Investigation of Software Aging Effects on the OpenStack Cloud Computing Platform

Carlos Melo[1*], Jean Araujo[1, 2], Vandi Alves[1], Paulo Maciel[1]

[1] Federal University of Pernambuco - UFPE, Recife, Pernambuco, Brazil.
[2] Academic Unity of Garanhuns – UAG, Garanhuns, Pernambuco, Brazil.

* Corresponding author. Tel.: +5587999265912; email: casm3@cin.ufpe.br
Manuscript submitted November 11, 2016; accepted January 5, 2017.
doi: 10.17706/jsw.12.2.125-137

Abstract: One of the most important goals for companies that provide cloud computing services is to maintain high availability on large computer systems. Therefore, to accomplish such objective it is necessary to determine the main causes and the main problems associated with it. Many of these problems are related with software failures, for instance software aging. Software aging is a degrading factor in systems, leading to software failures, poor performance and may result in system downtime. This paper investigates the software aging effects on the Openstack cloud computing platform. The environment was tested using a stressful workload, and monitored through Bash scripts that collects information about the utilization of hardware resources, as well as information about processes related to the platform. The results indicate software aging issues in the MySQL process; a growth on the memory consumption was detected, and a prediction analysis was also used to estimate the resources utilization in the future.

Key words: Cloud computing, memory leaking, openstack platform, software aging.

1. Introduction

Cloud computing is a paradigm that consists in the dynamical location of virtualized resources of hardware and software with one goal: provide all kinds of services through the Internet [1]. This model is being increasingly adopted by companies around the globe which seek to reduce the high expenses of maintainability of the infrastructure itself. A key feature of this model is to provide the user with the feeling of owning an unlimited amount of computational resources, through its availability and the on demand elasticity, which in turn allows the service to be expanded in nearly real-time [2].

Another possibility for companies that are willing to offer some kind of service over the Internet is to afford with the costs for the acquisition of its own cloud computing infrastructure. There are currently a number of software that enable such goal, one of which is the OpenStack [3]. There are some issues related to the management of your own cloud computing infrastructure. The issues go from sustaining the high costs of large computer environments, to cooling expenses. There are also issues related to software, called "software failures" that occur very often [4]. These failures are mostly partial, i.e. when the offered service is slow or its scalability is limited.

Software failures are responsible for a degree of degradation on the system during its lifetime. These problems lead to a progressive decrease in the system's performance, thereby leading it to stop functioning properly. This degradation process is called software aging. It is responsible for numerous problems such like disk fragmentation, corruption and loss of data in the storage unit, and memory leak [4]

There are some works that investigate the software aging phenomenon in a cloud computing platform, [5]

and [6] presents the detection of this phenomenon in the Eucalyptus cloud computing platform. In [7] is showed that the software aging is present even in the kernel of Linux operational system. While [8] and [9] used models to estimate where and when the hardware resources are going to be exhausted and define a policy to prevent such issue, this kind of study differs from the one executed in this paper.

Here we present an investigative study of software aging effects in the OpenStack cloud computing platform. We used a similar approach to the one showed in [5], trying to detect and demonstrate the aging effects in this platform, with focus on memory leaking. A study was carried out by performing repeated instantiations and terminations of virtual machines. Furthermore, it was performed the resource utilization prediction to identify possible failure scenarios. In addition, our investigation uses four different types of time series in order to identify which one presents the best fitting with the results.

The remaining of the paper is organized as follows. Section 2 shows the related works that were used as a basis for the development of this study; Section 3 presents cloud computing and software aging fundamental concepts, and the foundations of the usage of time series to forecast a system's behavior. Section 4 provides an overview of the methods used in this study. Section 5 shows the experimental study with the main aspects of the OpenStack platform, and the adopted testbed. Section 6 presents the results obtained from the experiments. Finally, section 7 presents the final remarks and future works.

2. Related Works

Despite not being a recent concept, software aging has not been sufficiently explored by researchers and other professionals from different fields of computing [7]. Many of them prefer to ignore it, especially because it is considered an immutable factor that tends to occur with every software system with time, being present not only in the applications being used, but also in the operating system responsible for its management [7].

Software aging has already been observed and detected in a number of systems massively used around the world [7], such as the Apache Web Server [10], [11], the Java Virtual Machine [12] and the Eucalyptus cloud computing platform [5] and [6]. Since the introduction to the term Software Aging (SA) made by [13] twenty years ago, the interest of the community, both from industry and the academia, only grew up [14].

In [7] the existence of software aging on the Linux operating system's kernel was identified, after a series of long runs that they were able to observe changes in the internal behavior of the kernel of the open source system. Almost 1.3% of the computers around the world have operating systems based in Linux [15], what implies that are millions of users affected by software aging.

The software aging effects in cloud computing environments were addressed in [5] and [6]. Those papers demonstrated the occurrence of faults in an Eucalyptus-based infrastructure due to the memory leaks. Matos *et al.* [16] shows some of the issues of software aging, such as memory leaking and the increase of CPU utilization during consecutive attachments of remote block storage volumes by means of Eucalyptus commands. In [17] and [18], rejuvenation strategies have been proposed for mitigating the downtime caused by the aging effects in that cloud computing framework.

There are some works that use formal modeling to represent the software aging phenomenon, most of them focus in strategies to mitigate its impact. Garg *et al*. [8] analyze and evaluate the benefits of the use of Markov chains and stochastic Petri nets to reduce and prevent in a proactive way the software aging impact in a computer system. In [9] non-homogeneous Markov chains were used to understand the behavior of systems; Therefore to optimize it by selecting the best policy, to reduce the software aging effects in computing environments, as well as the cost to apply this kind of strategy. In [19] and [20], the authors propose rejuvenation scheduling based on live virtual machine(VM) migration. They create SPN models that enable reaching proper rejuvenation schedule for each studied scenario.

3. Fundamental Concepts

This section presents basic concepts related to cloud computing, software aging, and time series, which are fundamental to understand this paper.

3.1. Cloud Computing

Cloud computing is not a new paradigm, since it makes use of long time established technologies, approaches, concepts and practices [21]. The first references to this term date 1996, in one of the offices of Hewlett Packard [22]. But only after nearly a decade this concept became popular, mainly because of the participation of companies like Google and Amazon, which started to make use of the term "cloud computing", that led to a great deal of information regarding such term [22].

Foster *et al*. [23] defines cloud computing as a powerful paradigm from the distributed computing; characterized by the abstraction of hardware and software resources, i.e. virtualized, dynamically allocated with great manageability. Providing infrastructure, platform and service on demand to the customers over the Internet.

Cloud computing can provide an integration between technological models, and different kinds of services provided. Including hardware infrastructure (Infrastructure as a Service), development platforms (Platform as a Service), applications on-demand (Software as a Service) [21], [24].

As for the cloud types, that refer to the nature of access and control of the cloud computing environments, we may cite the public ones, accessed by anyone with a network connection, they are open to public. Private ones, owned, managed, accessed and controlled by an organization; and hybrid ones, a combination between public and private clouds, where a private cloud can be expanded to use the resources provided by the public cloud [21].

3.2. Software Aging

The term software aging was evaluated just as an occasional phenomenon experienced only by badly designed systems. However, with the passing of the years, more and more studies about system failures were reported, and the main cause of those failures was the software aging. Therefore, it became a non-negligible problem on long-running systems [7].

Like people, animals and even the world itself, computer programs get old [25]. The age cannot be stopped by any usual means, but with some efforts we can mitigate its effects, reducing the damage and increasing the lifetime of the systems. By understanding its causes, and predicting where it is going to start, we can be prepared for the day when the software may no longer be available [25].

In [14] software aging is defined as a phenomenon that mainly consists in increasing degradation of the internal state of a software during its operational life. Due to its cumulative property, the software aging occurs with more intensity at systems that have been running for a long period of time. Such systems also have a higher probability of entering in a critical condition, for having their integrity damaged by the vast amount of software failures [4].

In this paper we investigate one of the most common problems associated with software aging, which is the memory leak [4]; that can lead the system to fail due to resource exhaustion. Programs in execution are running in the random access memory (RAM) of the computer, which is a fast access memory. If the system runs out of such resource, then the programs will try to use the swap memory, which is the slowest access memory, if the swap resource end, then the operating system can reboot itself and discontinue providing the service [4].

3.3. Time Series

A time series is a sequence of observations ordered in time [26]. Mostly of these observations are

collected at equally spaced, discrete time intervals [27]. Due to such fact, a time series enables one to build models which explain the behavior of the observed variable [26], [28]. In this case study, our time series models explain the behavior of our cloud computing platform, as well as its associated process.

The time series' values can be represented by a t that is described by a stochastic process; In this study \$t\$ represents time. A random variable $X(t)$, for each $t \in T$, where T is an arbitrary set, and its associated probability distribution. The applications of a time series analysis are mainly: description, explanation, process control and prediction, the last one was applied in our study.

This work adopts four models, namely: the linear model, the exponential growth model, the quadratic model, and the S-Curve Model. Based on $Y_t = E[X(t)]$, we can describe such models as follows:

Linear Trend Model (LTM): considered the default model used for trend analysis; Its equation is given by $Y_t = \beta_0 + \beta_1 \cdot t + e_t$ where β_0 is known as the y-intercept, β_1 represents the average of the exchange between one time period and the next one, and e_t is the error of fit between the model and the real series [29].

Growth Curve Model (GCM): this model represents the trend growth in exponential form; Its equation is given by $Y_t = \beta_0 + \beta_1^t + e_t$ [29].

Quadratic Trend Model (QTM): it takes in account a smooth curvature in the data; It is represented $Y_t = \beta_0 + \beta_1 \cdot t + \beta_2 \cdot t^2 + e_t$ where the coefficients have similar meanings of the Linear Trend Model [29].

S-Curve Trend Model (SCTM): this model is usually used in time series that follows the shape of the curve S. It is represented by $Y_t = 10^a/(\beta_0 + \beta_1 \cdot \beta_2^t)$.

Error measures were adopted to choose the model that best fit our observed data [30]. We adopted MAPE, MAD and MSD as our error measures:

MAPE (Mean Absolute Percentage Error) represents the precision of the estimated values of the time series expressed in percentage [30]. Its estimator is given by equation 1:

$$MAPE = \frac{\sum_{t=1}^{n} |(Y_t - \hat{Y}_t)/Y_t|}{n} \cdot 100, \tag{1}$$

where Y_t is the actual value observed at time t ($Y_t \neq 0$), $\hat{Y}_t$ is the estimated value and n is the number of observations.

MAD (Mean Absolute Deviation) represents the accuracy of the estimated values of the time series. It is expressed in the same unit of data [30]. It is an indicator of the error size and is represented by equation 2:

$$MAD = \frac{\sum_{t=1}^{n} |(Y_t - \hat{Y}_t)|}{n}, \tag{2}$$

where Y_t, t, $\hat{Y}_t$ and n have the same meanings index MAPE.

MSD (Mean Squared Deviation) is a more sensitive measure than the MAD index, due to such fact it is especially used in large forecasts [30]. Its expression is given by equation 3:

$$MSD = \frac{\sum_{t=1}^{n} |(Y_t - \hat{Y}_t)|^2}{n}, \tag{3}$$

where Y_t, t, $\hat{Y}_t$ and n have the same meanings of the previous indexes.

4. Methodology

One of the first tasks in an experimental study is to determine what are our goals and how we may reach them.

To evaluate a behavior of any computing system, that includes the cloud computing ones, it is necessary to define a list with the hardware and software resources; The considered resources should be important to the experiment, and help us complete our tasks [31].

After listing our important resources, we can determine a way to obtain the information that we need [31]. To achieve that, we implemented an application using Bash scripting language. We decided to use a scripting language instead of an ordinary monitoring tool, because it provides a more flexible approach to monitor the system behavior. Fig. 1 shows an organization chart that summarizes the strategy used in this paper.

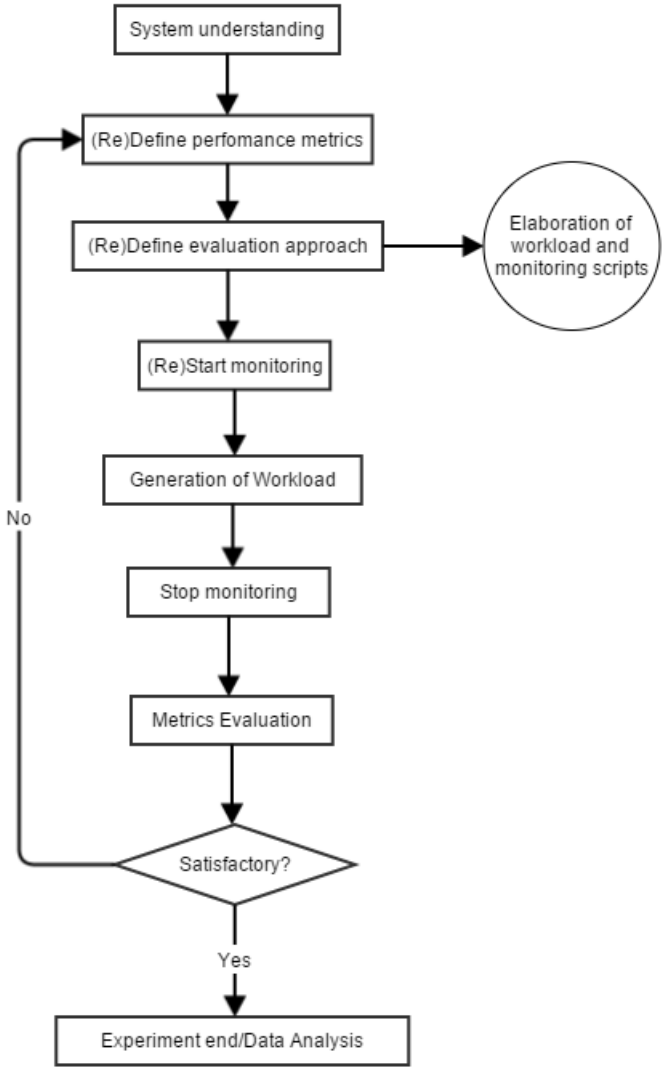


Fig. 1. Organization chart.

System understanding: A deep assessment of OpenStack cloud computing platform, that includes the behavior of the platform, the operational modes, and the processes associated with OpenStack;

(Re)Define performance metrics: Here are defined the performance metrics associated to the platform evaluation, such like hardware resources consumption, response time, and service providing;

(Re)Define Evaluation Approach: Establishment of the monitoring and the workload generation strategies,

seeking to accelerate the results generation;

Start Monitoring: The resources monitoring scripts are started;

Generation of Workload: The workload begins to be applied on the platform, trying to provide some stress level to the system, in order to understand the platform's behavior under stress circumstances;

Stop monitoring: The monitoring scripts are halted;

Metrics Evaluation: Based on the results we can define if our metrics are satisfactory to our study, and then carry out a deep analysis of the results; Otherwise we have to reestablish them;

Experiment end/Data Analysis: The end of our experiment and analysis of the results, generation of graphics, charts and tables with the main results;

5. Experimental Study

The computer used for the experimental study is composed by one AMD FX8350 octacore, 4.0 GHz per core, 8 GB RAM DDR3 1866 Mhz, running the Fedora Linux 20, Heisenbug, with the version 9, IceHouse, of the Openstack cloud computing platform. The operating system running within the virtual machines is the Slitaz Linux 64-bit. The cloud environment under test is fully based on the Openstack framework and the QEMU/KVM hypervisor. Next, we describe the Openstack platform and the main components of our testbed, as well as the workload adopted.

Fig. 2 shows the components of our experimental environment. The Openstack All-in-One configuration was chosen, with all processes and services running in the same machine. OpenStack is an open and extensible platform that emerged from the consortium initially formed between two large U.S. organizations: Rackspace and NASA [32]. The OpenStack is able to provide to both, large and small organizations, the ability to build their own cloud computing infrastructure; Enabling public or private clouds, reducing the risks of restrictions associated with the exclusive platforms [32].

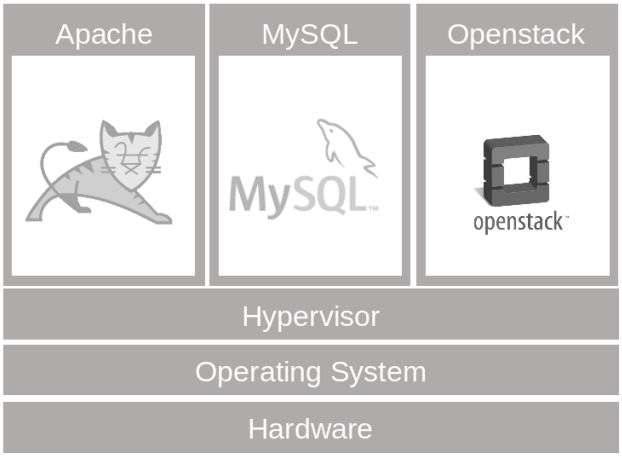


Fig. 2. Testbed environment.

Openstack is arranged in two main strands: *Computing* and *Storage*. *Computing* OpenStack, or *Nova*, provides management of large networks consisting of virtual machines, which enables the creation of a cloud computing platform redundant and scalable; In addition to providing the software, APIs and control panels required for managing the environment [3]. In OpenStack *Storage* or *Cinder*, we have a process of servers' clustering, allowing large-scale storage of data [3]. The Openstack also provides a strand called *Neutron*, it implements the "network as a service" model [3].

5.1. Workload

It is possible to accelerate the software aging process by adopting a stressful workload. Therefore what could take several months or even years, will take only a few days or depending on the issues' severity, it may take hours. Some scripts are used in order to start, reboot and terminate the VMs in a short time period. The behavior of these scripts is shown below, while the Fig. 3 shows the workload characterization adopted in this paper. Every five minutes the script checks if more than thirty minutes have passed from the beginning of the last initialization. If it is true, all VMs are terminated, otherwise all VMs are rebooted. The workload was set just to speed up the effects of software aging, so that the time values that we have chosen do not attempt to illustrate real cases.

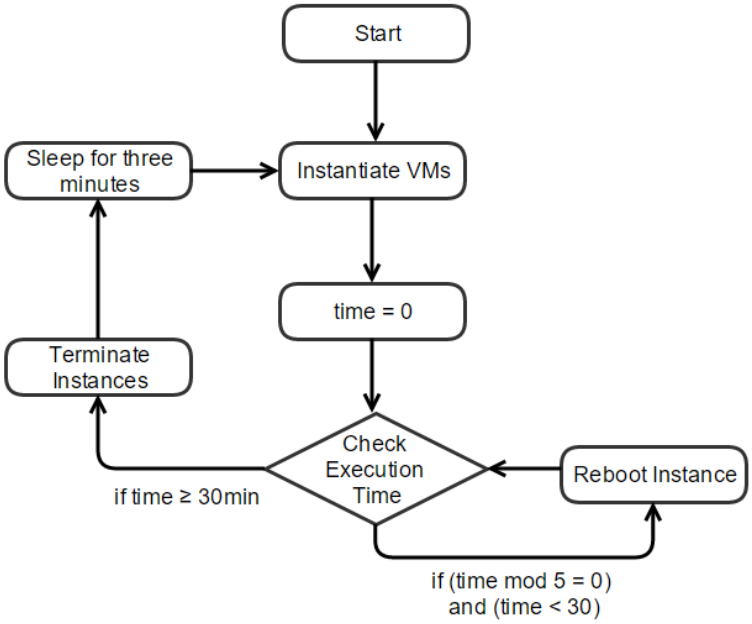


Fig. 3. Workload characterization.

Start: Performs allocation of hardware resources;

Instantiate VMs: VMs start running;

Time: Set the initial time;

Check Execution Time: Verify every five minutes if the actual time minus the initial time is equal to thirty minutes, if not then the VMs are rebooted, otherwise the instances are terminated;

Sleep for three minutes: A delay of three minutes is applied to guarantee that the virtual machines will be terminated, due to the natural delay of the shutting down operation;

5.2. Monitoring Strategy

In order to analyze possible aging effects in the Openstack cloud computing platform, some bash scripts were implemented and applied to our environment. We created monitoring scripts using Linux utilities, such as: *date, mpstat, free, ps* and *du* [33]. We use the scripts to monitor the utilization of resources such as CPU, disk and memory. Those resources were monitored for the entire system, as well as specifically for the services *Nova*, *Cinder* and *MySQL*. The data was collected every 60 seconds throughout the experiment.

A part of the monitoring scripts had their focus on the general consumption of the hardware resources. That includes the consumption realized by all the process in execution, threads and the operating system itself. While another part of the scripts aimed the resources' consumption of the processes related to the Openstack and the service provision; That way we could identify a specific process that might be experiencing a software aging issue. The monitoring scripts collected the data every sixty seconds, enough

time to observe any important changes in the system behavior, such as CPU and memory consumption, after that we could determine if the resources were being consumed excessively.

6. Result Analysis

We analyzed the consumption of hardware and software resources in a scenario which basic cloud operations are continuously performed, during a period of seven days. Throughout this time the general machine resources such as CPU, RAM, hard disk and swap memory were monitored. Whereas is very difficult to identify the existence of software aging when only the general resources of the system are monitored, some other important processes were also monitored: *Nova Compute, Cinder, Neutron, Libvirt, MySQL* and *Apache Server*.

The total used memory by the *MySQL* process is the most representative result found in this experimental study. CPU usage metrics do not show any perceptible aging behavior, so they are not included in this paper. Libvirt and Apache processes also did not show significant results.

To accomplish the forecasts we used time series. For this, we applied the concept of trend analysis. It is worth mentioning that, besides the total swap memory consumed by the system, only the processes that had a better adaptation to the time series are presented, such as the MySQL.

Fig. 4 depicts the trend analysis of the total of memory used by the MySQL, it shows a visible increase and great accuracy. The index for this analysis can be seen in Table 1, the best model for this analysis was the QTM. In Table 2 we see the comparison between actual values and those obtained from the time series calculations, the error ranges from 0.03% in the beginning of the experiment and finishes at 0.68% in the end of the experiment.

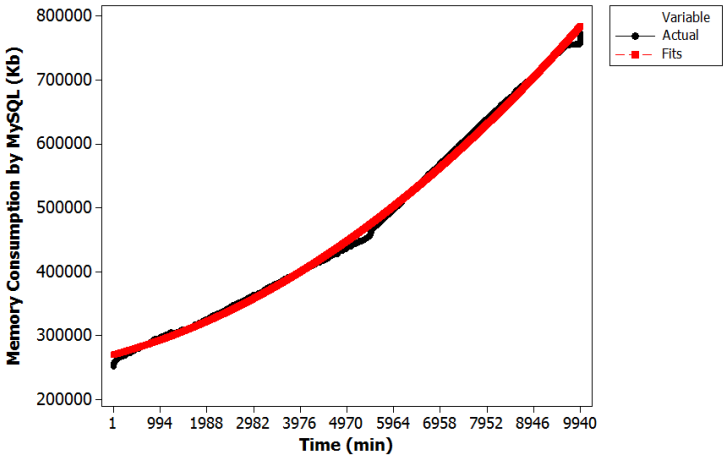


Fig. 4. Trend analysis of memory consumption by MySQL.

Table 1. Accuracy Rate for MySQL Memory Consumption

Model	$\hat{Y}_t$	MAPE	MAD	MSD
LTM	$217489 + 51.7t$	5	21229	603111801
QTM	$270307 + 19.8687t - 0.003204^2$	1	5302	45482143
GCM	$260608(1.00011^t)$	1	6127	62845666
SCTM	$(10^7)/(-13.4685 + 50.7424(0.999931^t))$	1	7481	153500827

Table 2. Comparison between MySQL Actual Consumption and Trend Analysis

Time(Min)	Actual(Kb)	Fits(Kb)	Error(%)
994	294676	294786	0.03

2982	363160	363476	0.08
4970	436760	449339	2.88
6958	569192	568226	0.17
8946	706392	701643	0.68

MySQL process has a growth of approximately 500 MB in 7 days. Before this point, the OpenStack could not instantiate VMs anymore. The workload failed after the first 3 days of experiment, but after this milestone the consumption by this process continued to grow, probably because of a system's failure.

Fig. 5. presents the Nova-API memory consumption, one of the many resources of the Openstack; There we could see a decrease of the total memory consumption over time. Some of the Openstack's processes are changing the utilization from a faster memory to a slower one. That affects the performance of the provided service.

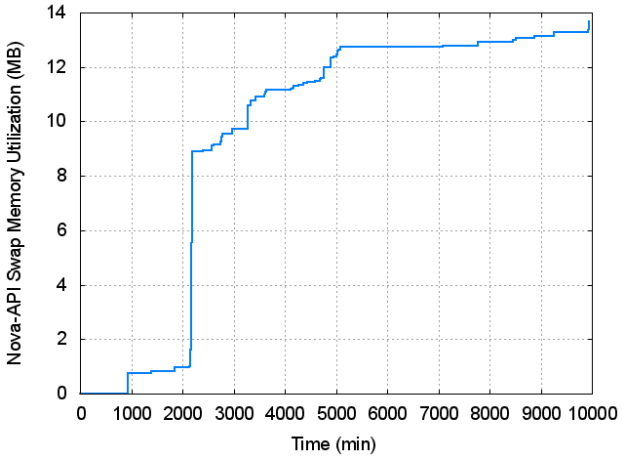


Fig. 5. Memory consumption by Nova-API process.

Fig. 6. shows the trend analysis of swap memory used by the entire system. The one that fits the obtained results better was the quadratic model, QTM. Here we do not have the values from an exponential trend analysis, because some of the data equals to 0 (zero), since the swap memory start at this point. The index for this analysis can be seen in Table 3. In Table 4 we see the comparison between actual values and those obtained from the fitting model, the average error starts at 88% but it stabilizes at 8% after 8946 minutes, both lines tends to touch each other somewhere in the future, when the resources becomes scarce.

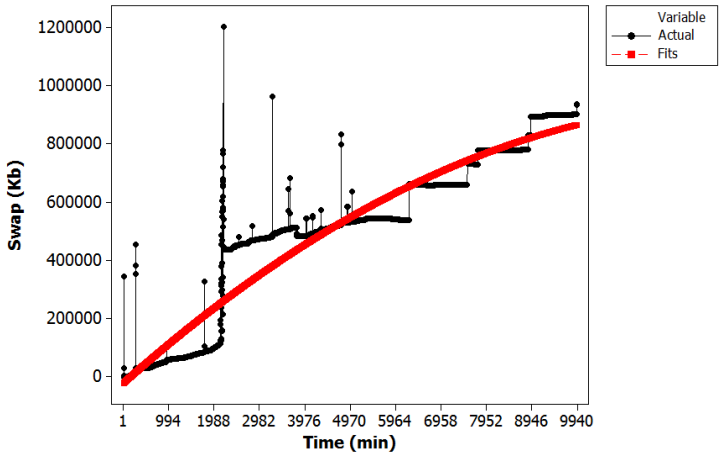


Fig. 6. Trend analysis of swap memory consumption.

Table 3. Accuracy Rate for Swap Memory Consumption

Model	$\widehat{Y}_t$	MAPE	MAD	MSD
LTM	$63176 + 89.2t$	264	67486	7450925599
QTM	$-20561 + 139.72t - (0.005083t^2)$	66	59540	6049390977
SCTM	$(10^7)/(12.9674 - 5600.93(0.998427^t))$	4.13150E+01	1.19432E+05	2.50808E+10

Table 4. Comparison between Swap Actual Consumption and Trend Analysis

Time(Min)	Actual(Kb)	Fits(Kb)	Error(%)
994	60216	113219	88.00
2982	474020	352473	25.65
4970	532536	548488	2.99
6958	659644	712795	8.05
8946	894136	819451	8.36

7. Final Remarks

This study investigated the software aging effects on the Openstack cloud computing platform. In order to accelerate the degradation process of the environment, the workload performed continuous reboot and terminate operations on the virtual machines. About three days later, there was a system crash. The resources were monitored every 1 minute. It was identified that the total memory consumed by *MySQL* process reached a state of exponential growth; *MySQL* is one of the most important component for the maintenance of the OpenStack's availability. *Cinder*, *Nova Compute* and other processes associated to Openstack started to use the swap memory due to the lack of RAM caused by the *MySQL* memory leak. The use of time series proved to be an efficient strategy to estimate the resources utilization through time. In this study we have used a compact cloud computing infrastructure. This shows the Openstack's power to deliver a cloud environment with low expenses to service providers. Our study differs from the ones realized in [5] and [6] by focusing on a platform on the rise, unlike the already renowned Eucalyptus.

It is important to note that this opens the possibility for the realization of many future works, using a larger infrastructure, such as a data center itself. As another future work, we intend to implement a software rejuvenation strategy and analyze its impact on the system's availability. We can also compare the performance loss and the level of degradation between Openstack and other cloud computing platforms, such as Eucalyptus; Aiming which platform has a better availability, thus providing a more lasting infrastructure for the companies which provide the services.

References

[1] Vaqueiro, L. M., Rodero-Merino, L., Caceres, J., & Lindner, M. (2009). A break in the clouds: Towards a clouddefinition. *Computer Communication Review*, *39*, 50–55.

[2] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., & Zaharia, M. (2009). Above the clouds: A berkeley view of cloud computing. *UC Berkeley ReliableAdaptive Distributed Systems Laboratory*.

[3] OpenStack. (2013). OpenStack cloud software. Retrieved from: http://www.openstack.org/software/

[4] Matias, R., Beicker, I., Leitao, B., & Maciel, P. (2010). Measuring software aging effects through os kernel instru-mentation. *Proceedings of the 21th IEEE Int. Symp. on Software Reliability Engineering Workshop on Software Aging and Rejuvenation*.

[5] Araujo, J., Junior, R. M., Maciel, P., & Matias, R. (2011). Software aging issues on the eucalyptus cloud computing infrastructure. *Proceedings of the IEEE Int. Conf. on Systems, Man, and Cybernetics*.

[6] Araujo, J., Junior, R. M., Maciel, P., Matias, R., & Beicker, I. (2011). Experimental evaluation of softwareaging effects on the eucalyptus cloud computing infrastructure. *Proceedings of the ACM/IFIP/USENIX International Middleware Conference (Middleware'11)*.

[7] Cotroneo, D., Natella, R., Pietrantuono, R., & Russo, S. (2010). Software aging analysis of the linux operating system. *Proceedings of the 2012 IEEE 23rd International Symposium on Software Reliability Engineering*.

[8] Garg, S., P. A., T. M., & Trivedi, K. S. (1995). Analysis of software rejuvenation using markov regenerative stochastic petri net. *Proceedings of the Sixth International Symposium on Software Reliability Engineering*.

[9] Koutras, V. P., Platis, A. N., & Gravvanis, G. A. (2007). On the optimization of free resources using non-homogeneous markov chain software rejuvenation model. *Reliability Engineering and System Safety*.

[10] Grottke, M., Vaidyanathan, K., & Trivedi, K. S. (2006). Analysis of software aging in a web server. *IEEE Transactions on Reliability*.

[11] Matias, R., & Filho, P. J. F. (2006). An experimental study on software aging and rejuvenation in webservers. *Proceedings of the 30th Annual Int. Computer Software and Applications Conference.*

[12] Cotroneo, S. O. D., & Russo, S. (2007). Characterizing aging phenomena of the java virtual machine. *Proceedings of the 26th IEEE Symp. on Reliable Distributed Systems (SRDS)*.

[13] Huang, Y., Kintala, C., Kolettis, N., & Fulton, N. D. (1995). Software rejuvenation: Analysis, module and appli-cations. *Proceedings of the 25th Symp. on Fault Tolerant Computing*

[14] Grottke, M., Matias, R., & Trivedi, K. (2008). The fundamentals of software aging. *Proceedings of the 1st Int. Workshop on Software Aging and Rejuvenation*

[15] Stat counter global stat. Retrieved from: Stat Counter. http://gs.statcounter.com/

[16] Junior, R. M., Araujo, J., Alves, V., & Maciel, P. (2012). Experimental evaluation of software aging effects in the eucalyptus elastic block storage. *Proceedings of the IEEE Int. Conf. on Systems, Man, and Cybernetics* .

[17] Araujo, J., Junior, R. M., Maciel, P., Vieira, F., Matias, R., & Trivedi, K. S. (2011). Software rejuvenation in eucalyptus cloud computing infrastructure: a method based on time series forecasting and multiple thresh-olds. *Proceedings of the 3rd International Workshop on Software Aging and Rejuvenation (WoSAR'11) in Conjuction with the 22nd Annual International Symposium on Software Reliability Engineering*.

[18] Junior, R. M., Araujo, J., Maciel, P., Vieira, F., Matias, R., & Trivedi, K. S. (2012). Software rejuvenation in eucalyptus cloud computing infrastructure: A hybrid method based on multiple thresholds and time series prediction. *International Transactions on Systems Science and Applications*.

[19] Machida, F., Kim, D. S., & Trivedi, K. S. (2010). Modeling and analysis of software rejuvenation in a server virtualized system. *Proceedings of the 2010 IEEE Second International Workshop on Software Aging and Rejuvenation*.

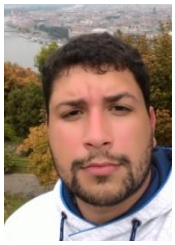
[20] Melo, M., Maciel, P., Araujo, J., Matos, R., & Araujo, C. (2013). Availability study on cloud computing environ-ments: Live migration as a rejuvenation mechanism. *Proceedings of the 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks.*

[21] Microsystems, S., *Introduction to Cloud Computing Architecture.* 1st ed. Sun Microsystems, Inc., Jun.2009.

[22] Regalado, A. (2011). Who coined cloud computing? Retrieved from: http://www.technologyreview.com/news/425970/

[23] Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008). Cloud computing and grid computing 360-degree compared.

Grid Computing Environments Workshop.
[24] Sa, T. T., Soares, J. M., & Gomes, D. G. (2011). Cloudreports: Uma ferramenta grafica para a simulacao de ambientes computacionais em nuvem baseada no framework cloudsim.
[25] Parnas, D. L. (1994). Software aging. *Proceedings of the 16th Int. Conf. on Software Engineering*.
[26] Bloomfield, P. (20000. *Fourier Analysis of Time Series: An Introduction*. Wiley Series in Probability and Statistics.
[27] R. V. (2007). *Time series Analysis*.
[28] Kedem, B., & Fokianos, K. (2002). *Regression Models for Time Series Analysis*. John Wiley & Sons, Inc., Publication., August 2002.
[29] Montgomery, D. C., Jennings, C. L., & Kulahci, M. (2008). *Introduction to Time Series Analysis and Forecasting*. Wiley series in probability and statistics, 2008.
[30] Schwarz, G. (1978). *Estimating the Dimension of a Model*. Annals of Statistics.
[31] Jain, R. (1991). The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling..
[32] Rackspace. (2013). Rackspace the open cloud company. Retrieved from http://www.rackspace.com/pt/?CMP=GEOUS
[33] Blum, R. (2008). *Linux Command Line and Shell Scripting Bible*. Wiley Publishing, Inc, May 2008.



Carlos Melo received his B.S. degree in computer science from the Academic Unity of Garanhuns and the M.Sc. degree in computer science from Federal University of Pernambuco in 2016, and is currently a PhD. student in computer science at the same university.

He author of scientific papers in international journals and conferences on cloud computing, synchronization systems and analytical modeling, also systems dependability and performance evaluation.

Carlos Melo is currently a member of IEEE.

Jean Araujo received his B.S. degree in information systems from the Seven of September Faculty, Brazil, in 2008. He specializes in network security computer by Gama Filho University. He earned his M.Sc. in computer science from the Informatics Center of the Federal University of Pernambuco in 2012, and is currently a PhD. student in computer science at the same university.

He is the author of scientific papers in international journals and conferences on software aging and rejuvenation, cloud computing and analytical modeling. Among the various areas, stand out performance and dependability evaluation, computational modeling and distributed systems.

Prof. Araujo is currently assistant professor by the Federal Rural University of Pernambuco, and a member of IEEE and Brazilian Computer Society.

Vandi Alves de Lira Neto recived his bachelor in computer engineering at Informatics Center, Federal University of Pernambuco (Recife, Brazil).

He currently works as an innovation analist at Neurotech.

Paulo Maciel received the degree in electronic engineering in 1987 and the M.Sc. and Ph.D. degrees in electronic engineering and computer science from the Federal University of Pernambuco, Recife, Brazil, respectively.

He was a faculty member with the Department of Electrical Engineering, Pernambuco University, Recife, Brazil, from 1989 to 2003. Since 2001, he has been a member of the Informatics Center, Federal University of Pernambuco, where he is currently an Associate Professor. In 2011, during his sabbatical from the Federal University of Pernambuco, he stayed with the Department of Electrical and Computer Engineering, Edmund T. Pratt School of Engineering,

Duke University, Durham, NC, USA, as a visiting professor. His current research interests include performance and dependability evaluation, Petri nets and formal models, encompassing manufacturing, embedded, computational, and communication systems as well as power consumption analysis.

Dr. Maciel is a research member of the Brazilian Research Council.