Open Source Cloud Computing Systems:

Practices and Paradigms

Luis M. Vaquero
HP Labs, UK

Juan Cáceres
Telefonica R&D Labs, Spain

Juan J. Hierro
Telefonica, Spain

Information Science
REFERENCE

Managing Director: Lindsay Johnston
Senior Editorial Director: Heather Probst
Book Production Manager: Sean Woznicki
Development Manager: Joel Gamon
Development Editor: Myla Harty
Acquisitions Editor: Erika Gallagher
Typesetter: Christopher Shearer
Print Coordinator: Jamie Snavely
Cover Design: Nick Newcomer, Greg Snader

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Open source cloud computing systems: practices and paradigms / Luis M. Vaquero, Juan Caceres, and Juan J. Hierro, editors.
p. cm.
Summary: “This book bridges the gap between solutions and users’ needs pertaining to the most relevant open source cloud technologies available today from a practical perspective”-- Provided by publisher.
Includes bibliographical references and index.
ISBN 978-1-4666-0098-0 (hardcover) -- ISBN 978-1-4666-0099-7 (ebook) -- ISBN 978-1-4666-0100-0 (print & perpetual access) 1. Cloud computing. 2. Open source software. I. Vaquero, Luis M. II. Caceres, Juan, 1973- III. Hierro, Juan J., 1966-
QA76.585.O64 2012
004.6782--dc23
2011043983

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 3
EMOTIVE Cloud:
The BSC's IaaS Open Source Solution for Cloud Computing

Alex Vaqué
Polytechnic University of Catalonia, Spain & Barcelona Supercomputing Center, Spain

Iñigo Goiri
Polytechnic University of Catalonia, Spain & Barcelona Supercomputing Center, Spain

Jordi Guitart
Polytechnic University of Catalonia, Spain & Barcelona Supercomputing Center, Spain

Jordi Torres
Polytechnic University of Catalonia, Spain & Barcelona Supercomputing Center, Spain

ABSTRACT

This chapter introduces Elastic Management of Tasks in Virtualized Environments (EMOTIVE), which is the Barcelona Supercomputing Center (BSC)'s IaaS open-source solution for Cloud Computing. EMOTIVE provides users with elastic fully customized virtual environments in which to execute their applications. Further, it simplifies the development of new middleware services for managing Cloud systems by supporting resource allocation and monitoring, data management, live migration, and checkpoints. These features and its facility to be extended and configured make EMOTIVE especially appropriate to support research on Cloud Computing scenarios. Offering functionality comparable to its commercial counterparts allows EMOTIME to be used on production to set up small Cloud platforms.

INTRODUCTION

EMOTIVE (Elastic Management of Tasks in Virtualized Environments) (EMOTIVE, 2009) is the Barcelona Supercomputing Center (BSC)'s IaaS open-source solution for Cloud Computing, which results from BSC's previous experience in European projects such as BREIN (BREIN, 2006-2009) and SORMA (SORMA, 2006-2009). EMOTIVE provides users with elastic fully customized virtual environments (supporting different hypervisors such as Xen, KVM, or VirtualBox) in which to execute their applications. Further, it simplifies

DOI: 10.4018/978-1-4666-0098-0.ch003

the development of new middleware services for managing Cloud systems by supporting resource allocation and monitoring, data management, live migration, and checkpoints.

EMOTIVE enables the smart management of the virtual environments using different scheduling policies. Additionally, it is very easy to extend thanks to its modular Web Service architecture. This framework is being used by BSC to do research in Cloud Computing, as well as in some research projects such as VENUS-C (VENUS-C, 2010-2012), OPTIMIS (OPTIMIS, 2010-2013), and NUBA (NUBA, 2009-2012).

In this chapter, we will describe the main functionalities of EMOTIVE, how they are implemented, and how EMOTIVE can be used to set up a private, public, or hybrid Cloud solution.

General Architecture of the Solution

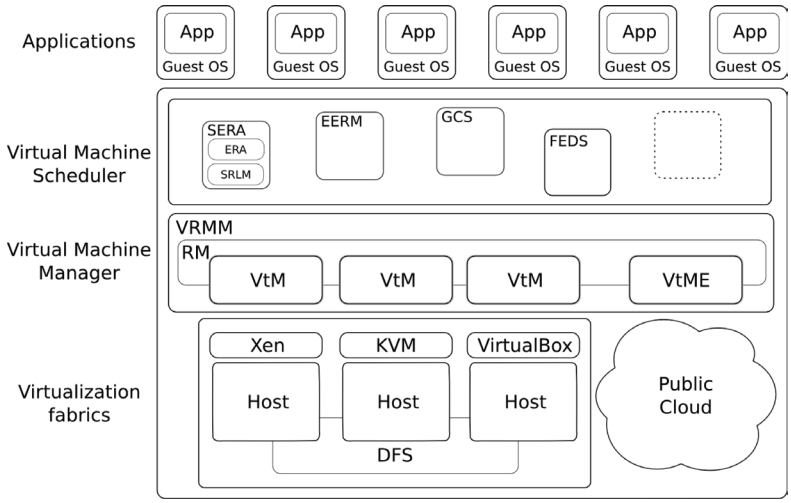
EMOTIVE middleware can be categorized as an IaaS solution, since it provides the users with virtualized environments where they can execute their tasks without any extra effort. These VMs, which aim to fulfill the user requirements in terms of software and system capabilities, are transparently managed by EMOTIVE in order to exploit the provider's resources. EMOTIVE can easily be extended with multiple scheduling policies in order to manage the VMs using different criteria.

Figure 1 illustrates the EMOTIVE Cloud architecture, which is mainly composed by three different and modular layers: *Virtualization Fabrics*, *Virtual Machine Manager*, and *Virtual Machine Scheduler*.

The *Virtualization Fabrics* layer comprises the physical resources where the VMs will run. This layer wraps the virtualized resources and offers them to the upper layers. EMOTIVE makes use of the Libvirt JAVA API (Libvirt, 2005), which makes it able to use multiple virtualization technologies. Actually, EMOTIVE currently supports Xen, KVM, and VirtualBox hypervisors. Furthermore, it implements a distributed shared file system (DFS) that supports efficient VM creation, migration (to move VMs across provider's hosts without stopping the execution), and checkpointing (to resume VM execution upon hardware failure). This file system also supports a global repository where users can upload the input files needed by the applications (i.e. data stage-in) and retrieve the resulting ones (i.e. data stage-out).

The *Virtual Machine Manager* layer is implemented by means of the *Virtualized Resource*

Figure 1. EMOTIVE cloud architecture



Management and Monitoring (VRMM), which includes several subcomponents. On one side, there is one Virtualization Manager (VtM) per physical host, which is in charge of creating and maintaining the whole virtual machine life cycle (create, destroy, migrate, etc.). VMs are created on demand, according to the application requirements, both hardware (CPU type, amount of resources required) and software (required packages). These requirements are specified by means of the Open Virtualization Format (OVF) (DMTF, 2010). VPN support (with SSL and PPTP protocol) and Virtual Networks (VLAN) are also available for the VMs. Once the VMs are created, the users can work directly with them using a SSH connection, or use the EMOTIVE API to execute tasks in them. Tasks are described by means of a Job Submission Description Language (JSDL) (OGF, 2008) file.

In addition, the VtM comprises all the local resource management decisions (i.e. in a single host): it is in charge of managing the physical resources of a host and dynamically distributing these resources among all the VMs running on that host in order to fulfill their respective Service Level Agreements (SLAs). EMOTIVE allows specifying fine-grain resource-level guarantees in the SLA (e.g. amount of computing power allocated to a given VM over time) (Goiri, 2010b), which are clearly superior to the availability guarantees supported by common providers such as Amazon EC2 (Amazon, 2006).

Furthermore, EMOTIVE also has, by means of the VtME component, the capability to use external resources, like the ones in public Cloud providers (i.e. Amazon EC2). This feature allows an EMOTIVE-enabled provider to be involved in a Cloud federation (insourcing/outsourcing) and create public, private, and hybrid clouds.

On the other side, the Resource Monitor (RM) component continuously monitors the status of tasks and resources. This status is stored in a historical database, but it can be also used to assess the fulfillment of the SLAs of the applications. If any SLA violation is detected, an adaptation process for requesting more resources to the provider is started, first locally in each host, then globally in the provider, and finally with other providers.

Finally, the *Virtual Machine Scheduler* layer comprises all the global VM placement decisions, both among different providers in a Cloud federation and different hosts in a single provider. This layer is in charge of deciding where a VM will be executed and managing its location during the execution (e.g. migration of VMs across provider's hosts, cancellation of VMs, resumption of VM execution from a checkpoint upon hardware failure, etc.). As a rule of thumb, the Scheduler tries to consolidate the VMs in the provider's physical resources to optimize their use, while allocating enough resources to fulfill the agreed SLAs.

Moreover, this framework allows multiple schedulers with different policies and capabilities such as machine learning, prediction, economic, fault tolerance, semantic description, or SLA enforcement. In this sense, it can use a simplistic Round Robin, or a consolidation-aware scheduling like Backfilling. This is achieved thanks to the usage of a common interface that allows developing new schedulers with different features and policies. In particular, we encourage using the Open Cloud Computing Interface (OCCI) (OGF, 2010), which allows EMOTIVE to be interoperable with other Cloud middleware supporting this interface.

Related Work

In the last years, numerous solutions for supporting IaaS in the Cloud have appeared. Whereas the majority of them provide similar functionality, there are some key factors that could be used to choose one over the others. We can firstly distinguish between proprietary and open-source solutions. Amazon EC2 is the main example of the former, while this book includes the most significant open-source solutions. The latter includes EMOTIVE, which has been released under a LGPL license.

In this way, it can be easily modified and adapted to the user's requirements.

We can then distinguish between production-aimed and research-aimed solutions. The former have strong requirements on stability, and for this reason, they tend to use well-known (and simple) management procedures. For this reason, they typically have low rates of resource utilization and energy efficiency. Production-aimed solutions have also powerful security constraints and that makes them sometimes difficult to deploy and configure (e.g. Eucalyptus [Nurmi, 2009]). Being a research-aimed solution, EMOTIVE provides their users with basic capabilities that can be used to implement complex management procedures for Cloud middleware. In addition, it allows to setup Cloud infrastructures that offer VMs on demand, while introducing minimal overhead, being easily deployable and configurable, and also highly scalable thanks to EMOTIVE layered distributed architecture where every node is almost autonomous and controls its own resources.

Finally, we can distinguish between interoperable and stand-alone solutions. EMOTIVE supports the most common Cloud standards, both at interface level (e.g. OCCI, OVF …) and at virtualization level (e.g. Xen, KVM, Libvirt …). In fact, it was originally designed to be used in Cloud federations. This makes it highly interoperable with the most popular Cloud solutions (e.g. Amazon EC2, OpenNebula [Fontan, 2008]).

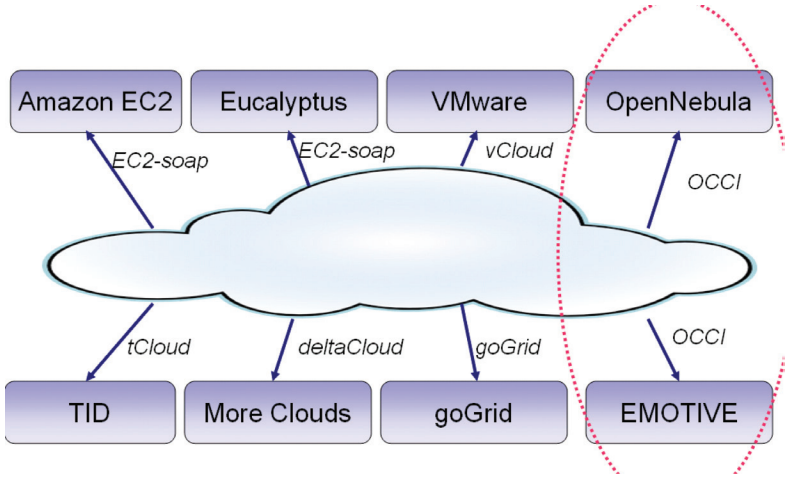
Whereas in some aspects EMOTIVE capabilities are not as powerful as the capabilities of their counterparts, this is clearly compensated by offering a controllable, scalable, interoperable, and extensible solution to set up IaaS Clouds.

Most Relevant Interfaces

The problem with interoperability in Cloud providers is well-known. As shown in Figure 2, different Cloud providers use their own and independent interface. This makes it difficult to communicate and federate multiple providers. Recently, OCCI API has been proposed as a common standard in order to overcome this problem. OCCI is a Cloud Interaction Layer which uses HTTP methods (like GET, POST, PUT, DELETE) using XML format. This interface uses multiple data structures (i.e. Compute, Network, Storage) to describe the different resources. Using these structures, it can operate the virtual resources (i.e. create, list, show, update, delete).

EMOTIVE was originally designed using a distributed SOAP architecture but now it uses RESTful Web Services. This architecture allows

Figure 2. Interfaces of different clouds



the usage of only some parts of EMOTIVE and supports agile and dynamic construction of new Cloud environments. Its REST interface makes EMOTIVE highly interoperable with other Cloud solutions.

Furthermore, popular Cloud solutions such as OpenNebula have adopted OCCI to define their interfaces. Aiming at interoperability with other Cloud solutions, EMOTIVE also implements an OCCI interface. Notice, however, that the standard OCCI interface does not support all the original EMOTIVE functionality. For this reason, there are some methods for job and cluster management that EMOTIVE supports using its original REST interface. According to this, EMOTIVE Cloud currently supports two interfaces: EMOTIVE REST API and the standard OCCI. In the following lines, we describe briefly these two interfaces.

OCCI describes five methods that use Compute and four for Network and Storage. EMOTIVE supports four of the Compute methods, four Network methods, but it does not support Storage methods. However, this does not mean that EMOTIVE does not support storage at all. In fact, EMOTIVE allows the specification of bootable images for VMs in the disk section of the OVF parameter that describes the VMs in the Create methods. EMOTIVE is able to parse this description and get the VM image from the location specified by the user (e.g., an FTP server, an Amazon S3 repository).

The methods comprising the EMOTIVE REST interface are described in Table 1. The methods with a correspondence in the OCCI interface are shown boldfaced. Our interfaces basically allow:

- **Compute:** create, get, list, and cancel Virtual Machines (we use OVF format to describe the VMs).
- **Network:** similar to Compute methods but used to describe virtual networks.

Table 1. EMOTIVE REST API. Correspondence with OCCI methods is noted in bold.

COMPUTE
· **String Env-ID = createEnvironment (Compute)** · String Env-ID = createEnvironmentAndJob (Compute, JSDL) · **terminateEnvironment (String Env-ID)** · **List <Env-ID> = getEnvironments ()** · **Compute = getEnvironment (String Env-ID)** · String state = getEnvironmentState (String Env-ID)
NODES
· String [Node-ID or Env-ID] = getLocation (String [Env-ID or Act-ID]) · List <Node-ID> = getNodes () · nodeDown (String Node-ID) · nodeUp (String Node-ID)
JOBS
· List <Act-ID> = getActivities () · Act-ID = submitActivity (JSDL) · cancelActivity (String Act-ID) · String status = getActivityStatus (String Act-ID) · List <String Act-ID> = getAllActivities ()
NETWORK
· **String Net-ID = createNetwork (Network)** · **deleteNetwork (String Net-ID)** · **Network = getNetwork (String Net-ID)** · **List <Network> = getListNetworks ()** · String Net-ID = createVPN (Network)

- **Jobs:** used to submit jobs to Virtual Machines (we use JSDL format to describe the jobs).
- **Nodes:** describes the system topology (used for EMOTIVE internals).

Table 2 shows the equivalence between the methods used in EMOTIVE REST API and OCCI API. It basically describes the mapping of the OCCI REST methods to the EMOTIVE REST methods. In fact, this is how we have implemented our support to the OCCI API, that is by means of a wrapper that translates OCCI methods to EMOTIVE REST ones.

Regarding the data structures used to describe the resources, our *createEnvironment(Compute)* method is able to support the same *Compute* structure used in OpenNebula. An example of this *Compute* structure is shown below.

```
<COMPUTE href="http://www.open-
nebula.org/compute/32">
<ID>12342-4356-12345-24324</ID>
<NAME>Web Server</NAME>
<STATE>running</STATE>
<DISK>
<STORAGE href="http://www.open-
nebula.org/storage/34"/>
<TYPE>OS</TYPE>
<TARGET>hda</TARGET>
</DISK>
<DISK>
<STORAGE href="http://www.open-
nebula.org/storage/24"/>
<TYPE>CDROM</TYPE>
<TARGET>hdc</TARGET>
</DISK>
<NIC>
<NETWORK href="http://www.open-
nebula.org/network/12"/>
<MAC>00:ff:72:31:23:17</MAC>
<IP>192.168.0.12</IP>
</NIC>
</COMPUTE>
```

Similarly, an example of *Network* structure, which is used in *createNetwork(Network)* method is in the following.

```
<NETWORK href="http://www.open-
nebula.org/network/12">
<MAC>00:ff:72:31:23:17</MAC>
<IP>192.168.0.12</IP>
</NETWORK>
```

Table 2. Methods used in EMOTIVE cloud

COMPUTE	
EMOTIVE Methods (Java)	***API OCCI (REST)***
createEnvironment(Compute)	/compute POST (PR)
terminateEnvironment(String id)	/compute/id DELETE (ER)
getEnvironments()	/compute GET (PR)
getEnvironment(String id)	/compute/id GET (ER)
NETWORK	
EMOTIVE Methods (Java)	***API OCCI (REST)***
createNetwork(Network)	/network/id POST (PR)
getNetworks()	/network GET (PR)
deleteNetwork(String id)	/network/id DELETE (ER)
getNetwork(String id)	/network/id GET (ER)

Instead of using *Compute*, our *createEnvironment(Compute)* and *createEnvironmentAndJob (Compute, JSDL)* methods can also support the usage of simple Open Virtualization Format (OVF) files to describe the features of the VMs to be created. The following is an OVF example of one simple VM with 2 CPUs and 2GB of memory.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns1:Envelope xmlns:ns2="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_VirtualSystemSettingData" xmlns:ns1="http://schemas.dmtf.org/ovf/envelope/1" xmlns:ns4="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/CIM_ResourceAllocationSettingData" xmlns:ns3="http://schemas.dmtf.org/wbem/wscim/1/common">
<ns1:References>
<ns1:File ns1:href="/cosa/fina.img" ns1:id="root"/>
<ns1:File ns1:size="1073741824" ns1:id="home"/>
</ns1:References>
<ns1:VirtualSystem>
<ns1:Info>EMOTIVE Cloud Virtual Machine Description</ns1:Info>
<ns1:VirtualHardwareSection>
<ns1:Item>
<ns4:AllocationUnits>cpu</ns4:AllocationUnits>
<ns4:Description>Number of CPUS</ns4:Description>
<ns4:ElementName>x86</ns4:ElementName>
<ns4:InstanceID>1</ns4:InstanceID>
<ns4:ResourceType>3</ns4:ResourceType>
<ns4:VirtualQuantity>2</ns4:VirtualQuantity>
</ns1:Item>
<ns1:Item>
<ns4:AllocationUnits>byte * 210</ns4:AllocationUnits>
<ns4:Description>RAM Memory</ns4:Description>
<ns4:ElementName>2046MB of Memory</ns4:ElementName>
<ns4:InstanceID>2</ns4:InstanceID>
<ns4:ResourceType>4</ns4:ResourceType>
<ns4:VirtualQuantity>2046</ns4:VirtualQuantity>
</ns1:Item>
<ns1:Item>
<ns4:Caption>Home drive</ns4:Caption>
<ns4:HostResource>ovf:/file/home</ns4:HostResource>
<ns4:InstanceID>3</ns4:InstanceID>
<ns4:ResourceType>17</ns4:ResourceType>
</ns1:Item>
<ns1:Item>
<ns4:Caption>Root drive</ns4:Caption>
<ns4:HostResource>ovf:/file/root</ns4:HostResource>
<ns4:InstanceID>4</ns4:InstanceID>
<ns4:ResourceType>17</ns4:ResourceType>
</ns1:Item>
</ns1:VirtualHardwareSection>
</ns1:VirtualSystem>
</ns1:Envelope>
```

In addition, EMOTIVE supports Job Submission Description Language (JSDL) to submit jobs using the methods *submitActivity(JSDL)* and *createEnvironmentAndJob (Compute, JSDL)*. JSDL is an extensible XML specification for describing requirements of computational jobs. It was initially focused in Grid but it is not restricted to this environment. JSDL describes: job name, description, resource requirements (RAM, swap, CPU, number of CPUs, operating System, etc.), execution limits, file staging, command to execute… The following is an example of an ANSYS CFX simulation *JSDL*.

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:JobDefinition xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl" xmlns:jsdl-hpcpa="http://schemas.ggf.org/jsdl/2006/07/jsdl-hpcpa">
<jsdl:JobDescription>
<jsdl:JobIdentification>
<jsdl:JobName>AnsysDemo</jsdl:JobName>
</jsdl:JobIdentification>
<jsdl:Application>
<jsdl:ApplicationName>AnsysCfx</jsdl:ApplicationName>
<jsdl:ApplicationVersion>PM26</jsdl:ApplicationVersion>
<jsdl-hpcpa:HPCProfileApplication>
<jsdl-hpcpa:Argument>-cpu_load=1.0</jsdl-hpcpa:Argument>
<jsdl-hpcpa:Argument>-threads_num=2</jsdl-hpcpa:Argument>
</jsdl-hpcpa:HPCProfileApplication>
</jsdl:Application>
</jsdl:JobDescription>
</jsdl:JobDefinition>
```

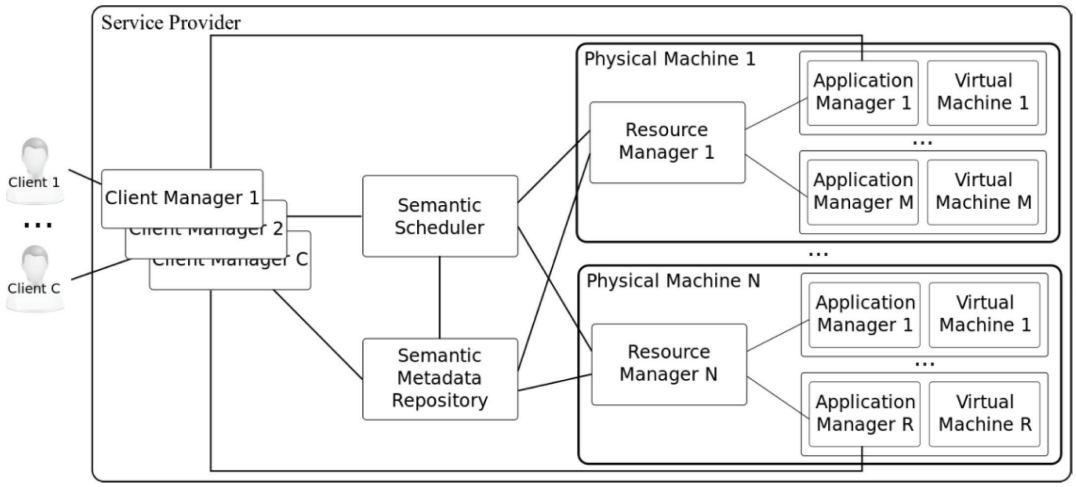
Relevant Use Cases

EMOTIVE enables the smart management of virtual environments using different scheduling policies. Additionally, it is very easy to extend it thanks to its modular Web Service architecture. According to this, this framework is being used in BSC and UPC to do research in Cloud Computing, as well as in some research projects such as BREIN, OPTIMIS, VENUS-C, and NUBA. This is the main objective of EMOTIVE Cloud and not to be a product like Eucalyptus, OpenNebula, or OpenStack. There are a lot of research works that use EMOTIVE Cloud to create test environments or to perform research about virtual machines scheduling or management. In the following lines, we summarize some of the research works that have used EMOTIVE as IaaS Cloud solution.

SERA Scheduler: Within the BREIN European project, BSC has developed a Semantically-Enhanced Resource Allocator (SERA) (Ejarque, 2008; Goiri, 2009), which was an initial version of EMOTIVE that distributed resources using semantic information and used agents to communicate and support each component. In SERA, tasks and resources are semantically described and these descriptions are used to infer the resource assignments. Virtualization is used to provide a full-customized and isolated virtual environment for each task. In addition, the system supports fine-grain dynamic resource distribution among these virtual environments based on SLAs. The required adaptation is implemented using agents, guarantying to each task enough resources to meet the agreed performance goals.

Figure 3 shows the main components of SERA. The Client Manager (CM) manages the client's task execution by requesting the required resources and by running jobs. In addition, it makes decisions about what must be done when unexpected events such as SLA violations happen. The Semantic Scheduler (SeS) allocates resources to each task according to its requirements, its priority and the system status, in such a way that the clients with

Figure 3. Architecture of SERA



more priority are favored. Allocation decisions are derived with a rule engine using semantic descriptions of tasks and physical resources. These resource descriptions are automatically generated from the system properties and stored in the Semantic Metadata Repository (SMR) when the machine boots. The Resource Manager (RM) creates virtual machines (VM) to execute clients' tasks according to the minimum resource allocation (CPU, memory, disk space...) given by the SeS and the task requirements (e.g. needed software). Once the VM is created, the RM dynamically redistributes the remaining resources among the different tasks depending on the resource usage of each task, its priority and its SLA status. This resource redistribution mechanism allows increasing the allocated resources to a task by reducing the assignment to other tasks that are not using them. Finally, the Application Manager (AM) monitors the resource usage in order to evaluate if an SLA is being violated.

Multifaceted Scheduler: Multifaceted scheduler (Goiri, 2010a) implements a new scheduling policy to model and manage a virtualized data center, which mainly focuses on the allocation of VMs in the data center hosts according to multiple facets while optimizing the provider's profit. In particular, it considers energy efficiency, virtualization overheads, fault tolerance, and SLA violation penalties, while adding the ability to outsource resources to external providers.

This multifaceted scheduler is directly implemented on the *Scheduler* layer of the EMOTIVE architecture. It takes advantage of the capabilities of the underlying layer. For example, in addition to the dynamic creation of VMs, it uses the efficient migration and the checkpointing mechanisms. The scheduling policy is run periodically and every time a task arrives.

High Availability Scheduler: High-availability scheduler (Alonso, 2011) allows managing and reconfiguring virtualized platforms, offering a transparent mechanism to overcome the software failures at application level. It presents a set of strategies to guarantee the availability of the services while accepting the maximum possible number of services in the platform.

From the provider's perspective, every customer's service is composed by a VM where the service is deployed (Service VM) and a user-transparent VM for high availability purposes (HA

VM). The HA VM is composed by two software components: the Failure Predictor (FPdr) and the High Availability Load Balancer (HA-LB).

The High-availability scheduler uses EMOTIVE capabilities to rapidly create VMs. When the system detects that a VM is about to crash, it uses EMOTIVE to create a new worker and replace the previous one. In addition, it is able to migrate VMs to provide consolidation while the services are running.

Cloud Hosting Provider: Cloud Hosting Provider (CHP) (Fitó, 2010) is an elastic web hosting provider that makes use of the outsourcing technique to take advantage of Cloud computing infrastructures for providing scalability and high availability capabilities to the web applications deployed on it.

This system includes a Scheduler that manages the provider's in-house resources to run client's web application, monitors high-level performance metrics (such as the response time) and outsources additional VMs from third-party providers to execute more web servers when these performance metrics are not fulfilled (i.e. typically when the in-house servers become overloaded).

In this work, both the provider's in-house resources and the external resources are implemented using

EMOTIVE by directly using the external API to request new VMs and running web servers on top of them.

COMPs Superscalar: COMPs Superscalar (Tejedor, 2008) exploits the inherent parallelism of applications when running them on the Grid. It is a new programming paradigm for Grid enabling applications, composed of an interface and a run-time. COMP superscalar is a software tool developed at BSC. It was originally intended to be used in a cluster of physical servers. Currently, BSC is developing the same idea but it uses EMOTIVE Cloud to build an underlying Cloud infrastructure where COMPs Superscalar is deployed. COMPs Superscalar with EMOTIVE is being used in the European project VENUS-C. The first implementation uses EMOTIVE directly by calling the OCCI methods to create and destroy new workers in the COMPs architecture.

Hadoop Scheduler: Hadoop Scheduler (de Nadal, 2010) appears like a possible solution to the difficulties in resources provisioning for computations in Hadoop environments. Virtualization can solve those management problems for Hadoop as it does for other software solutions, giving dynamic resource allocation capabilities, which give the customer the possibility of making requests to the system and, if possible with the available resources, set up the virtual Hadoop environments, or modify existing ones according with the requirements.

The implemented system virtualizes and manages Hadoop environments by using EMOTIVE Cloud, and provides an interface to interact with the internal job scheduler present in Hadoop for collecting information about the job progresses over the running jobs. Hadoop Scheduler uses this information to decide how to distribute the physical resources.

INSTALLATION AND DEPLOYMENT GUIDE

This section presents how to install EMOTIVE in a provider and explains step by step how to deploy the system using Xen and Tomcat to run the applications. The system can run with other virtualization hypervisors like KVM and VirtualBox. The only difference with respect to the procedure presented in this section is the setup of the hypervisor that has to operate with Libvirt.

Prerequisites

Before starting, EMOTIVE requires the installation and setup of some base software in the hosts that will be part of the system. The minimum requirements for installing and use EMOTIVE Cloud are:

- Debian 3.0 or higher (other GNU/Linux distributions can be used)
- Xen 3.1.0 or higher and/or KVM 2.6.28.1
- Java 1.5 or higher
- Libvirt 0.7.5 or higher
- Apache Tomcat 5 or higher.

Step by Step Procedure from Scratch

1. Install basic packages:
   a. Xen and Libvirt:

```
# apt-get install xen-linux-sys-
tem-2.6.26-2-xen-amd64 libvirt0
libvirt-bin xen-hypervisor-3.2-1
```

   b. Java:

```
# apt-get install openjdk-6-jdk
```

   c. Tomcat:

```
# apt-get install tomcat6
```

   d. Others:

```
# apt-get install libc6-dev
zlib1g-dev debootstrap dhcp3-
server bind9 module-init-tools
```

   e. Maven and SVN to compile the code:

```
# apt-get install make ant ma-
ven2
```

2. Create the EMOTIVE directory and download EMOTIVE source code into this directory:

```
# mkdir /usr/share/EMOTIVE
# svn co https://emotivecloud.
svn.sourceforge.net/svnroot/emo-
tivecloud//EMOTIVE_PATH*
```

   Alternatively, binaries can be downloaded from: http://sourceforge.net/projects/emotivecloud/files/

3. Edit "install.cfg" file:

```
# Your APACHE directory
export APACHE_PATH=/aplic/igo-
iri/apache-tomcat-6.0.24/
# Your EMOTIVE directory
export EMOTIVE_PATH=/usr/share/
EMOTIVE
```

4. Install EMOTIVE Cloud using the installation script:

```
# ./install full
```

5. The installation script compiles and installs EMOTIVE into Apache Tomcat (puts the EMOTIVE WAR files into the webapp directory), but additionally it is necessary to create "/etc/VtM/rm.properties" and "/etc/VtM/vtm.properties" configuration files. First of all, install files in "/etc/VtM" and "domU":

```
# ./install create
```

   After that, it is required to check the domain name in "/etc/VtM/rm.properties," the paths in "/etc/VtM/vtm.properties," and finally create the domU apps, which are some required extra applications required to operate the VMs:

```
# ./install domu
```

6. Start the application server that will run EMOTIVE Cloud on top:

```
# ./catalina.sh run
```

7. Finally, when the core has already started, the system can be tested using some clients.

To run these clients, JAR files can be used. The clients interact with the "Scheduler" and the "VtM":

```
# java -jar SimpleSchedulerClient.
jar
# java -jar VtM.jar
```

It is also possible to get the JAR and WAR files in our web page and put them manually in the Apache Tomcat webapp directory and run the Apache Tomcat that has EMOTIVE on top. However, it is recommendable to use the installation script because this generates the *pool* (directory that will contain the images, it will be described in detail later) and configuration files (in "/etc/VtM/*.cfg") of EMOTIVE.

Xen Configuration

This section gives an overview of some steps that need to be performed to setup Xen properly. First of all, in order to increment number of disk images usable by Xen, add the next option to the kernel (for instance, in grub options):

```
max_loop=128
```

Different systems can have different network interfaces. In order to avoid this problem, a generic interface must be created, for instance, "brein0." If not, you can use the usual network interface "ethX." Replace default network-script line in ``/etc/xen/xend-config.sxp" by:

```
(network-script 'network-bridge
bridge=brein0 netdev=ethX')
```

Finally, reboot and in the next system boot select the Xen 3.3.1 kernel and the system will be ready to run VMs thanks to Xen. Notice that, Xen 3.3.1 may have some problems when booting from a SATA boot-disk.

In order to enable migration, we must modify Xend configuration file by changing next lines in "/etx/xen/xend-config.sxp"' in every machine involved in migration:

```
(xend-http-server yes)
(xend-relocation-server yes)
(xend-address '')
(xend-relocation-hosts-allow '')
```

Restarting Xen will enable live migration to other machines. This will migrate specified domains to the other host without losing connectivity.

Network Bridge Preparation

In order to use the networking capabilities of EMOTIVE, you have to prepare the network bridge. First of all, install the bridge-utils using "apt-get install bridge-utils" or similar.

```
# apt-get install bridge-utils
```

Later, you need to create a new net interface. This net interface allows doing a bridge with the new virtual machines and the network. So we need to define a bridge in "/etc/network/interfaces."

You need to modify the net interfaces file, so you have defined the new bridge interface (br0) and disabled the current (brein0 for example). The following is "/etc/networks/interfaces."

```
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
#auto eth0
#iface eth0 inet manual
#auto brein0
iface brein0 inet static
address 172.20.0.101
netmask 255.255.0.0
network 172.20.0.0
broadcast 172.20.255.255
```

```
gateway 172.20.0.1
dns-nameservers 172.20.0.1
dns-search edx

#auto eth0
iface eth0 inet static
address 172.20.0.101
netmask 255.255.0.0
network 172.20.0.0
broadcast 172.20.255.255
gateway 172.20.0.1
dns-nameservers 172.20.0.1
dns-search edx
auto br0
iface br0 inet static
address 172.20.0.102
netmask 255.255.0.0
network 172.20.0.0
broadcast 172.20.255.255
gateway 172.20.0.1
dns-nameservers 172.20.0.1
dns-search edx
bridge-ports eth0
bridge_fd 9
bridge_hello 2
bridge_maxage 12
bridge_stp off
```

We also have to make sure that bridge works correctly: with the command 'brctl show,' we show the state of the bridge and virtual network interfaces.

```
# brctl show
bridge name bridge id STP en-
abled interfaces
br0 8000.0015177e9660 no eth0
virnet0
vnet0
vnet1
```

To conclude with the network settings, add the following lines to: /etc/sysctl.conf:

```
net.bridge.bridge-nf-call-ip6ta-
bles = 0
net.bridge.bridge-nf-call-ipta-
bles = 0
net.bridge.bridge-nf-call-arpta-
bles = 0
```

We load this with 'sysctl -p /etc/sysctl.conf' command.

EMOTIVE Pool

As mentioned before, EMOTIVE uses a path in each host to locate local files such as VM images and user disks, which we will refer as *pool*. This section gives a detailed view about how to setup the pool.

To setup EMOTIVE, first of all we have to create empty EMOTIVE pool directories. For example, it can be located in "/usr/share/EMOTIVE/pool."

```
#./install.sh directory
```

EMOTIVE will use this pool to store VM images and user's data. If this pool is not created before, EMOTIVE will create the base directories at run time.

```
pcroot# ls /usr/share/EMOTIVE/
pool/pcroot/
cache checkpoint domU extensions
home images kernels log
```

You need to create and copy the disk image in "/usr/share/EMOTIVE/pool/pcroot/images/default.img" and put the kernels used in the virtual machines in "/usr/share/EMOTIVE/pool/pcroot/kernels/."

```
pcroot:~# ls -ltr /usr/share/
EMOTIVE/pool/pcroot/images/
total 1025004
```

```
-rw-r-r- 1 root root 1048576000
2010-06-04 13:59 default.img
pcroot:~# ls -ltr /usr/share/
EMOTIVE/pool/pcroot/kernels/
-rw-r-r- 1 root root 13654779
2010-06-04 14:09 libmodules-de-
fault.tar.gz
-rw-r-r- 1 root root 2670757
2010-06-04 14:11 vm-
linuz-2.6.18.8
-rw-r-r- 1 root root 1535487
2010-06-04 14:11 vmlinuz-default
```

Common Pitfalls

EMOTIVE installation is easy because it only needs to put JAR files into Apache Tomcat (webapps directory). The hardest part is the EMOTIVE pre-installation, configuring the hypervisor, the Libvirt API, and the Libvirt bridge configuration to communicate with the hypervisor.

The most typical error is in the EMOTIVE environment preparation, mainly due to the bad configuration of system environments variables. Check the next files:

- "/etc/VtM/vtm.properties": This file is necessary to know all system environments variables: folders, domains, defaults settings, network configuration, server and system architecture, etc.
- "/etc/VtM/rm.properties": This file is necessary to know the network topology. The file has the hostname of the hosts used by EMOTIVE: schedulers and VtM's hosts.

Another typical error is having the EMOTIVE pool with wrong configuration. For example, the directory location, the disk images and kernels into wrong directories, etc.

Finally, there is a small possibility that EMOTIVE is unable to create a good initial virtual image (*.img). To solve this, it is necessary to create it manually, or download another ISO image and put this image into EMOTIVE pool directory.

FAQ

Q: Should I use Xen or KVM hypervisor?

A: It depends on which Linux distribution you are using. We prefer to use Xen in Debian distributions, but Ubuntu has better support for KVM, so in Ubuntu we prefer KVM. Generally, it is easier to install and configure Xen but the most important is the possibility to use the Linux distribution repositories. Some distributions have better support for Xen and others for KVM. The best solution is to install the hypervisor with apt-get, aptitude, yum, zypper or others.

Q: What is the best Linux distribution to use EMOTIVE?

A: EMOTIVE has been successfully installed in Debian, Ubuntu, and CentOS distributions.

Online Support and Community Aspects

EMOTIVE is the BSC/UPC IaaS open-source solution for Cloud Computing, which results from BSC's previous experience in European and national projects. EMOTIVE has limited support because it was originally developed to support BSC and UPC research. There are some Web resources and tutorials which can be freely downloaded from our web page (www.emotivecloud.net). Additionally, you can also contact with the authors.

Sample Service Deployed on top of the OSS Cloud

One of the main advantages of EMOTIVE Cloud is its support for dynamically creating VMs. Other approaches are based in instantiating previously created disk images and configuring some values using contextualization. Our middleware also

supports this typical approach where the provider just instantiates previously created images. Nevertheless, our approach also allows creating and configuring VMs on demand. Using this approach, the user selects the features of the VM and the packages he needs.

For creating a new Virtual Machine, the following steps are required on each host: downloading the guest operating system in packaged form (a Debian Lenny through debootstrap for this prototype), creating an image with this base system installed, copying extra software needed by the client in an image that will be automatically mounted in the VM, creating home directories and swap space, setting up the whole environment, packing it in an image, and starting the VM. Once the VM has completely started, the guest operating system is booted. After this, the additional software needed by the client needs to be instantiated (if applicable). These phases can be clearly appreciated in the next summary.

Phases for Creating a Virtual Machine

Virtual Machine creation procedure:

- Configuration and creation.
  - Obtain base system
    - Creating disk spaces
    - Configuring environment
- Cache archives and disk images.
  - Reduces network and disk bottleneck
    - Creation time from 40 to 10 seconds
- Starting virtual machine.
  - Booting operating system

From this description, one can derive that this process can have two bottlenecks: the network (for downloading the guest system; around 100MB of data) and the disk (for copying extra software needed by the client and creating all the needed images, namely base system, software, home, and swap; nearly 1.6GB of data).

The network bottleneck has been solved using a caching system per host that creates a default image of the guest system with no settings when it is downloaded for the first time. Then, this image is copied for each new VM created in that host. This almost eliminates the downloading time (base system is only downloaded once per host and can be reused for each new VM in that host), but contributes to the disk bottleneck. The disk bottleneck has been solved by adding a second caching system per host that periodically copies the default base system image and the images with the most commonly used software to a cache space. When a new VM is created, EMOTIVE Cloud just needs to move these images (just an i-node change) to the final location. Using both caching systems, the complete creation of a VM has been reduced to an average time of 7 seconds.

More details about the VM creation times are shown in (Goiri, 2009). Additionally, our proposal includes a data repository that allows storing the VM images used by each customer. These images can be later reused for creating new VMs.

In addition to dynamic creation, the user can also use a VM and store the VM image for later usage. In this way, the user can use a VM customized by him. To access the VM and work with it, the user can directly use the machine by connecting to the machine IP through simple SSH. The user can obtain the IP address from the Compute object returned by the *getEnvironment (Env-ID)* function of the EMOTIVE API. Alternatively, the user can also use the EMOTIVE API described before to run jobs on top of the VMs. Once the user has finished using the VM and wants to store the image, it only has to ask EMOTIVE to destroy the VM but specifying he wants to keep its state.

LESSONS LEARNED: MAIN SHORTCOMINGS AND FUTURE DIRECTIONS

EMOTIVE provides facilities for on-demand creation and life cycle management of virtual machines and for supporting resource management in Cloud environments. EMOTIVE is based on well-founded technologies (including support for different hypervisors) and has demonstrated to be useful to build Cloud Computing virtualized infrastructures. Actually, EMOTIVE simplifies the use of Cloud Computing both for clients and providers. The former are offered a simple interface based on de-facto standards (such as OCCI, OVF, and JSDL) to manage VMs and execute applications, where the latter are offered with basic management capabilities that they can use to build complex middleware services to manage their infrastructure in an autonomic way. Successful use cases have demonstrated that these capabilities allow achieving eco-efficient computing and other new challenges (Manageability and Self -*, Federation, Interoperability, Virtualization, Elasticity and Adaptability).

EMOTIVE can be easily extended due to its decentralized and modular architecture, being also highly interoperable because it supports the most popular standards. EMOTIVE supports good functionality with respect with its commercial counterparts, although these probably include additional capabilities. As commented before, this occurs due to the main goal of EMOTIVE, which is being used to support research at UPC and BSC. Due to the same reason, EMOTIVE support and documentation is not as exhaustive as their counterparts.

Future developments in EMOTIVE highly depend on the research projects of people using it. According to this, we are currently adding data ownership, user accounting, and security support to EMOTIVE. In addition, we also plan to evolve current functionalities depending on technologies and standards evolution, for instance improving OVF and OCCI support, VLAN/VPN management, and support for other virtualization hypervisors.

ACKNOWLEDGMENT

This work is supported by the Ministry of Science and Technology of Spain and the European Union under contract TIN2007-60625 (FEDER funds), the Ministry of Industry of Spain under contract TSI-020301-2009-30 (Avanza2 NUBA project), and Generalitat de Catalunya under contract 2009-SGR-980.

REFERENCES

Alonso, J. (2011). *Autonomic high availability and resource usage optimization: Proactive software rejuvenation solution for web environments on virtualized platforms*. PhD Thesis. Barcelona, Spain: Polytechnic University of Catalonia.

Amazon. (2006). *Elastic compute cloud (EC2)*. Retrieved March 22, 2011 from http://aws.amazon.com/ec2/.

API Libvirt Virtualization. (2005). *Webpage*. Retrieved March 22, 2011 from http://libvirt.org.

BREIN. (2006-2009). *FP6-IST-2005-2.5.4 European project*. Retrieved March 22, 2011 from http://www.eu-brein.com.

de Nadal, D., & Becerra, Y. (2010). *Support for managing dynamically Hadoop clusters*. Master Thesis. Barcelona, Spain: Polytechnic University of Catalonia.

DMTF. (2010). *Open virtualization format (OVF) specification: Version 1.1.0.* Retrieved March 22, 2011 from http://www.dmtf.org/standards/published_documents/DSP0243_1.1.0.pdf.

Ejarque, J., de Palol, M., Goiri, I., Julià, F., Guitart, J., Torres, J., & Badia, R. (2008). Using semantic technologies for resource allocation in computing service providers. In *Proceeding of the 5th IEEE International Conference on Services Computing (SCC 2008)*, (pp. 583-587). Honolulu, HI: IEEE Press.

EMOTIVE Cloud. (2009). *Autonomic systems and ebusiness platforms research line*. Retrieved March 22, 2011 from http://www.emotivecloud.net.

Fitó, J. O., Goiri, I., & Guitart, J. (2010). SLA-driven elastic cloud hosting provider. In *Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP 2010)*, (pp. 111-118). Pisa, Italy: Euromicro Press.

Fontan, J., Vazquez, T., Gonzalez, L., Montero, R. S., & Llorente, I. M. (2008). *OpenNebula: The open source virtual machine manager for cluster computing*. Paper presented at the Open Source Grid and Cluster Software Conference. San Francisco, CA.

Goiri, I., Fitó, J. O., Julià, F., Nou, R., Berral, J. L., Guitart, J., & Torres, J. (2010a). Multifaceted resource management for dealing with heterogeneous workloads in virtualized data centers. In *Proceedings of the 11th ACM/IEEE International Conference on Grid Computing (Grid 2010),* (pp. 25-32). Brussels, Belgium: IEEE Press.

Goiri, I., Julià, F., Ejarque, J., de Palol, M., Badia, R., Guitart, J., & Torres, J. (2009). Introducing virtual execution environments for application lifecycle management and SLA-driven resource distribution within service providers. In *Proceedings of the 8th IEEE International Symposium on Network Computing and Applications (NCA 2009)*, (pp. 211-218). Cambridge, MA: IEEE Press.

Goiri, I., Julià, F., Fitó, J. O., Macías, M., & Guitart, J. (2010b). Resource-level QoS metric for CPU-based guarantees in cloud providers. *Lecture Notes in Computer Science*, *6296*, 34–47. doi:10.1007/978-3-642-15681-6_3

NUBA. (2009-2012). *MITyC TSI-020301-2009-30 Avanza2 project*. Retrieved March 22, 2011 from http://nuba.morfeo-project.org.

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youse, L., & Zagorodnov, D. (2009). The eucalyptus open-source cloud computing system. In *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2009)*, (pp. 124-131). Shanghai, China: IEEE Press.

OGF. (2008). *Job submission description language (JSDL) specification: Version 1.0.* Retrieved March 22, 2011 from http://www.gridforum.org/documents/GFD.136.pdf.

OGF. (2010). *Open cloud computing interface (OCCI) infrastructure: Version 1.* Retrieved March 22, 2011 from http://forge.ogf.org/sf/docman/do/downloadDocument/projects.occi-wg/docman.root.drafts.occi_specification/doc16162.

OPTIMIS. (2010-2013). *FP7-ICT-2009-5 European project*. Retrieved March 22, 2011 from http://www.optimis-project.eu.

SORMA. (2006-2009). *FP6-IST-2005-2.5.4 European project*. Retrieved March 22, 2011 from http://www.sorma-project.eu.

Tejedor, E., & Badia, R. (2008). COMP superscalar: Bringing GRID superscalar and GCM together. In *Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2008)*, (pp. 185-193). Lyon, France: IEEE Press.

VENUS-C. (2010-2012). *FP7-INFRASTRUCTURES-2010-2 European project*. Retrieved March 22, 2011 from http://www.venus-c.eu.