OpenStackFT: Fault Tolerance in Open Source Cloud Computing Environment

H. P. Martins[1], **R. Spolon**[1], **N. G. Bachiega**[1], **R. S. Lobato, A. Manacero**[2], **M. A. Cavenaghi**[3]
[1]Departamento de Ciências da Computação, Universidade Estadual Paulista "Júlio de Mesquita Filho" Bauru, SP, Brasil
[2]Departamento de Ciências da Computação e Estatística, Universidade Estadual Paulista "Júlio de Mesquita Filho", São José do Rio Preto, SP, Brasil
[3]Humber Institute of Technology & Advanced Learning, The Business School, Toronto, ON

Abstract - *Cloud Computing is a set of features and services offered over the internet, delivered from data centers located around the world. As the Cloud Computing grows fast, the concern with the need of services offered increases, and the major challenge is to implement a fault-tolerant environment. The main issues of fault tolerance in Cloud Computing are fault detection and recovery. In order to combat such problems, many techniques are projected. Paid managers offer this kind of support, but the open source managers do not provide evidence to tolerate failures and leave users vulnerable to failures of the technology environment. This study presents the OpenStackFT, a fault-tolerant mechanism developed for the OpenStack manager. A redundancy mechanism was created in virtual machines instantiated in cloud nodes. If a node presents a transient or intermittent failure, the virtual machine will be stored on a backup node, waiting for the node to return from a failure. Experimental results show that the mechanism developed is viable and efficient because, right after a node has recovered from a failure, the virtual machine is not lost, thus becoming active again to the user.*

Keywords: *cloud computing; openstack; fault tolerance.*

1 Introduction

As there is a constant increase of computational use, problems like energetic demand and space in the data center are occurring all over the world. Many solutions are being projected to solve this kind of situation, among them is the Cloud Computing, term used initially by IBM in its white paper about technology in 2007 [1].

Cloud can be defined as a network environment based on the sharing of computing resources. Clouds are based on the internet and they try to make the complexity transparent to customers. Cloud Computing refers to applications (from hardware and software) delivered as services over the internet from data centers. Companies that provide clouds use virtualization technologies, combined with their abilities to provide computing resources through the network infrastructure [2].

In cloud environments, the concern is whether there will be high availability in the services offered by cloud managers. Many companies are migrating their services to cloud and choosing to implement their own cloud, using some open source managers such as the OpenStack.

Given the aforementioned context, there is a concern about having a fault-tolerant environment, if there is an environment failure, it means the environment is not providing the services properly to what it was designed for. If a distributed system is designed with a set of servers that communicate with each other and with customers, the inadequate supply of services means that the servers are not doing what they should. However, the fault is not always on the server that presents it, since if the server depends on other servers to provide its services, the error may have to be looked in another place [3]. This study shows the mechanism developed for the OpenStack manager.

2 Related Research

OpenStack has information about its high availability and fault tolerance on its support website, but no information on how to implement the solutions is displayed or demonstrated. The company Rackspace uses OpenStack[1] as a solution. It informs it has implemented fault tolerance and high availability, but it does not display information about how they are implemented.

Another highlighted open source is the Apache CloudStack[2], a platform that gathers computing resources for the construction of infrastructure as a service. CloudStack has some high availability characteristics such as the Management Server, which can make its implementation by itself in many nodes, where servers are balanced between data centers. The database MySQL can be set to use replication, preventing a failure situation in case of a data loss.

[1] https://www.openstack.org/
[2] http://cloudstack.apache.org/index.html

Paid clouds servers such as VMware, Amazon and Citrix have implemented fault tolerance solutions in their distributions. This research was designed from these paid managers, which are presented in this section.

The vSphere product of the VMware provides availability for a company entire virtual environment, minimizing unplanned downtime by restarting the virtual machine, providing a level of high availability [4]. A number of 5 products of high availability resources are presented such as: High Availability, Data Protection, App HÁ, Fault Tolerance and the Replication.

The AWS - Amazon Web Services (Amazon Web Services) of Amazon is one of the tools and resources that allows the creation of fault-tolerant systems that are reliable and demand little human intervention. The Amazon Services: Elastic Compute Cloud (EC2) and the Amazon Elastic Block Store (EBS) provide resources such as snapshots and availability zones, which fault-tolerant systems are highly available [5].

The Citrix XenServer is a complete virtual infrastructure solution, with a management interface, live migration resources, and tools to convert workloads from a physical environment to a virtual one. It is possible to create and manage virtual machines that can be executed from a management interface, it allows the active virtual machines to be transferred to a new physical host without the interruption of its applications, generating inactivity [6].

3 Fault tolerance Mechanism

As a distributed system environment, Cloud Computing is susceptible to faults, and fault tolerance techniques must be used to improve environment availability. A vulnerability considered critical to cloud functioning was chosen for this research. The vulnerability chosen was the hardware fault-tolerance problem in the nodes. Thus, the fault-tolerance mechanism labeled as OpenStackFT was created.

For the implementation of OpenStackFT, a node was used with the Linux OpenFiler operating system, NAS (Network Attached Storage) and SAN (Storage Area Network) for open source storage appliance, which provides data storage device access via network.

The OpenFiler3 was chosen because it has support for volume-based partitioning (iSCSI - Internet Small Computer System Interface) and management of settings via the Web. Another reason to choose the iSCSI was the test performed with other tools that did not work in OpenStack, these tools were the DRBD (Distributed Replicated Block Device), NFS (Network File System) and RSync.

The iSCSI is used to connect storage devices through a network via TCP/IP. It can be used through a local area network (LAN), a wide area network (WAN) or through the Internet. The iSCSI devices include disks, tapes, CD-ROMs and other storage devices in another computer in the network which can be connected. Sometimes, these storage devices are part of a storage area network (SAN). In the relation between the computer and the storage device, the computer is labeled as the Initiator, because it initiates the connection with the device, which is labeled as Target..

Figure 1 shows the OpenStackFT environment. In this environment, it is possible to observe the OpenFiler, responsible for the storage of the instances initiated by the OpenStack.

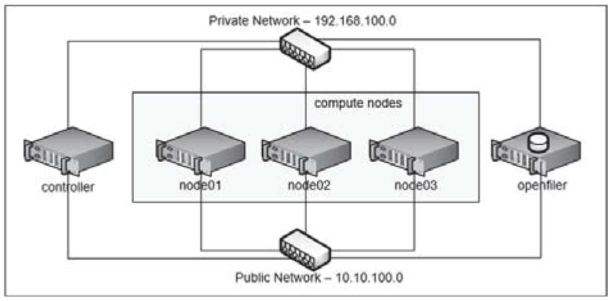


Figure 1. OpenStackTF environment.

Figure 2 shows the flow of a virtual machine when instanced by Horizon. When the virtual machine is instanced in an available node, the iSCSI protocol that is configured and active in the node replicates in the OpenFiler, creating a redundancy of the virtual machine in a directory available to the node. In the example of Figure 2, the virtual machine instanced by *node01* will be stored in the OpenFiler /dev/sdb directory. This virtual machine that was sent to OpenFiler is constantly updated by iSCSI, until a fault occurs in the node. If a fault occurs in the node, the virtual machine turns to stand by in the OpenFiler until the node recovers itself from the fault.

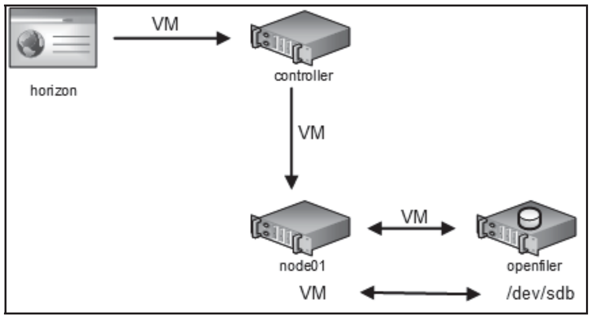


Figure 2. Virtual Machine (VM) flow in the environment.

It is emphasized that the OpenStackFT test environment has limitations, but the same environment can be reproduced in more robust hardware environments.

4 Tests and Results

The tests were carried out based on the requirements established by Tanenbaum and Steen [3], who described a fault-tolerant environment, implementing availability, reliability, safety and maintainability.

3 https://www.openfiler.com/

The tests were intensified in the nodes, since, as mentioned above, a user creates a virtual machine which will be allocated on a node, and if this node has problems, the virtual machine that is connected is lost and must be discarded by the cloud administrator. This failure makes the user lose the virtual machine, having to create a new one.

To simulate a node failure, three types of tests were carried out: unplugging the node's network cable, restarting the node, and disconnecting the node by removing it from the outlet. These failure tests are classified as transient or intermittent. For the tests, the *ping*[4] tool was used to test if the node did not respond and if it showed the expected failure. In addition to using the ping tool on two computers, the researcher personally made sure that the machine was off the network or powered off.

The failure simulation tests were timed to verify how long it takes for the virtual machine to be available for the user again. To accomplish this control of time, the controller operating system clock was used. The time control was performed as follows:

- Test 1: In the event of a network failure, the network cable was removed and a confirmation by the ping that the machine did not respond was waited. Then the network cable was plugged in and the timer was initiated. When the status of the virtual machine became "Active", the timer was stopped.
- Test 2: In the event the machine restarts, the command "shutdown -r now" was used, at the time the command was executed, the timer was initiated, and this was ceased at the time the status of the virtual machine became "Active".
- Test 3: In the event the machine abruptly turned off, the power cable was removed and plugged in soon after. At the time the machine was turned off, the timer was initiated, and it was ceased when the status of the virtual machine became "Active".

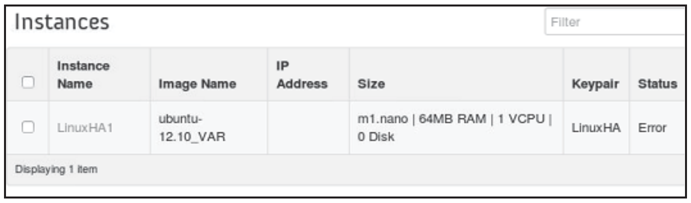
To ensure fairness and efficiency in the data collection, a number of 30 tests were performed in the environment; the results are shown in Table 1. The average time observed was 3 seconds for Test 1, 95 seconds for Test 2 and 120 seconds for Test 3, with a standard deviation of 0.00 for Test 1, 0.79 for Test 2 and 1.26 for Test 3.

In the following sections, two scenarios of tests are presented. In the first scenario, the failure tests were performed in the initial environment; to perform the same test afterwards with the active settings. The test was conducted with only one virtual machine to verify the efficiency of the solution. In the second scenario, tests were performed with the largest number of machines possible. This scenario was performed to verify if the solution would be as effective as it had been in the first scenario.

4.1 First Test Scenario

In this stage, it was taken in consideration only if the virtual machine was not lost after the recovery of a node fault.

Figure 3 shows that the virtual machine presents the Error status. Some node faults were simulated so that the Error status appeared. It is worth to highlight that after a node recovers itself from a fault, the virtual machine can no longer be re-used and it was necessary to delete the instance.


Instances

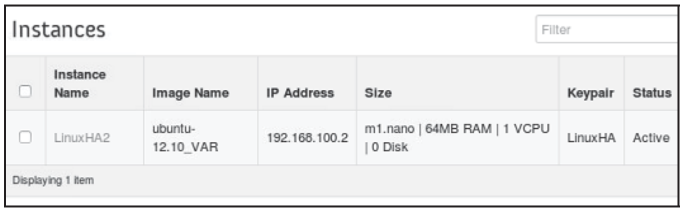
Filter

	Instance Name	Image Name	IP Address	Size	Keypair	Status
☐	LinuxHA1	ubuntu-12.10_VAR		m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Error

Displaying 1 item

Figure 3. Simulating a node fault.

Next, the fault-tolerance configuration was performed using the OpenFiler. As shown in Figure 4, a new virtual machine was initiated, but this time, the OpenStackFT was active.


Instances

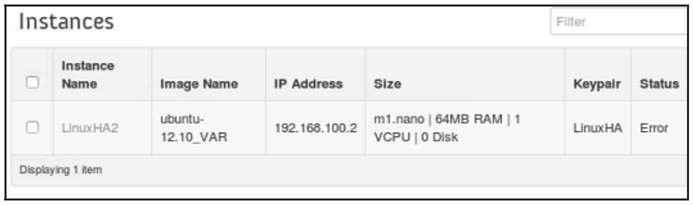
Filter

	Instance Name	Image Name	IP Address	Size	Keypair	Status
☐	LinuxHA2	ubuntu-12.10_VAR	192.168.100.2	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active

Displaying 1 item

Figure 4. Initiating a virtual machine with a fault tolerance solution.

According to Figure 5, the machine status presented Error because node faults were simulated.


Instances

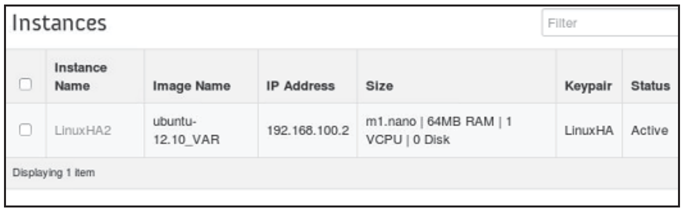
Filter

	Instance Name	Image Name	IP Address	Size	Keypair	Status
☐	LinuxHA2	ubuntu-12.10_VAR	192.168.100.2	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Error

Displaying 1 item

Figure 5. Simulating a node fault with a fault tolerance solution.

After replacing the network cable or reinitializing the node, the machine status became Active again. In this case, there was no need to delete the instance as shown in Figure 6.


Instances

Filter

	Instance Name	Image Name	IP Address	Size	Keypair	Status
☐	LinuxHA2	ubuntu-12.10_VAR	192.168.100.2	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active

Displaying 1 item

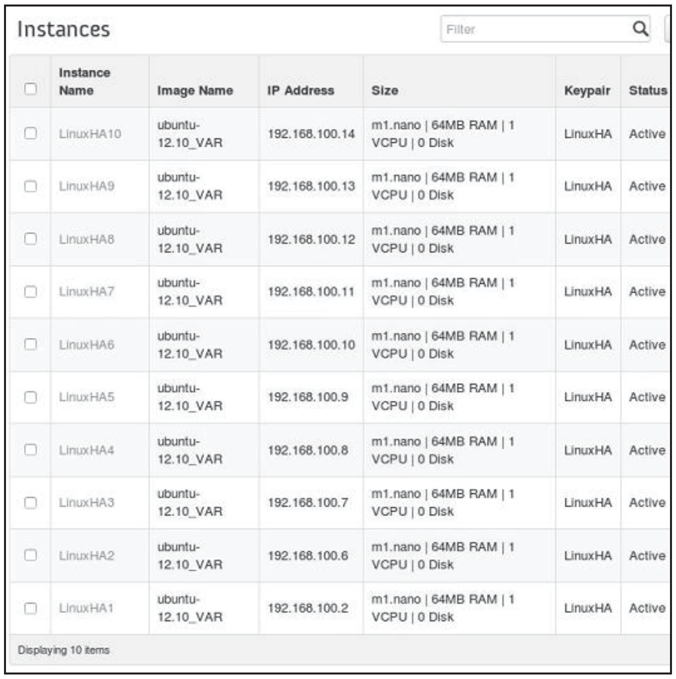
Figure 6. Virtual machine active after a node fault.

[4] Ping is a utility to test connectivity between devices. (CISCO, 2014).

In all performed tests, the behavior observed was always the same. The instance became active after the node recovered itself from the failure.

4.2 Second Test Scenario

The second test scenario was performed in a similar way as the first one, but the test was executed in the maximum of virtual machines available for the environment research. Figure 7 shows the entry of 10 virtual machines.


Instances

Filter

	Instance Name	Image Name	IP Address	Size	Keypair	Status
☐	LinuxHA10	ubuntu-12.10_VAR	192.168.100.14	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active
☐	LinuxHA9	ubuntu-12.10_VAR	192.168.100.13	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active
☐	LinuxHA8	ubuntu-12.10_VAR	192.168.100.12	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active
☐	LinuxHA7	ubuntu-12.10_VAR	192.168.100.11	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active
☐	LinuxHA6	ubuntu-12.10_VAR	192.168.100.10	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active
☐	LinuxHA5	ubuntu-12.10_VAR	192.168.100.9	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active
☐	LinuxHA4	ubuntu-12.10_VAR	192.168.100.8	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active
☐	LinuxHA3	ubuntu-12.10_VAR	192.168.100.7	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active
☐	LinuxHA2	ubuntu-12.10_VAR	192.168.100.6	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active
☐	LinuxHA1	ubuntu-12.10_VAR	192.168.100.2	m1.nano | 64MB RAM | 1 VCPU | 0 Disk	LinuxHA	Active

Displaying 10 items

Figure 7. Entry of 10 virtual machines.

After activating the 10 virtual machines in the environment, the same procedures were performed as in the first scenario. All virtual machines obtained a successful result after the node returned from the fault. In the case of the failure test, the three types of node faults were simulated.

In all the performed tests in the second scenario, the behavior was always the same. Therefore, all virtual machines became active again after the nodes recovered from the faults.

4.3 Results Evaluation

After the implementation and the tests performed in the OpenStackFT, it can be concluded that the solution found solved a fault in the OpenStack: reactivating node instanced virtual machines. Thus, the users will not be affected if some hardware fault occurs in the node in which this virtual machine is linked to. The cloud managers that have implemented the OpenStackFT will also be benefited because they will not need to recreate new virtual machines for the users in case of faults in the nodes.

According to the collected results, the requirements of reliability shown by Tanenbaum and Steen [3] were reached: the availability requirement was reached, since the virtual machine becomes available to the user after a node fault. The reliability requirement was reached, since the virtual machine recovered itself and stood active after the node fault, offering the user the reliability of not losing the virtual machine. The security requirement was reached, since the node was inoperative for some time and, nevertheless, it started functioning normally. Besides, the virtual machines active at the time of the fault were not damaged. The maintainability requirement was reached, since the node that presented a fault was recovered.

Table 1 presents the results obtained with the tests performed in the OpenStackFT. The comparative presents the average and the time standard deviation in seconds that each scenario took to activate the virtual machines after the three failure tests. By observing this table, it is also possible to verify that the higher the number of virtual machines that need to be reactivated, the higher will be the average time that the environment will take to become fully active.

TABLE I. TIME COMPARATIVE

	Scenario 1	**± SD**	**Scenario 2**	**± SD**
Test 1	1 second	0.00	3 seconds	0.00
Test 2	35 seconds	0.83	95 seconds	0.79
Test 3	55 seconds	0.83	120 seconds	1.26

5 Considerations and Conclusions

The difficulties encountered by a user to ensure the high availability of technology services, primarily in open source environments, was an important factor for the development of the OpenStackFT mechanism.

This work is relevant to users and cloud administrators who intend to implement a open source private cloud.. In order that the administrator is able to reproduce the proposed mechanism, it is, first, necessary to have the OpenStack and subsequently implement the OpenStakTF.

6 References

[1] Z. He and Y. He, "Analysis on the security of cloud computing". Proc. Spie, Qingdao, China, n., 2011, p.7752-775204.

[2] F. SABAHI, "Cloud computing security threats and responses". Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on May 2011.

[3] A.S. TANENBAUM and M.V. STEEN, "Distributed Systems: Principles and Paradigms". 2.ed, Pearson Prentice Hall, 2007.

[4] VMWARE, Availability, Accessed on: <http://www.vmware.com/>, February 2016.

[5] AMAZON, Architecture Center of the AWS, Accessed on: <http://aws.amazon.com/pt/architecture/>, February 2016.

[6] CITRIX, Accessed on: <http://www.citrix.com>, February 2016.