

AEF Tutorial

Version 1.3 – August 2010

1. Introduction

This tutorial describes utilization of basic services from AEF to run emulated distributed systems experiments. In such experiments, computing nodes are represented by virtual machines running in a virtualization-enabled cluster of workstations. AEF—Automated Emulation Framework – allows automated deployment and configuration of emulated distributed systems and automated execution of applications and management of both application and environment.

For the purpose of this tutorial, several complex features from AEF are disabled. It reduces requirements of the physical system hosting the experiments, but limits AEF capabilities. This AEF version has the following features:

- Simple mapping of virtual machines to hosts, considering only virtual machine and hosts attributes. Network issues are not considered.
- Automatic deployment of virtual machines without external tools. Virtual machines are copied to hosts via ssh, and images are replicated and started inside each host.
- Automatic execution of experiments via ssh.

Features disabled in this version include:

- Network configuration and emulation of WANs (requires SNMP).
- Virtual DHCP (requires SNMP).
- Deployment via XSM/XSD (requires PXE and SystemImager).
- WBEM-based deployment (requires WBEM).
- Experiment monitoring, reconfiguration, and logging (requires WBEM).
- Automatic transferring of files to and from the experiment (requires WBEM).

Even though these features are disabled, all the classes that compose AEF, including the ones that enable these features, are present in the packet. They can be used if the following steps are taken: (i) the required services are configured in the cluster head node, cluster nodes, and virtual machines (for WBEM); (ii) configuration and support files (present under the *etc* directory in the AEF package) are installed in their proper places; and (iii) relevant AEF classes are recompiled, enabling services and replacing services in use by the desired ones.

1.1 Package contents

- `aef.properties`: basic configuration of AEF
- `cluster.xml`: example of a cluster configuration.
- `experiment.xml`: example of an experiment description.
- `logconfig.xml`: used by logging purposes.
- `runExperiment.sh`: script to trigger an emulation experiment.
- `tutorial.odt`: this tutorial.
- `virtual.xml`: example of an emulated distributed system.
- **br**: this is the base directory for the source files and class files.
- **doc**: base directory of AEF javadoc.

- **etc**: directory that contains configuration files for using SNMP and WBEM-based features.
- **lib**: this directory contains auxiliary jar files used by AEF.
- **utils**: this directory contains DTDs for XML files read by AEF and scripts used by AEF services.
- **utils/wbem**: files used by AEF when using WBEM features.

2. System Requirements

2.1. Head node

- Linux operating system.
- Sun Java 6.
- Direct access to cluster nodes via ssh without password.

2.2. Cluster nodes

- Xen 3.0.1 or higher.
- Direct access to head node via ssh without password.

2.3. Virtual Machine images

- Debian-based virtual machine images. For using VMs based in other operating systems, a new deployer has to be written, or advanced features such as SNMP must be enabled.

3. System installation and configuration

1. Extract the contents of the package in a directory from the head node.
`$ tar xzf aef.tgz`
2. A directory `aef` will be created. Access it.
`$ cd aef`
3. Edit the file `aef.properties`. The only value that must be edited for a simplified use of `aef` is the `hostname` property, in the first line of the file. Complete it with the name of the machine:

```
hostname = xsdserver.lad.pucrs.br
guest.image.location = /tmp
guest.overwrite.images = false
interface = eth1
enable.virtual.services = false
dhcp.port = 1000
vserver.ip = 192.168.6.1
log.memory = memoryLog.txt
```

log.cpu = processorLog.txt

4. Edit the file *cluster.xml*. This file contains characteristics of each node from the cluster:

4.1. The whole cluster must be defined in a *network* element, which describes the cluster network characteristics. Here, the cluster name, subnet mask and gateway (head node) address must be specified

```
<network name="cluster name" subnet_mask="netmask" gateway="gw">
```

```
</network>
```

4.2. Inside the *network* element, cluster nodes have to be described. For each node, a line like the following one has to be added. Furthermore the correct values of name (because DNS is not enabled, so it must contain node's IP address), memory (in MB), operating system, MAC address, and IP address have to be supplied. The attribute *operating system* is not used currently. Therefore, any value may be assigned to it. The attribute *power* is used relatively to other elements. Therefore, an arbitrary value may be assigned to one of the nodes and the relative power of other nodes comparing to the first one assigned may be used to other nodes.

```
<cpu name="IP" power="power" memory="memory"
operating_system="Linux" hw_address="MAC" network_address="IP"/>
```

4.3. A *network_link* element, describing the network in use, have to be added, and the bandwidth (in bits per second) and latency (in seconds) of such a network have to be defined. An example for a Fast Ethernet network is:

```
<network_link name="ethernet" bandwidth="100000000" latency="0.0001"/>
```

4.4. Finally, routes between nodes have to be defined. Routes are bidirectional. Furthermore, for clusters connected by switches, its only required that each *cpu* element appears at least once in the routes. The name in *src* and *dst* attributes must be the same names defined for the *cpu* element. Similarly, the name in the *route_element* must be the same name of a defined *network_link*.

```
<route src="source IP" dst="dest IP">
  <route_element name="ethernet"/>
</route>
```

A complete cluster description file for a 4-node fast Ethernet cluster is given in Appendix 1.

4. AEF utilization

AEF utilization consists of four steps, namely:

1. Definition of the emulated distributed system.
2. Definition of the applications to be executed in the emulated distributed system.
3. Creation of VM images.
4. Start of the experiment.

Each of these steps is described in the following sections.

4.1. Definition of the emulated distributed system

The emulated distributed system, to be built and configured by AEF, is described in a XML file. This file is similar to the file used to configure the cluster. However, some elements and attributes are interpreted differently in either case. The file name is assigned by the AEF user, and the file name is one of the inputs passed to AEF during its execution.

Once again, the basic element is the *network* element. Here, this element represents a LAN (site) that may be connected to other LANs via WAN connections. The difference between the same element in the cluster description is that here *gateway* refers to the machine in the LAN that has access to the WAN and therefore can communicate with other sites. So, an emulated environment containing three sites will have three *network* elements, each one containing its own set of machines.

Inside each virtual network, each *cpu* element represents a machine in the virtual distributed system. AEF creates one virtual machine for each *cpu* element. In the context of the emulated environment description, *memory* attribute in the *cpu* element means the amount of memory of the virtual machine emulating such a *cpu*. The other attributes have the same meaning as in the case of the cluster.

Any IP address may be assigned to each *cpu*, but it is important that this assigned is coherent: *cpus* belonging to the same network must have addresses belonging to the same network (considering both other IPs in the same network and the subnet mask). Similarly, different networks might have different IP network addresses.

The same considerations about defining the network routes are valid for the virtual environment: nodes in the same network have to appear only once in the description .

A complete example of a virtual distributed system containing is given in Appendix 2.

4.2. Definition of the applications to be executed in the emulated distributed system

Applications to be executed in the emulated distributed system created by AEF is described in a separate XML file. Entries of such a file are elements describing the application to be executed, the virtual machine where the application runs and application parameters. The file name is defined by AEF users, and it is passed as an argument during AEF execution, as described later in this tutorial.

Each application is an element of type *process*, and its attributes are the executable to be called and the host, as follows:

```
<process host="IP" start_function="executable full path">
</process>
```

Additionally, parameters are elements that are defined inside the *process*. Each parameter is defined separately, as follows:

```
<process host="IP" start_function="executable full path">  
  <argument value="param1"/>  
  <argument value="param2"/>  
</process>
```

All the executables that are supposed to be executed, together with configuration files and other files required for the correct execution of the application must be present in the virtual machine image deployed in the cluster.

Similarly, in the end of execution output files will be stored in the VM image that generated the file.

4.3. Creation of VM images

Emulated computing nodes are represented by virtual machines in AEF. Therefore, it is necessary that virtual machine images are supplied by AEF users. Two images are required: an operating system image and a swap image. The latter is a simple image that is used by the operating system for swap purposes. It does not require any special configuration. A swap image of 512MB may be created by the following steps:

1. \$ dd if=/dev/zero of=swap.img bs=1024k count=512
2. \$ mkswap swap.img

Regarding the operating system image, its requirements depend on the hardware from the cluster nodes. If the cluster nodes have hardware-enabled virtualization (e.g, Intel vt processors), these images may be cloned images of regular operating systems, or they can be created from regular boot CDs. Currently, AEF supports only Debian-based VMs. Virtual machines based in other operating system cannot have network parameters set during deploy time. In this case, SNMP-based network configuration must be used.

If hardware does not support virtualization, then a special Linux kernel must be used (paravirtualization). These images are available on Internet or they can be created manually or by virtualization tools that are part of current Linux distributions.

In either case, all the services, configurations, files, and executables required by the applications executed in the experiments have to be available in the images. Furthermore, services have to be configured to be initialized during boot time.

Network configuration is performed by AEF. Therefore, any network configuration present in the images are overwritten by AEF during deployment stage. Any other configuration and files are kept untouched by AEF. During generation of the operating system image, it is important to make it large enough to accommodate output files from the applications, otherwise experiment may be compromised due to lack of storage space.

4.4. Start of the experiment

To start emulation experiments, the script runExperiment.sh is available along with AEF. This script receives as arguments the emulated environment description, experiment description, virtual machine image and swap image:

```
$ ./runExperiment.sh environment.xml experiment.xml vm.img swap.img
```

If all the files are stored in the *aef* directory, experiment must be executed without problems. In case of problems, they are displayed in terminal. Furthermore, the file *aef.log* contains additional information from the execution.

Once all the applications are started in the virtual machines, AEF finishes execution and the shell is released.

5. Running interactive experiments

AEF supports interactive execution of experiments, via a shell-based user interface. To run AEF shell, execute in the AEF directory the following command:

```
$ ./runCommandLine.sh
```

All the available options will be presented. At any time, the options may be presented again by typing 'h',

For submitting a cluster description file, 'c' must be typed. Then, the path to the file will be asked. Similarly, virtual system is passed by the command 'e' followed by the path to the file. Experiment runs when 'b' is typed. In this case, the system also asks for the path to the VM files.

In the interactive mode, experiments are triggered manually by the 'r' command. Next, the VM where the command run (among the active VMs – see below for details) and the command have to be typed.

In the interactive mode, VMs are not readily available for use after deployment. Instead, they are in an 'inactive' state where they cannot execute commands. To active a machine, type 'a' then chose a machine from the list. You may also specify a delay for activation. Machines are disabled with the command 'd' followed by the selection of the machine among those that are active. AEF shell quits with command 'q'.

References

1. Rodrigo N. Calheiros, Rajkumar Buyya, César A. F. De Rose. *Building an automated and self-configurable emulation testbed for grid applications*. Software: Practice & Experience, v.40, i.5, p.405-429, 2010.
2. Rodrigo N. Calheiros, Everton Alexandre, Andriele B. do Carmo, César A. F. De Rose, Rajkumar Buyya. *Towards Self-Managed Adaptive Emulation of Grid Environments*. IEEE Symposium on Computers and Communications (ISCC 2009), 2009, Sousse, Tunisia.
3. Rodrigo N. Calheiros, Mauro Storch, Everton Alexandre, César A. F. De Rose, Marcus Breda. *Applying Virtualization and System Management in a Cluster to Implement an Automated Emulation Testbed for Grid Applications*. 20th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2008), 2008, Campo Grande, Brazil.

Appendix 1. Cluster description example

```
<?xml version='1.0'?>
<!DOCTYPE platform_description SYSTEM "utils/description.dtd">
<platform_description version="1.0">

  <network name="cluster_xen" subnet_mask="255.255.255.0" gateway="192.168.5.254">
    <cpu name="xen01" power="100000" memory="2232" operating_system="Linux"
      hw_address="00:12:79:64:89:09" network_address="192.168.5.101"/>
    <cpu name="xen02" power="100000" memory="2232" operating_system="Linux"
      hw_address="00:12:79:64:8A:46" network_address="192.168.5.102"/>
    <cpu name="xen03" power="100000" memory="2232" operating_system="Linux"
      hw_address="00:12:79:64:89:17" network_address="192.168.5.103"/>
    <cpu name="xen04" power="100000" memory="2232" operating_system="Linux"
      hw_address="00:12:79:64:8A:15" network_address="192.168.5.104"/>
  </network>

  <network_link name="ethernet" bandwidth="1000000000" latency="0.0001"/>

  <route src="xen01" dst="xen02">
    <route_element name="ethernet"/>
  </route>
  <route src="xen01" dst="xen03">
    <route_element name="ethernet"/>
  </route>
  <route src="xen01" dst="xen04">
    <route_element name="ethernet"/>
  </route>

</platform_description>
```


Appendix 2. Emulated system example

```
<?xml version='1.0'?>
<!DOCTYPE platform_description SYSTEM "utils/description.dtd">
<platform_description version="1.0">

  <network name="site1" subnet_mask="255.255.255.0" gateway="192.168.8.1">
    <cpu name="192.168.8.1" power="10" memory="1024" operating_system="Linux"
      network_address="192.168.8.1"/>
    <cpu name="192.168.8.2" power="10" memory="1024" operating_system="Linux"
      network_address="192.168.8.2"/>
    <cpu name="192.168.8.3" power="10" memory="1024" operating_system="Linux"
      network_address="192.168.8.3"/>
    <cpu name="192.168.8.4" power="10" memory="1024" operating_system="Linux"
      network_address="192.168.8.4"/>
    <cpu name="192.168.8.5" power="10" memory="1024" operating_system="Linux"
      network_address="192.168.8.5"/>
    <cpu name="192.168.8.6" power="10" memory="1024" operating_system="Linux"
      network_address="192.168.8.6"/>
  </network>

  <network_link name="ethernet" bandwidth="100000000" latency="0.0001"/>

  <route src="192.168.8.2" dst="192.168.8.1">
    <route_element name="ethernet"/>
  </route>
  <route src="192.168.8.2" dst="192.168.8.3">
    <route_element name="ethernet"/>
  </route>
  <route src="192.168.8.2" dst="192.168.8.4">
    <route_element name="ethernet"/>
  </route>
  <route src="192.168.8.2" dst="192.168.8.5">
    <route_element name="ethernet"/>
  </route>
  <route src="192.168.8.2" dst="192.168.8.6">
    <route_element name="ethernet"/>
  </route>

</platform_description>
```

Appendix 3. Experiment description example

```
<?xml version='1.0'?>
<!DOCTYPE platform_description SYSTEM "utils/description.dtd">
<platform_description version="1.0">

  <process host="192.168.8.1" start_function="/home/ourgrid/ourgrid/peer">
    <argument value="start"/>
  </process>

  <process host="192.168.8.2" start_function="/home/ourgrid/ourgrid/mygrid">
    <argument value="addjob"/>
    <argument value="/home/ourgrid/ourgrid/simpleJob.jdf"/>
  </process>

  <process host="192.168.8.3" start_function="/home/ourgrid/ourgrid/useragent">
    <argument value="start"/>
  </process>

  <process host="192.168.8.4" start_function="/home/ourgrid/ourgrid/useragent">
    <argument value="start"/>
  </process>

  <process host="192.168.8.5" start_function="/home/ourgrid/ourgrid/useragent">
    <argument value="start"/>
  </process>

  <process host="192.168.8.6" start_function="/home/ourgrid/ourgrid/useragent">
    <argument value="start"/>
  </process>

</platform_description>
```