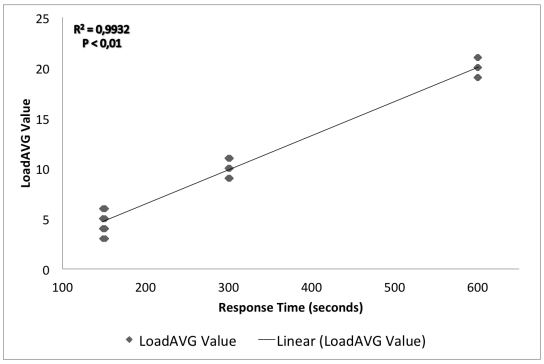

Figure 5. Correlation Plot between Response Time and LoadAVG Values

Based on these results, we can estimate, from the infrastructure, how much the response time is being influenced, and if there is the need for more or less resources so that an SLA is met. The next step towards enabling the estimation of the application response time using loadavg was building a model that captures the relation between these two variables. The modeling process is discussed next.

B. Modeling

Because loadavg values are stored for logging purposes over time, these values can be analyzed to verify that the environment is going towards overutilization or underutilization. This data can be used to automatically scale the application resources to an amount that meets application needs.

The correlation plot from Figure 5 provided some evidence that a linear regression model would be a good fit for our model to estimate the response time. The purpose of multiple regression is to predict or estimate a dependent variable y, in response to the values taken by a set of independent variables X. In this case, we assume our dependent variable Y to be the response time, and we estimate it using the value of loadavg from the last 1 minute, 5 minutes, and 15 minutes (our independent variables).

$$Rt = c_0 + c_1 \cdot avg1 + c_2 \cdot avg5 + c_3 \cdot avg15, \qquad (2)$$

Where Rt represents the estimated cloud response time of applications in the entire nodes. The coefficients c_0, c_1 and c_2 are the weights assigned to each variable, in each loadavg times. $avg1$, $avg5$, and $avg15$ are monitored available values in `/proc/loadavg`, for 1, 5, and 15 minutes, respectively. The c_0, c_1, c_2, and c_3 (4.133, 1.005, 1.478, and 1.597, respectively) values serve to the best fit of the model, and show the curve in the correlation plot.

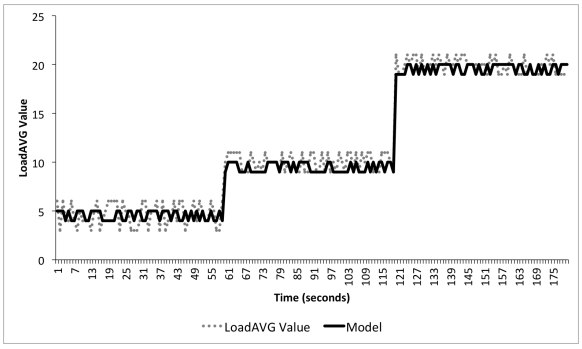

Figure 6. Model Accuracy Test

An important validation aspect of the model is the residue analysis, which shows the model significance and evaluates the contributions of regression variables. In the proposed model, it is possible to assert that all the points follow the behavior of the line, indicating that the errors are normally distributed.

V. Model Evaluation

This section presents an evaluation of our model applied in a real environment, aiming to validate the proposed model.

A. Testbed

For conducting the evaluation of our model, we used HammerDB, because it is a tool that reproduces the behavior of most applications offered by cloud environments as a service. It supports multiple clients accessing a three-tier service. Another advantage of HammerDB is its flexibility for tests configuration. Both the number of users and the number of simultaneous connections can be set and changed on-the-fly during the test. This allows a script to be created to describe the desired behavior. The client-server environment was deployed on the same two physical servers used in preliminary experiments, and using the same network between them.

As the HammerDB allows controlling the client application behavior (increasing or decreasing the number of users and connections), we choose three behaviors that represent routine situations in environments that support cloud services: The first scenario represents an environment in which the arrival rate of requests is low, and gradually increases over the time; The second scenario represents an environment in which initially the arrival rate of requests is low, and gradually increases over the time, and then returns to a low rate; The third test represents an arbitrary behavior fluctuating between moments of slow and high rates of requests.

Each test was performed for 30 minutes, and the results of the real environment tests and the results generated by the model based on loadavg data were compared. All tests were performed 35 times, and the repetitions did not show a significant variation in the results less than 0.03%.

B. Results

Figure 7 depicts the results for the first scenario, where the increase in application demand causes the response time to worsen until the end of the test. We can see that the behavior of the proposed model follows the real trace behavior. Importantly, we can notice that the estimating response time presents variations between real and model results. This difference is due to loadavg update times, which features a refresh rate of 1 minute. Thus, the model responds at least at 1 minute later on the real response time. This update time is inherent to the Linux kernel, and perhaps in future versions, if that time is reduced, the model can track more quickly the application response time.

 ISBN: 978-1-61208-398-8

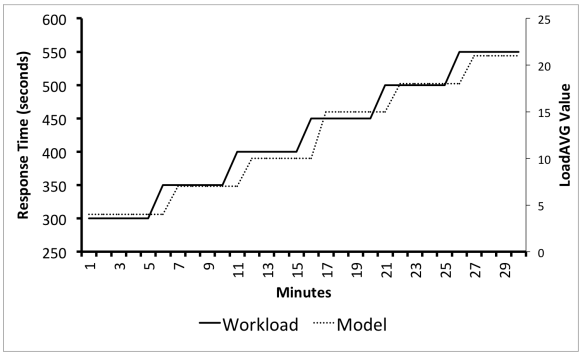

Figure 7. Scenario starting from a situation of an acceptable rate to a gradual increase in the number of requests

Although there is a delay in the model in relation to the real response time, it remains faithful to the real trace, as we can see in Figure 8. Even when the response time worsens and returns to an acceptable level, the model can track all changes and represent, with a very little difference, the real behavior of the response time.

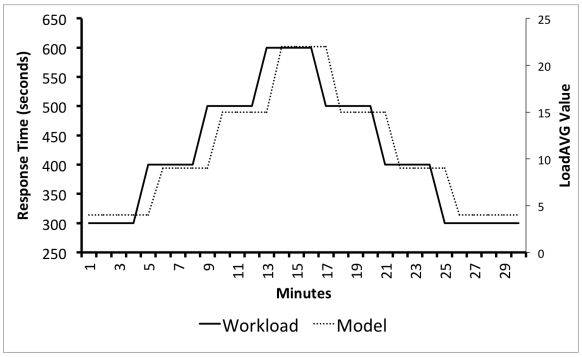

Figure 8. Scenario with fluctuation in the number of requests following a normal distribution

The last test is shown in the Figure 9, where the response time behavior is very diverse, often fluctuating and forcing the model to fit faster way to these changes. Still, the model could represent the fluctuations of the response time fairly well.

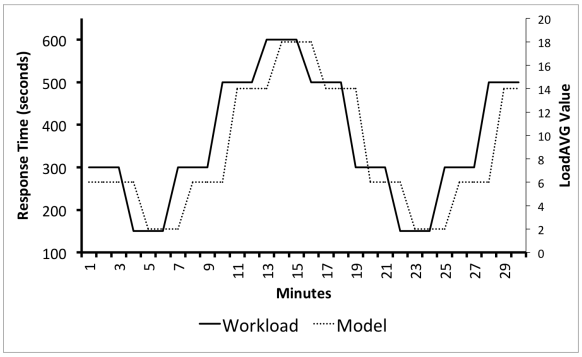

Figure 9. Scenario with arbitrary fluctuation in the number of requests

The model proved to be adjusted during the execution of the three tests against real environments (Figures 7, 8, and 9). To confirm the trend of the model results against the real tests, again we used the Pearson's correlation coefficient. Thus, we can see in Table I, the results for each set of tests. The results showed that on average, the model is accurate to represent the cloud response time up to 94%. The three tests showed P-value < 0.01, what enables us to reject the null hypothesis that both variables are independents.

TABLE I. PEARSON CORRELATIONOF THE TESTS

Test Scenario	*Correlation Coefficient*	*P-Value*
1)	0.944	$p < 0.01$
2)	0.968	$p < 0.01$
3)	0.932	$p < 0.01$

C. Limitations

Models, in most cases, present some limitations to represent real production environments. Our model presents limitations about the accuracy during runtime, because the load values show a delay when compared to the real executions. This is not exactly a limitation of the model, but an operating system counter feature that displays a delay of 1 minute between each update. Unlike High Performance Computing (HPC) jobs that must have a finite time, cloud services mostly keep running throughout the service run time. Thus, cloud services do not have a time of total execution time. So the operating system counter update time (and model) can be neglected, because one minute will not impact strongly on the final result. In addition, the update time is a choice by kernel architects, and this time can be reduced in the future, either on a future release of the Linux kernel. Furthermore, as Linux is an Open Source software, its source code can be easily modified and a new kernel generated that provides more frequent updates in the loadavg.

The second important point to note consists of the architecture used to feed and create the model. The preliminary and final tests were based on one hardware/software architecture. However, the model allows weights adjustment in their c_1, c_2 and c_3 coefficients, which will allow the fitting for each new environment.

The most important feature of the model is to estimate, at one point, what is the application response time. With this information at hand, the infrastructure manager can make decisions in order to maintain an acceptable response time in case of overuse, through the new entities deploy in the resource pool, or in the case of underutilisation, save costs with removal of these. However, although there are limitations set forth above, the main advantage of this model consists of not requiring access to the virtual machine nor interfering directly in the channel between the client and the server, which could cause overhead during a measurement process, slowing the channel.

 ISBN: 978-1-61208-398-8

We believe that our methodology applies, because in cases of flash crowd, the excess requests do not cause excessive utilization of resources in the same proportion: the metrics analyzed in our work displaying the machine's point of view, then the utilization of resources can not go beyond 100%. In this case the excess requests are rejected.

VI. Conclusion and Future Work

Cloud computing environments are rapidly becoming a standard platform to support computational services. As the access to these services is performed via the network, this becomes a decisive factor for the quality of services offered. Among the performance metrics and quality involving quality of users' experience of services, one of the most important is the response time. In addition to impact on the user experience, the response time could indicate that available resources may be insufficient to ensure the proper functioning of the service to users, impacting both users' cost and the total cost of ownership.

High-level metrics, such as response time is often measured between the customer and the cloud provider. However, the infrastructure that supports cloud environments (IaaS) typically monitors low-level metrics, such as CPU, memory, and network usage. Therefore, the translation between high-level metrics to low-level metrics is a very complex challenge in today's cloud environments. Moreover, monitoring high-level metrics are quite difficult, and impact performance and privacy issues of the user. Furthermore, the communication channel between the customer and the cloud listener cannot be monitored by the cloud infrastructure without overhead on the communication channel.

In this paper, we proposed the use of loadavg, a counter of the operating system that stores information on-the-fly on the load of the node, as a parameter to estimate the behavior of the response time. Based on this, we developed a model that analyzes the loadavg values, and estimates what the response time of applications with 94% of accuracy. Such model allows to estimate the time to process the cloud application request within the infrastructure and without any impact on the users or their communication channel. As a future work, we intend to explore other operating system counters such as iostat and netstat, to develop more complete models.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computing Systems, vol. 25, no. 6, Jun. 2009, pp. 599–616.

[2] K. Kritikos, B. Pernici, P. Plebani, C. Cappiello, M. Comuzzi, S. Benrernou, I. Brandic, A. Kertész, M. Parkin, and M. Carro, "A survey on service quality description," ACM Computing Surveys, vol. 46, no. 1, Jul. 2013, pp. 1:1–1:58.

[3] W. Tang, M. S. Kim, and E.-N. Huh, "Analysis of qoe guarantee on hybrid remote display protocol for mobile thin client computing," in Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, ser. ICUIMC '13. New York, NY, USA: ACM, 2013, pp. 118:1–118:8.

[4] V. Gabale, P. Dutta, R. Kokku, and S. Kalyanaraman, "Insite: Qoe-aware video delivery from cloud data centers," in Proceedings of the 2012 IEEE 20th International Workshop on Quality of Service, ser. IWQoS '12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 8:1–8:9.

[5] P. C. Brebner, "Is your cloud elastic enough?: Performance modelling the elasticity of infrastructure as a service (iaas) cloud applications," in Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, ser. ICPE '12. New York, NY, USA: ACM, 2012, pp. 263–266.

[6] V. Emeakaroha, I. Brandic, M. Maurer, and S. Dustdar, "Low level metrics to high level SLAs - LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments," in International Conference on High Performance Computing and Simulation (HPCS), 2010, June 2010, pp. 48–54.

[7] V. C. Emeakaroha, T. C. Ferreto, M. A. S. Netto, I. Brandic, and C. A. F. De Rose, "CASViD: Application level monitoring for SLA violation detection in clouds," in Proceedings of the 2012 IEEE 36th Annual Computer Software and Applications Conference, ser. COMPSAC '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 499–508.

[8] R. Schatz, T. Ho, L. Janowski, and S. Egger, "Datatraffic monitoring and analysis," E. Biersack, C. Callegari, and M. Matijasevic, Eds. Berlin, Heidelberg: Springer-Verlag, 2013, ch. From Packets to People: Quality of Experience As a New Measurement Challenge, pp. 219–263.

[9] T. Rosado and J. Bernardino, "An overview of openstack architecture," in Proceedings of the 18th International Database Engineering & Applications Symposium, ser. IDEAS '14. New York, NY, USA: ACM, 2014, pp. 366–367.

[10] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," SIGOPS Operating Systems Review, vol. 37, no. 5, Oct. 2003, pp. 164–177.

[11] D. Josephsen, Building a Monitoring Infrastructure with Nagios. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2007.

[12] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey," Computer Networks, vol. 57, no. 9, Jun. 2013, pp. 2093–2115.

[13] G. Dobson and A. Sanchez-Macian, "Towards unified qos/sla ontologies," in Proceedings of the IEEE Services Computing Workshops, ser. SCW '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 169–174.

[14] F. Rosenberg, C. Platzer, and S. Dustdar, "Bootstrapping performance and dependability attributes ofweb services," in Proceedings of the IEEE International Conference on Web Services, ser. ICWS '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 205–212.

 ISBN: 978-1-61208-398-8

[15] A. D'Ambrogio and P. Bocciarelli, "A model-driven approach to describe and predict the performance of composite services," in Proceedings of the 6th International Workshop on Software and Performance, ser. WOSP '07. New York, NY, USA: ACM, 2007, pp. 78–89.

[16] M. Comuzzi, C. Kotsokalis, G. Spanoudakis, and R. Yahyapour, "Establishing and monitoring slas in complex service based systems," in Proceedings of the 2009 IEEE International Conference on Web Services, ser. ICWS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 783–790.

 ISBN: 978-1-61208-398-8