Migration of Mobicents SIP Servlets on Cloud Platform

Mr. Manish Giri, Sachin Waghmare, Balaji Bandhu, Akshay Sawwashere, Atul Khaire

Department of Computer Engineering, Maharashtra Academy of Engineering, Alandi, Pune, India

***Abstract*-** Mobicents Sip Servlets helps the users to create, deploy, and manage services and applications that integrate voice, video and data in real time over IP networks. These applications require high resources to run. Mobicents Sip Servlets has not offered a cloud support yet. Running Mobicents Sip Servlets in a cloud would allow the users to adapt the resources used by Mobicents Sip Servlets to the real time load of the applications. We have proposed a solution to migrate Mobicents Sip Servlets on a cloud platform. Our solution provides a set of cloud images that support the Mobicents Sip Servlets platform. Besides, our solution provides an auto-scaling algorithm that automatically scales the number of instances of Mobicents Sip Servlets. It is now possible to deploy SIP applications in a cloud and the platform will adapt its resource according to the need.

***Index Terms*-** Cloud Computing, SIP servlet, JBOSS AS, Cloud images.

I. INTRODUCTION

Mobicents Sip Servlets is an application server which runs on top of JBoss AS and which provides an Implementation of the SIP Servlet specification. The purpose of this platform is to help the users to create, deploy, and manage services and applications that integrate voice, video and data in real time over IP networks. Because of the large amount of data the applications must process, they require high resources to run. Most vendors of application servers now offer an Amazon image in order to use their products on a cloud. Indeed, a cloud can provide the high resources needed by the applications since it is easy to add or remove some resource allocated to the application in a cloud. Mobicents Sip Servlets has not offered a cloud support yet. In order to offer a cloud support, Mobicents Sip Servlets should offer cloud images that would contain the Mobicents Sip Servlets platform. Besides offering these images, another problem arises in cloud environment. This problem is how to scale the number of instances of these images according to the real time load. A cloud platform needs an entity which control the resources usage and which can allocate more or less resources depending on the load. This document will study the migration of Mobicents Sip Servlets on a cloud platform. We will provide cloud images that will contain the Mobicents Sip Servlets platform. Besides, we will propose a solution to scale up and down the resources used by the cloud.

II. CLOUD

2. *Cloud Architecture*

Cloud computing is a client-server approach where the server is a network of nodes. The job of the server is distributed among several nodes that form a cloud. Some nodes can be dynamically added or removed at runtime to the cloud .Depending on the cloud, the client may either send its request to any node in the cloud, or it may send its request to a load balancer, which will act as an entry-point for the cloud and forward the request to a node. Where, a node is an instance of a virtual machine. Thus a node is also called an instance. The virtual machines are executed on a set of inter-connected physical machines. The set of physical machines is called a cluster. There are three types of cloud service models.

2.1.1 IaaS

If the user uses an IaaS, she is basically gaining a computer. The user has access to a virtual machine. She can decide which OS and software install on it. Then, she can use it as any server. An example of IaaS provider is Amazon Web Services.

2.1.2 PaaS

If the user uses a PaaS, she is gaining software or a framework. A PaaS is a hosted tool that the user can leverage to build something on. An example of PaaS is Google AppEngine. With AppEngine, the user can run her web apps on Google's infrastructure. AppEngine provides the hardware, the OS and the framework. All she has to provide is her application. Then, the clients can access her application.

2.1.3SaaS

If the user uses a SaaS, she is gaining business functionality. Basically, a SaaS is an software the user uses, but that is installed on a provider's hardware. For example, all web applications are SaaS. The user uses the software, but she has not had something to install.

2.2 Cloud Properties

Cloud has two main properties: pay per use and elasticity.

2.2.1 Elasticity

The user can scale resources usage up and down rapidly. In the IaaS cloud model, this means that it is possible to start new instances or stop existing instances rapidly. The newly created instances must be used by the cloud in order to decrease the load of the other instances.

2.2.2 Pay per use

The user only pays for the resources that she actually uses. The IaaS cloud providers usually charge an instance on an hourly basis.

III. MOBICENTS SIP SERVLETS

Mobicents Sip Servlets is used to build real time system, such as VoIP software's. The user wants her application to be able to handle a large amount of traffic in a very short time. Mobicents is the most popular Open Source SIP Application Server for the Java platform. It facilitates the implementation of new services in a simple way and quickly enables the development of a Market oriented and cost effective Service Delivery Platform.

IV. RELATED WORK

4.1 Current *situation*

Mobicents Sip Servlets is used to build real time systems, such as VoIP software's. The user wants her application to be able to handle a large amount of traffic in a very short time. Some applications may face a huge amount of traffic. Those applications will require a consequent computation power in order to process all this traffic. In order to ensure a quick response time, a single machine may not be able to handle all the traffic by itself. In this case, the application can be deployed on a cluster. A SIP load balancer receives the traffic from outside and distributes it to the nodes of the cluster. A more sophisticated configuration would contain several SIP load balancers. The JBoss AS cluster provides three properties: elasticity, high availability and failover. Since Mobicents Sip Servlets is built on top of JBoss, those properties are also ensured by the former. Those properties ensure that it is possible to add or remove nodes at runtime in the cluster. The new nodes are automatically added to the cluster and are responsible for processing a part of the traffic. If a node fails or is removed, the traffic that was handled by this node is redirected to the other nodes of the cluster.

4.2 Problem in Current situation

The problem that may occur is that the number of clients of the user's application may vary considerably in a short period of time. At some time in the day, there might be a lot of clients, and at some other time, there might be a few clients. As a result, some resources of the cluster in which the application is running may be unused, or on the contrary, the cluster may be overloaded by the amount of traffic it has to process. A first solution would be to add or remove nodes in the cluster depending of the needs. In order to do so, one would have to monitor the status of each node of the cluster. The monitoring process would then decide if some nodes need to be added or removed, and would do so. However, this solution has two drawbacks. First, it is hard to predict the amount of traffic the application will meet. As a consequence, it is also hard to predict the amount of resources that should be added to or removed from the cluster. Such a prediction would be useful, as it is needed to have those resources available in order to put them in the cluster. Secondly, the resources that are not currently used in the cluster are not costless. Indeed, the user needs to own those resources in order to be able to add them quickly in the cluster. In other words, the problem that a Mobicents Sip Servlets cluster is facing is the waste of unused resources (because they are reserved in case of higher load), or on the contrary, the overload of the cluster (because no other resource is available).

V. PROPOSED SOLUTION

5.1 *Requirements*

The solution of the problem requires that we migrate, Mobicents Sip Servlets to a cloud platform. The goal for the user is to be able to access in some way to some cloud images and to use them. The user shall be able to start a set of images that will compose a Mobicents Sip Servlets platform. The user shall be able to use this platform as a Mobicents Sip Servlets cluster. Finally, to achieve a full usage of the resources, the user shall be able to adjust the number of instances allocated to the Mobicents Sip Servlet platform depending on the load of this platform. Thus, the cloud images are the building blocks of the solution. Once Mobicents Sip Servlets will run on a cloud, we will be able to solve the problem. We will have the resources we need in order to scale according to the load. Because of the pay per use property of the cloud, no resource will be wasted anymore.

5.1.1 Scaling Mode

There are two types of scaling modes provided.

5.1.1.1 Manual-scaling

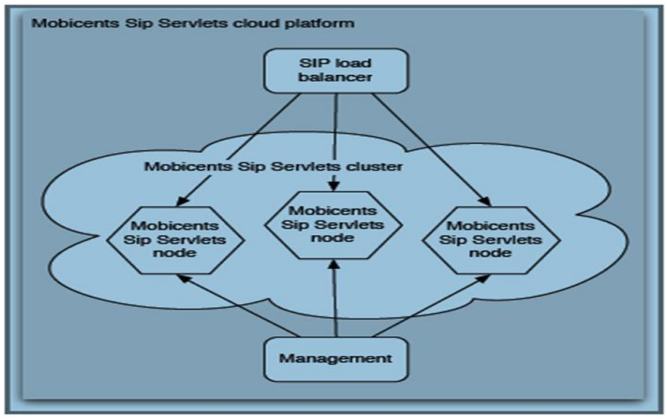
The user is responsible of the scaling. The scaling is made consecutively to a manual action of the user. This requirement will be fully achieved if Mobicents Sip Servlet runs on a cloud, as the user will be able to start or stop some Mobicents Sip Servlets instances.

5.1.1.2Auto-scaling

The environment scales automatically depending of the load. This requires the development of a management solution. The management solution will especially need to define the factors whereby the scaling is made.

5.2 *Images*

In order to put Mobicents Sip Servlets in the clouds, we will define the role of the images that need to be provided. We will provide three images: a Mobicents Sip Servlets image, a Mobicents load balancer image and a management image.



5.2.1 The Mobicents Sip Servlets image

The Mobicents Sip Servlets image will contain an instance of Mobicents Sip Servlets running in clustering mode. The JBoss AS clustering mode will allow several instances of this image to work as a cluster. Besides, thanks to the elasticity provided by the JBoss AS clustering mode, new instances will be

automatically detected by the cluster and added therein. Existing instances will also be allowed to leave the cluster. This image will also contain the applications of the user. Finally, this image will need to be monitored. The monitoring will provide measurements of the auto-scaling factors. Thus, this image will contain an agent, which will communicate with an instance of the management image.

5.2.2 The Mobicents load balancer image

Since the Mobicents Sip Servlets image will work as a JBoss AS cluster, a SIP load balancer is needed in order to route the traffic among the running instances of the Mobicents Sip Servlets image. Although it is possible to have multiple instances of the SIP load balancer running a distributable algorithm, we plan to initially have only one instance of the SIP load balancer. The Mobicents load balancer image will contain, as its name indicates, the SIP load balancer. The Mobicents Sip Servlets instances will need to know somehow the IP address of the Mobicents load balancer instance in order to register to the SIP load balancer.

5.2.3 The Management image

The management image is responsible for collecting measurements of the monitoring factors from the Mobicents Sip Servlets instances. According to those data, the management instance will be responsible for starting or stopping new instances of the Mobicents Sip Servlets image.

5.3 *Auto-scaling*

Auto-scaling is the fact that, depending of the states of all Mobicents Sip Servlets instances, some instances may be started or stopped. The goal to achieve is to have all the instances running to be in a state such as they are not idle, i.e. they are processing a substantial part of the traffic. Also, none of the instances running should be in state such as they are overloaded, i.e. they are not lacking some resources to process the traffic. Auto-scaling is a process that can be decomposed in three steps

5.3.1 Monitoring

It is the action of retrieving the state from an (Mobicents Sip Servlets) instance. This step is done by the agent.

5.3.2 Deciding

It is the ability to decide if an instance should be started or stopped. This step is done by the management system.

5.3.3 Scaling

It is the ability to start or stop instances. This step is done by the management system.

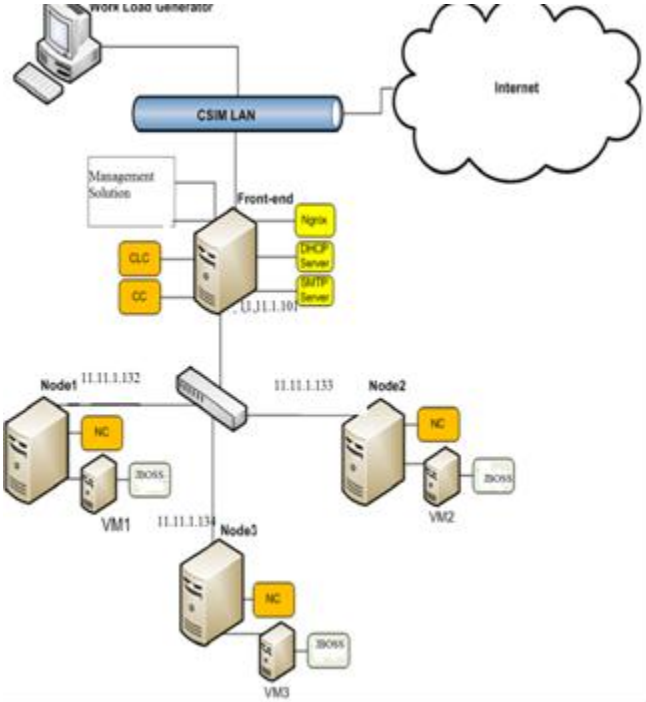
We must also define the state of an Mobicents Sip Servlets instance. This state is defined by a set of metrics. Those metrics are the following:

- The uptime of the instance.
- The CPU load.
- The memory usage.
- The Java heap memory usage of the Java virtual machine, Mobicents Sip Servlets is running on.
- The number of active SIP sessions.
- The number of active SIP application sessions.

We will define two thresholds: the minimum threshold and the maximum threshold. Each threshold will contain a value for each metric that is monitored. Periodically, the scalar will get the list of the instances of each environment it has to monitor from the Nginx (Engine-x) server. Auto Bench is a tool which gives the above parameters. It will then trigger the monitor method and get some values for each metric monitored for each instance. We will have the parameter MONITOR_INTERVAL, which allows the user to set the duration of the periods. Here three sets will be created: below, averages and above. The instances for which all the metrics are below the minim threshold are put in the bellow's set. The instances for which at least one of the metrics is above the maximum threshold are put in the above set. The others (for which the metrics are between the minimum and maximum thresholds) are put in the averages set. If there is not any instance in both the below and the above sets, the system is stable and no action is taken. If there are as many instances in the below as in the above set, the system is well distributed, and no action will be taken. If there are more instances in the below set than in the above set, one instance is stopped. On the other hand, if there are more instances in the above set than in the below set, one new instance is started.

VI. IMPLEMENTATION DETAILS

Below fig shows our proposed system for this project.



6.1 Eucalyptus and NginX

Eucalyptus enables the creation of on-premise private clouds, with no requirements for retooling the organization's existing IT infrastructure or need to introduce specialized hardware. Eucalyptus implements an IaaS (Infrastructure as a Service) private cloud that is accessible via an API compatible with Amazon EC2 and Amazon S3. We stored the virtual machine image in EUCALYPTUS, which cached it on all three NCs.
Nginx is a HTTP and mail proxy server capable of load balancing. We used Nginx as a load balancer because it offers detailed logging and allows reloading the configuration file without termination of existing client sessions. We installed the Nginx as a load balancer on the front-end node and the Tomcat application server on the virtual machine. Our Web application contains one Java servlet that performs a matrix calculation. The Jboss server hosts the Web application while Nginx balances the load. We used Httperf and Autobench to generate a HTTP workload for the Web application. We developed a software component named VLBManager to monitor the average response time of requests by real-time monitoring of Nginx logs. We developed another software component named VLBCoordinator to invoke the virtual machines on the EUCALYPTUS-based cloud. Both components interact with each other using XML-RPC.

The VLBManager detects the violation of the average response time for each virtual machine in the load balancer due to the heterogeneous testbed cloud.

Whenever VLBManager detects a violation of the average response time, it signals the VLBCooridnator to invoke another virtual machine and add it to the load balancer.

6.2 Mobicents SIP servlet s Application: ClicktoCall

This simple Application shows how SIP Servlets can be used along with HTTP servlets as a converged application to place calls from a web page

This application consists of following steps:

1. Alice and Bob each register a SIP Softphone
2. Alice clicks on the "Call" link to place a call to Bob
3. Alice's phone rings
4. When Alice picks up her phone, Bob's phone rings
5. When Bob answers his phone, the call is connected
6. When one of them hangs up, the other one is also disconnected

Starts two sip phones. Open up a browser to http://localhost:8080/click2call/. If you have no registered Sip clients you will be asked to register at least two. Configure your SIP clients to use the sip servlets server as a register and proxy. (Ip address: 127.0.0.1, port: 5080) By default it will accept any password, see below for instructions on how to enable security and authentication. After the registration you will see a table where each cell will initiate a call between the corresponding clients. You can also navigate to http://localhost:8080/click2call/simplecall.html, which is a simplified version that doesn't require registered clients. You will see the index page where you can enter two SIP URIs. Enter the URIs of the two SIP phones you just started and click "Submit". The SIP phones don't have to be registered. After you pick up both phones the RTP session starts.

VII. CONCLUSION

It is possible to propose images with Mobicents Sip Servlets that are ready to run on a cloud. The user only needs to upload its applications to the appliance, and she will be ready to use Mobicents Sip Servlets in a cloud. We have also proposed an Auto-Scaling algorithm which will use the elasticity and pay per use property of cloud. We have proposed to use JBoss as server which will support clustering and uses high availability, failover properties of clusters.

ACKNOWLEDGMENT

We would like to express our gratitude towards a number of people whose support and consideration has been an invaluable asset during the course of this work. I would like to thank the Mobicents community, and more generally the whole JBoss AS community.

REFERENCES

[1] Securing Session Initiation Protocol in Voice over IP Domain . http://ieeexplore.ieee.org . Authors: Alsmairat, I.; Shankaran, R.; Orgun, M.; Dutkiewicz, E.

[2] Eucalyptus Systems, Inc.: Eucalyptus. http://www.eucalyptus.com/

[3] Mobicents : http://www.mobicents.org

[4] The Session Initiation Protocol: Internet-centric signaling. http://ieeexplore.ieee.org . Authors: Dutta, A.; Makaya, C.; Das, S.; Chee, D.; Lin, J.; Komorita, S.; Chiba, T.; Yokota, H.; Schulzrinne.

[5] Eucalyptus Systems, Inc.: Eucalyptus. http://www.eucalyptus.com/.

[6] Mihir Kulkarni and Yannis Cosmadopoulos: Sip servlet 1.1. Specification JSR 289, Oracle, August 2008,

[7] http://jcp.org/aboutJava/communityprocess/_nal/jsr289/index.html.

[8] Red Hat, Inc.: JBoss AS. http://www.jboss.org/jbossas/.

[9] Red Hat, Inc: Mobicents Sip Servlets. http://www.mobicents.org/products sipservlets.html.

[10] Partitioning, and Clustering Working Group: Open Virtualization Format Specification. Specification, [7] DSP0243, Distributed Management Task Force, January 2010.

AUTHORS

First Author – Mr. Manish Giri, Master of Engineering, Maharashtra Academy of Engineering, Pune, E-mail: mbgiri@comp.maepune.ac.in

Second Author – Balaji Bandhu, Bachelor of Engineering, Maharashtra Academy of Engineering, Pune, E-mail: balajibandhu@gmail.com

Third Author – Sachin Waghmare, Bachelor of Engineering, Maharashtra Academy of Engineering, Pune, E-mail: sachinwaghmare3@gmail.com

Fourth Author – Akshay Sawwashere, Bachelor of Engineering, Maharashtra Academy of Engineering, Pune, E-mail: arsawwashere@gmail.com

Fifth Author – Atul Khaire, Bachelor of Engineering, Maharashtra Academy of Engineering, Pune, E-mail: atulkhaire.om@gmail.com