# EMOTIVE User's Guide

Iñigo Goiri (*igoiri AT ac.upc.edu*)

current revision by:
Josep L. Berral (*berral AT ac.upc.edu*)
Marc Gonzalez Mateo and
Alexandre Vaque (*alex.vaque AT bsc.es*)

March 5, 2010

# Contents

# 1 Introduction

Virtualization has open a wide new range of opportunities for managing computer resources since it allows isolating different tasks in a single node without any interference with others. Moreover, a virtual machine is easier to monitor and offers a powerful but simple interface for managing its allocated resources which brings new chances for resource management. In addition to this, machine maintenance acquires a new level thanks to virtualization since it allows managing machines as software making it a perfect environment for executing tasks and complex services.

Nevertheless, dealing with some virtualization capabilities, such as the creation, implies an effort for the user in order to take benefit from them. In order to avoid this problem, we are contributing the research community with the EMOTIVE (Elastic Management of Tasks in Virtualized Environments) middleware, which allows executing tasks and providing virtualized environments to the users without any extra effort in an efficient way. This is a virtualized environment manager which aims to provide virtual machines that fulfills with the user requirements in terms of software and system capabilities.

Figure 1 represents the EMOTIVE Cloud which is mainly composed by three different layers: the data infrastucture, the node management (VRMM), and the global Scheduler. The data infrastructure offers a distributed storage for supporting virtualization capabilities such as migration and checkpoint support. VRMM is in charge of creating and maintaining the whole virtual machine lifecycle. Finally, the scheduling layer is in charge of distributing tasks and VMs among the physical nodes.

## 1.1 Data infrastructure

The data infrastructure makes EMOTIVE able to efficiently support different virtualization features such as migration and checkpointing. It distributes the data among the cluster nodes. It uses NFS in order to make the data of every node available from the other nodes. Thanks to this technique, Virtual Machines can be moved between nodes without losing connection. This capability allows new approaches such as consolidating the global system or giving more resources to a given application if the node is not able to do this locally.

In addition, this data infrastructure allows each VM accessing data re-

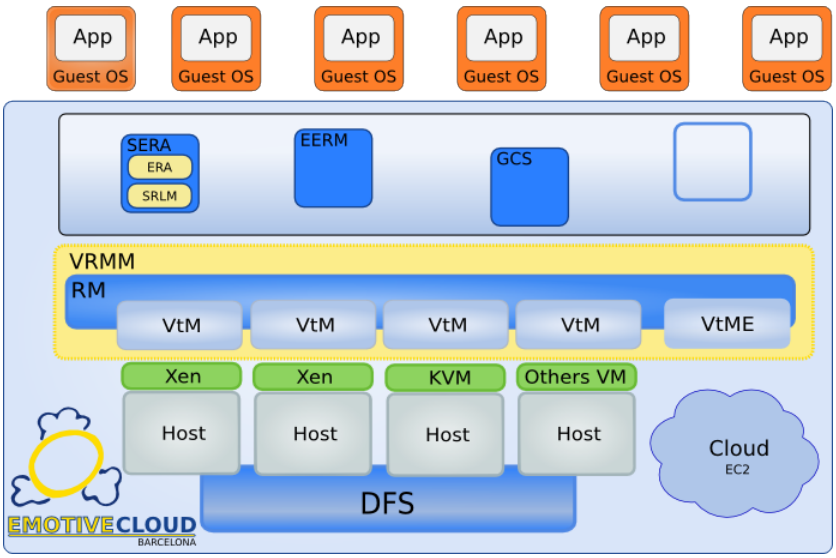


Figure 1: EMOTIVE Cloud architeture

quired by the user by using a shared repository also distributed among the nodes. It also allows storing data in the system in order to be reused later from other VMs.

## 1.2 Virtualized resource management

The first capability of the Virtualized Resource Management and Monitoring (VRMM) is the task of creating and maintaining virtual machine, which is done by the Virtualization Manager (VtM). This component is in charge of making this task as fast as possible in order to reduce the needed time to make the environment accessible to be used by the suer. In addition to the virtual machine life-cycle management, this layer takes proffit of the resource management capabilities that are offered by the virtualization. This allows the system managing the resources of each task in a dynamic way. VRMM also provides an interface for monitoring the resource usage of each VM called Resource Monitor (RM). This components allow and efficient usage of the providers usages as well as supporting different quality of service levels according to the agreement reached with the user (SLA).

With regard to the resource management infrastructure, this includes

virtual machine migration in an efficient well and a checkpoint management system. On the one hand, migration allows moving a VM between different nodes in order to offer it more resources or to consolidate the overall system. On the other hand, checkpoint mechanisms makes the system able to be fail tolerant since it allows the system recovering tasks that where running in a broken machine. These can be recovered from other nodes that have enough resources.

Another feature of this layer is the task execution support. This mechanism makes the data required by a task to be executed available inside the VM in the moment the virtualized environment is up and running. This allows the execution of tasks at the very beginning of the VM life and it finally allows the recovering of the output data generated by the applications. This user data management also includes the capability of storing user data and the whole virtual machine configuration between different user sessions. For instance, the user can work with a virtual machine and when his work has finished this VM data can be stored in the system. The execution of a a VM with all the stored data and with the same state of the previous could be retaken in the future by a different node of the provider.

### 1.3 Scheduling

A key issue of EMOTIVE is the global resource management, in other words, the global management of the different nodes of the system. For instance, choosing which node will execute each task or deciding the moment to migrate a VM in order to optimize the whole resources of the system. This is done by the scheduling layer which takes into account the overheads added in order to maintain and manage the virtualized environment. EMOTIVE has different scheduler implementations with different capabilities such as SERA, which supports semantic descriptions, or EERM, which takes into account economical parameters. Currently, an effort is being done in order to develop new schedulers that supports different features sucg as machine learning and a distributed decission system based on agents.

## 2 Installation

Minimum requirements for installing and use VtM 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 (next subsection explains the process)
- Java 1.5 or higher
- Libvirt 0.7.5 or higher

**Infrastructure installation** EMOTIVE requires the installation and setup of base software in the nodes that will be part of the execution nodes. This section presents the installation of this components.

DHCP+BIND

## 2.1 Dependences and build essential

First of all, install all the dependences you will need for building Xen, EMOTIVE, and setting up the Cloud:

```
# apt-get install build-essentials make gcc libc6-dev zlib1g-dev python
    python-dev python-twisted mercurial gawk libx11-dev gettext
    libncurses5-dev texinfo libssl-dev sun-java6-* ant tomcat6 maven2
    subversion debootstrap dhcp3-server bind9

apt-get install make gcc libc6-dev zlib1g-dev python python-dev python-twisted\
        bridge-utils iproute libcurl3 libcurl4-openssl-dev bzip2\
        module-init-tools transfig tgif libncurses5-dev patch libvncserver-dev\
        libsdl-dev libjpeg62-dev bcc bin86 gawk pciutils-dev mercurial\
        build-essential libc6-i386 libc6-dev-i386
```

## 2.2 Xen

Download latest Open Source Xen binary file for your platform and its source from http://xen.xensource.com/download. For instance, for the amd64 platform:

```
$ cd /aplic
$ wget http://bits.xensource.com/oss-xen/release/3.3.1/xen-3.3.1.tar.gz
```

Extract files to your machine:

```
$ tar xvfz xen-3.3.1.tgz
$ ln -s xen-3.3.1 xen-src
```

Install Xen and configuring the system can be done by using:

```
$ cd xen-src
$ make
# ./install.sh
```

Once the Xen is already installed is necessary to make the hypervisor executable in the system startup (XenDomains will store running domains in the shutdown momment and will restore them in the next boot):

```
# update-rc.d xend defaults 20 21
# update-rc.d xendomains defaults 21 20
```

Xen is already installed and will be executed in the machine startup. Now is needed to create the initrd image in order to load modules and update grub to execute the kernel with Xen support:

```
# depmod 2.6.18.8-xen
# mkinitramfs -o /boot/initrd.img-2.6.18-xen 2.6.18.8-xen
# update-grub
```

Often you will need to edit the /boot/grub/menu.lst file, and add an *initrd.img* line, if update-grub does not:

```
title Xen 3.3.1 / Ubuntu 8.10, kernel 2.6.18.8-xen
uuid 2330d19c-934a-48ff-8092-c80d4c4a9fb6
kernel /boot/xen-3.3.1.gz
module /boot/vmlinuz-2.6.18.8-xen root=/dev/sda1 ro console=tty0
module /boot/initrd.img-2.6.18-xen
quiet
```

In order to increment number of disk images usable by Xen, add 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. *Note: Xen 3.3.1 may have some problems when booting from a SATA boot-disk.*

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

```
# xm migration -l [domain] [host]
```

This will migrate specified domain to the other host without losing connectivity. Take into account relying architecture because guest OS doesn't support different architectures.

**XenMonitor** In order to manage Xen, XenMonitor will be used. It needs "libxenstat" libraries that are distributed in the Xen Source release. It must be compiled with "-fPIC" option in order to be integrated with XenMonitor. Thise code can be found in "tools/xenstat/libxenstat".

First of all, download the EMOTIVE source from the svn repository and place it on your deployment directory (https://emotivecloud.svn.sourceforge.net/svnroot/emotivecl
).

Edit the Makefile in "/aplic/xen-src/tools/xenstat/libxenstat/Makefile" and add "-fPIC" option to CFLAGS. Also change the paths of this Makefile and the build.xml found on "/aplic/emotive/XenMonitor", adjusting them to your system and architecture. Once done, change to the XenMonitor directory and compile it using ant:

```
$ cd /aplic/emotive/XenMonitor/
$ ant compile jar
```

In addition, libxenstat needs xenstore and libxc to be compiled. Using some new compilers it could fail due to a warning and is needed to remove -Werror flag from its Makefile.

```
$ cd /aplic/emotive/xen-src/tools/xenstore
$ make
$ cd /aplic/emotive/xen-src/tools/libxc
$ make
```

XenMonitor also uses Xen-API and it must be enabled to have access to the Xen core. Add this line to "/etc/xen/xend-config.sxp" and "/etc/xen/xend-config-xenapi.sxp". Xen will support receiving XML-RPC commands from the Xen-API java bindings.

```
(xen-api-server ((9363 'none' '' )(unix none)))
```

In order to finally install XenMonitor and make it available for VtM usage, 'add "libxenmonitor.so" to the java library path (for instance "$JAVA_HOME/jre/lib/amd64").

*Remember to export the JAVA_HOME environment variable to "/usr/lib/jvm/sun-java-1.6-*" or whatever your java installation path is*

```
$ cp XenMonitor.jar /aplic/
$ export  BREIN_COMMONS=/aplic/XenMonitor.jar
$ cp libxenmonitor.so $JAVA_HOME/jre/lib/amd64
```

## 2.3 Libvirt

**Libvirt Installation**

You can install libvirt using "*apt-get install libvirt-bin*" or download the last version of libvirt " *http://libvirt.org/*" and later install with the nexts commands:

```
configure --prefix=/usr/
make
make install
```

**Network bridge preparation**

First of all, install the bridge-utils using "apt-get install bride-utils" or similar.

And later you need to create a new net interface. This net interface allow do a bridge with the new virtual machines and the net. So we need to define a bridge in */etc/network/interfaces*.

You need to modificate the net interfaces file, so you have defined the new bridge interface (br0) and disabled the current (brein0 for example).

**Example of /etc/networks/interfaces**

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
\textbf{#auto eth0
}#iface eth0 inet manual

\textbf{#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

\textbf{#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

\textbf{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 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 enabled  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-ip6tables = 0
net.bridge.bridge-nf-call-iptables  = 0
net.bridge.bridge-nf-call-arptables = 0
```

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

## 2.4 KVM

To prepare this platform is necessary the next things:

- install KVM hipervisor: - Libvirt libraries

Installation and configuration of KVM virtualization system:

**KVM PREPARATION**

First of all, It is necessary to check if the CPU of the computer is compatible with KVM. KVM needs Intel or AMD virtualization instructions. You have to have a later model processor, with virtualization support, for KVM to work properly. This can be checked by examining /proc/cpuinfo.

If you have an Intel processor then do:

```
grep vmx /proc/cpuinfo
```

If you get results, your Intel processor is KVM ready.

```
flags! ! : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx lm constant_t
arch_perfmon pebs bts rep_good pni dtes64 monitor ds_cpl \textbf{vmx} est tm2 ssse
```

NOTE: if you have an AMD processor then do:

```
grep svm /proc/cpuinfo
```

Later, we need to verify that we have enabled the kernel with support for KVM. You need to do '*make menuconfig*' or similars and install KVM modules:

```
   [*] Virtualization  --->
<M>   Kernel-based Virtual Machine (KVM) support
<M>     KVM for Intel processors support
```

and install virtualization modules (KVM Support):

```
    Processor type and features  --->
    [*] Paravirtualized guest support  --->
       [*]   KVM paravirtualized clock (NEW)
       [*]   KVM Guest support (NEW)
```

To confirm the correct installation, you need to see kernel file configuration *.config*

```
cloud00:/lib/modules/2.6.28.1/build# cat .config | grep KVM
CONFIG_KVM_CLOCK=y
CONFIG_KVM_GUEST=y
CONFIG_HAVE_KVM=y
CONFIG_KVM=m
CONFIG_KVM_INTEL=m
```

You can see that CONFIG-KVM and CONFIG-KVM-INTEL has a ***m***. The ***m*** indicates that the kernel will be installed as a module, which will be loaded during system startup.

And later you can compile the new kernel to be used with KVM.

```
make
make modules
make modules_install
make install
cd /boot
mkinitramfs -o initrd.img-2.6.28.1 2.6.28.1
reboot
```

You need to update grub and reboot the machine. Later modificate /etc/modules file to add the *kvm* and *kvm-intel* kernel modules. You can reboot or simply load with *modprobe* command

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

loop
kvm
kvm_intel
```

**KVM INSTALLATION**

First of all, you need to install dependencies and prerequisites for KVM

```
apt-get update
apt-get install gcc libsdl1.2-dev zlib1g-dev libasound2-dev linux-kernel-headers p
```

To finish the installation, you need to do a typic installation. Descompress, compile, install and use 'lsmod' command to check if you will have KVM in the system well installed.

```
tar xzf kvm-kmod-2.6.28.1.tar.gz.tar.gz
cd kvm-kmod-2.6.28.1
./configure --prefix=/usr/
make
sudo make install
sudo /sbin/modprobe kvm-intel
lsmod | grep kvm
```

We recommend create a softlink executable linking the KVM to the library to avoid problems

```
ln -s /usr/bin/qemu-system-x86_64 /usr/bin/kvm
```

NOTE: Other opcion is use *apt-get install qemu kvm*

### 2.5 Data infrastructure

EMOTIVE can work with typical machine configurations but it will not take advantage of features such as efficient migration. In order to achieve it, the storage of every node must be shared between all the nodes using a remote file system, for instance NFS...

### 2.6 Network infrastructure

DHCP+BIND9

## 3 VtM installation

Once Xen is up and running, it is time to install Virtualization Manager. Some package are required in order to run VtM, debian utilities can be used in order to install them:

```
# apt-get install openjdk-6-jdk maven2 subversion
```

Once the code is downloaded (it can be obtained from different subversion locations), it has to be compiled using *maven*. It is composed by different nodes but in order to run VtM, it is only required to compile: VtM, VtMWS, and VtMWSClient. This setting can be set up in the main *pom.xml* file.

The compilation will generate a war file that contains all the required files and this has to be deployed in an application server (Apache Tomcat for instance).

Once Xen is up and running, it is time to install Virtualization Manager. You must compile the "Emotive" source code using maven.

## 3.1 Compiling and deploying WAR files

### 3.1.1 The Nodes (VtM, VtMWS, VtMWSClient)

It is composed by different nodes but in order to run VtM, it is only required to compile: VtM, VtMWS, and VtMWSClient. This setting can be set up in the main pom.xml file. Enter in the Emotive source directory and run "./compile"

```
$ cd /aplic/emotive
$ ./compile
```

The compilation will generate a war file that contains all the required files and this has to be deployed in an application server (Apache Tomcat for instance). Once Tomcat is installed in TOMCAT_HOME directory (for example "/var/lib/tomcat6"), you have to move the generated "war" files to the TOMCAT_HOME/webapps/, and if the Tomcat service is online, the files will automatically deployed inside the webapps directory.

Then, modify the TOMCAT_HOME/conf/server.xml, in order to register the new WebServices added to the webapps. And restart the Tomcat service.

```
server.xml:
<Context path="/VtM" docBase="VtM" debug=0 reloadable="true">
```

### 3.1.2 The Control (RM, Scheduler, GUI)

Compiling the Scheduler, RM and GUI will also generate "war" files. In the same way the VtM was deployed into the Tomcat Web Server, the RM, Scheduler and GUI must be deployed into the Control Node.

### 3.1.3 DomU utils

Install libxenstore applications (binaries) contained in "/aplic/xen-src/tools/xenstore/" at the "TOMCAT_HOME/webapps/WEB-INF/VtM/bin"

## 3.2 VtM setup

VtM can be configured by using the *vtm.properties* file which can be allocated in */etc/VtM* folder. This file contains some self-explained configuration

parameters such as the system architecure, the debian mirror used, or the amount of default home space of a VM.

The configuration files are:

- rm.properties: contains the location of the nodes (control)
- poolEnv.cfg: contains the default values for the creation of a node (node)
- vtm.properties: contains the values for a node (node)
- vtme.properties: contains the values for a node (node)
- hadoop.xml: contains the information for hadoop services (node)

## 3.3 First execution

VtM requires some basic files in order to create a Virtual Machine, this can be done automatically by VtM when it creates the first VM. Nnvertheless, it is a heavy operation that only needs to be performed once. It is achieved by using the VtM script called *rm.sh*, which is used in order to create and manage VMs. In order to setup the system and create the required base files, next order must be executed:

```
# sh rm.sh base
```

*Note: in some shells the "chroot" lines must be mended, not allowing to run a command at the same time than chroot-ing a directory.*

It will download Debian source files for your architecture and will create a basic system with no configuration.