An Assessment of Eucalyptus Version 1.4

Tingxi Tan and Cameron Kiddle

Grid Research Centre, University of Calgary, Canada
{txtan,kiddlec}@cpsc.ucalgary.ca

May 4, 2009

1 Introduction

Cloud Computing is the emergent technology that promises on-demand, dynamic and easily accessible computing power. The "pay-as-you-use" scheme is attractive for small to medium sized businesses as these organizations are less inclined to purchase large amounts of physical machines to satisfy their immediate computing needs. Various cloud services are already available on the market. Many of them implement some form of dynamic provisioning of computing resources through the use of Virtual Machine (VM) technologies like Xen [13], VMWare [28] or KVM [16]. Among them, the Amazon Elastic Cloud (EC2) [3] can be considered the most popular and mature solution.

Eucalyptus [20], a cloud enabling infrastructure is the result of a research project from the University of California, Santa Barbara. Eucalyptus stands for "Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems". It aims to provide a simple to set up cloud solution for the research and development of cloud driven applications. By combining common web-service, Linux tools and the Xen Virtual Machine Hypervisor, Eucalyptus successfully implemented partial functionality of the popular Amazon EC2. As a consequence of recreating a "free" version of EC2, this open source project has attracted much attention and it is scheduled to be included into Ubuntu 9.10 (code name Karmic Koala) [15], the to-be-release version of a popular Linux distribution.

This document records a recent effort to evaluate Eucalyptus as a viable open source solution to cloud computing. The evaluation focuses on the design, setup, usability and performance of Eucalyptus. In Section 2, we discuss the general design goals and infrastructure layout of Eucalyptus. Section 3 documents the process of setting up a Eucalyptus environment. Section 4 covers the usage and general impressions of Eucalyptus's functionalities. In Section 5, we developed a demonstrator to illustrate the potential real-world usage of Eucalyptus v1.4. Finally in Section 7 and 8, we provide some related work and a conclusion to this document.

2 Design

The initial design goal of Eucalyptus was to create an open source cloud computing platform where individuals can deploy their own cloud computing infrastructure to develop or research cloud-driven applications.

The developers of Eucalyptus had chosen the Amazon EC2 interface as an interface to a Eucalyptus cluster. This is to allow users already using EC2 easy adaptation to Eucalyptus. They have successfully emulated many EC2 functionalities like security groups, elastic IPs, zones and even the Amazon Simple Storage Service (S3). This emulation effort has effectively created a "free" version of EC2 and as mentioned before, has gained much attention among the open source community. The increase in popularity has widened the target audience for Eucalyptus; apart from computer scientist, IT professionals are also exploring Eucalyptus as an all-in-one platform solution to build their own cloud.

Due to the very successful attempt at emulating Amazon EC2, the Eucalyptus design goals have also deviated from their original plans. Now, most of Eucalyptus developments focus on implementing and supporting EC2 functionalities rather than creating a general cloud computing platform.

2.1 Infrastructure

The Eucalyptus infrastructure (Figure 1) consists of four main components, the **Cloud Controller** (CLC), the **Cluster Controller** (CC) the **Node Controller** (NC) and a storage service called **Walrus**. These components are implemented as stand-alone web services. They leverage WS-Security policy for secure communication via a well-defined WSDL API. They interact via SOAP and HTTP to dynamically provision Virtual Machines (VMs) or manipulate VM images. Eucalyptus currently supports only Xen VMs but plans to support other popular virtualization platforms like KVM/QEMU and VMWare.

Each physical machine in the cluster capable of hosting VMs is installed with a NC. The NC starts and stops VM instances and monitors the health of those instances on that particular machine. The CLC is the interface for external entities to communicate with the Eucalyptus environment. It receives requests to start or stop VM instances and forwards these requests to the CC. The CC maintains resource information about the entire Eucalyptus cluster by periodically polling the NCs. Based on the global resource availability, the CC then decides which NC in the cluster should be contacted to start or stop a VM.

A typical set up of Eucalyptus on a single cluster consists of a head machine dedicated as the CLC and CC, and multiple slave machines as NCs. The slave machines must support the starting of Xen VMs, i.e., a Xen Hypervisor must be installed on each NC. In theory the CLC can communicate with multiple Eucalyptus clusters via multiple CCs, however this aspect of Eucalyptus is not currently implemented. The developers expect support for additional CCs to be in the next release of Eucalyptus.

Walrus is a storage service similar to that of Amazon S3. The primary use of Walrus is to store VM images called **Eucalyptus Machine Images** (EMIs). The EMI format is used by Eucalyptus to encode images which will in turn be used to start Xen VM instances. EMI is similar in concept to the **Amazon Machine Image** (AMI) format that is used to start EC2 instances. Walrus can be accessed using `Curl`, a popular command-line tool for interacting with HTTP services. Other than storing EMIs, Eucalyptus users can also use Walrus to store raw data the same way an EC2 user employs S3. Currently, Walrus must be installed on the same physical machine as the CLC.

2.2 Usage

Access to start and stop VM instances in Eucalyptus is controlled through the `Amazon EC2 API`. This is a set of command-line tools originally used for interacting with EC2. The Eucalyptus backend emulate the SOAP and Query interface of the EC2 API to provide a common method for users to access both EC2 and Eucalyptus through the same tool set. Similarly, the `Amazon AMI API` is used by Eucalyptus to provide VM image creation functionality. Further discussions on these tools are provided in Section 3.4, 4.1 and 4.2.

A web portal at the CLC controls user creation and keys generation. From this portal, a Eucalyptus user obtains their X.509 certificate and other secret keys to authenticate with the CLC and Walrus respectively.

3 Setup

In this section, we document the installation and configuration of a Eucalyptus cluster for our evaluation. This environment is installed in the HP Labs Data Centre at the University of Calgary.

3.1 Physical Environment

Three physical blades are used in the Eucalyptus installation (Figure 2). Each blade has 2 x 2.4 GHz AMD Opteron 2216 HE processors, 8 GB of physical memory, 2 physical 1 Gb/s NICs, a 73 GB SCSI local disk and is running Linux SUSE 10.3. One machine (`b02b10`) is configured as the CLC/CC. It is assigned two

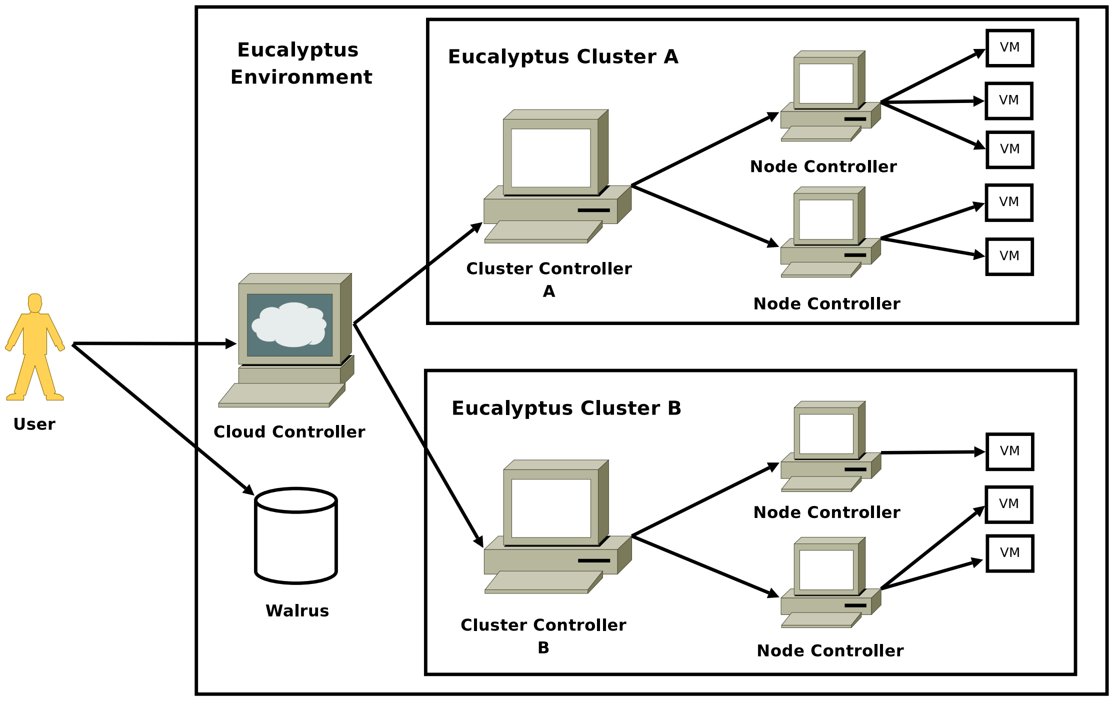


Figure 1: Eucalyptus Environment

IP addresses, a public one for communicating with the outside world and an internal IP for communication within the Eucalyptus cluster.

The remaining two machines (`b02b09` and `b02b11`) are configured as NCs. They are responsible for hosting Xen VMs instances and have Xen 3.2.1 installed. The nodes communicate with the CLC/CC via internal IP addresses.

3.2 Installation

The Eucalyptus documentation [9] is well laid out and describes clearly the steps involved to set up the environment. Installation can be performed via a Rocks roll (if a Rocks enabled cluster is available), pre-packaged RPMs or from source. Installing from source was the method chosen for this evaluation. A few software dependencies had to be fulfilled before the Eucalyptus source was compiled from source. These dependencies are Apache, Ant, Axis2, Axis2/C, Rampart/C, Libvirt and Java Development Kit (JDK). Once the dependencies were satisfied, the Eucalyptus source was compiled on the head node (b02b10) and distributed via `rsync` to the the virtualization nodes (b02b09 and b02b11). A configuration file called `eucalyptus.conf` on each machine determines which Eucalyptus services to start. On b02b10, the CLC, CC and Walrus web services will start whereas on b02b09 and b02b11, the NC web service will start.

3.3 Network

Eucalyptus supports three types of network setup, **SYSTEM**, **STATIC** and **MANAGED**. In SYSTEM mode, each VM is assigned a random MAC address and obtains a dynamic IP address via a DHCP server. This mode is the easiest and quickest way for a VM instance to obtain network. However it is limited as there is no way for an administrator to assign a specific IP to a VM.

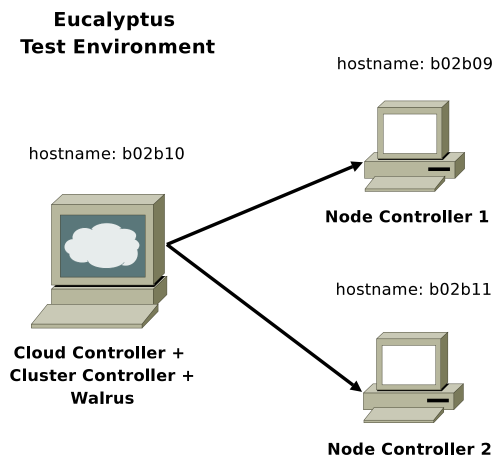


Figure 2: Deployment of Eucalyptus Environment

In a cluster environment where greater control over IP allocation is desired, STATIC mode is more suitable. In STATIC mode, an administrator defines a set of MAC to IP address pairs in the Eucalyptus configuration file on the CLC. As a VM starts, Eucalyptus assigns an unused MAC and IP address pair to it, ensuring that each VM in the system will obtain a unique IP address based on the range defined by the administrator. This method provides more flexibility than STATIC mode but manifests some major bugs which we will be discussing in Section 4.5.1.

MANAGED mode has the most function-rich networking setup. In MANAGED mode, each VM obtains an IP from a DHCP server similar to STATIC mode. A Eucalyptus user can now define 'named networks' where they wish to start their VM instances in. These private networks created using a VLAN allow users to specify ingress rules that can be applied to all VMs in that network. For example, a user can block all ping(ICMP) traffic from reaching a certain set of VMs he or she owns. This is similar in functionality to Amazon EC2's security groups. The administrator can also choose to define a list of public IP addresses that users can request for their VMs. These public IPs can be attached or detached from VMs as the user requires, allowing IPs to float between VM instances. This mimics the elastic IP functionality of EC2. As a consequence of the particular VLAN set up in MANAGED mode, the VM network is isolated from the physical machine network. This implies that a VM cannot communicate directly with other physical machines in the Eucalyptus cluster, except to the CLC/CC.

All three network setups were tested in our Eucalyptus installation. We settled for STATIC mode as the final setup. The following are the configuration parameters. A total of 9 MAC/IP address pairs are defined. This implies that we can currently start up to a maximum of 9 VMs in the Eucalyptus cluster.

```
VNET_MODE="STATIC"
VNET_SUBNET="192.168.0.0"
VNET_NETMASK="255.255.0.0"
VNET_BROADCAST="192.168.255.255"
VNET_ROUTER="192.168.1.1"
VNET_DNS="192.168.101.1"
VNET_MACMAP="
0a:16:3f:2e:02:bf=192.168.151.192 0a:16:3f:2e:02:c0=192.168.151.193
```

```
0a:16:3f:2e:02:c1=192.168.151.194 0a:16:3f:2e:02:c2=192.168.151.195
0a:16:3f:2e:02:c3=192.168.151.196 0a:16:3f:2e:02:c4=192.168.151.197
0a:16:3f:2e:02:c5=192.168.151.198 0a:16:3f:2e:02:c6=192.168.151.199
0a:16:3f:2e:02:c7=192.168.151.200"
```

3.4 Admin and User Tools

The Eucalyptus installation configures a secure HTTP portal on the CLC on port 8443. This web interface is used mainly to manage users and configure the cluster. We will discuss more on the Eucalyptus web portal in Sections 4.2 and 4.3.

Apart from the web portal, users interact with Eucalyptus services to start/stop VM instances or create EMIs through third party software like the Amazon EC2 API and Amazon EC2 AMI tools. These java command-line tools can be downloaded from the Amazon EC2 site and installed on any machine that a user will be accessing the Eucalyptus services from.

To set up the tools, administrators or regular users need to download their authentication credentials from the Eucalyptus web portal and set up appropriate paths and variables. The following .eucarc file defines these variables. Each user and administrator of the Eucalyptus cluster will obtain a unique EC2_CERT, EC2_PRIVATE_KEY, EC2_ACCESS_KEY and EC2_SECRET_KEY. This file should be sourced before using Eucalyptus.

```
EUCA_KEY_DIR=$(dirname $(readlink -f ${BASH_SOURCE}))
export EC2_HOME=~/eucalyptus/ec2-api-tools-1.3-30349
export EC2_AMITOOL_HOME=~/eucalyptus/ec2-ami-tools-1.3-26357
export S3_URL=http://10.0.0.1:8773/services/Walrus
export EC2_URL=http://10.0.0.1:8773/services/Eucalyptus
export EC2_PRIVATE_KEY=${EUCA_KEY_DIR}/euca2-txtan-0b539a01-pk.pem
export EC2_CERT=${EUCA_KEY_DIR}/euca2-txtan-0b539a01-cert.pem
export EUCALYPTUS_CERT=${EUCA_KEY_DIR}/cloud-cert.pem
export EC2_ACCESS_KEY='DAY4oep0TgYrAr8LERF-90'
export EC2_SECRET_KEY='RmgY_MAVhQG8aDdIBRm4uLE707ruYG8KwXTZIg'
alias ec2-delete-bundle="ec2-delete-bundle -a ${EC2_ACCESS_KEY} -s
${EC2_SECRET_KEY} --url ${S3_URL}"
alias ec2-bundle-image="ec2-bundle-image --cert ${EC2_CERT} --privatekey
${EC2_PRIVATE_KEY} --user 000112525872 --ec2cert ${EUCALYPTUS_CERT}"
alias ec2-upload-bundle="ec2-upload-bundle -a ${EC2_ACCESS_KEY} -s
${EC2_SECRET_KEY} --url ${S3_URL} --ec2cert ${EUCALYPTUS_CERT}"
```

To facilitate easy access for interaction with Walrus, Amazon S3 Curl can be downloaded. This perl-based command-line tool computes the appropriate signature to authenticate with S3 before invoking curl calls. The only change required to make S3-curl work with Walrus is to define the HTTP endpoint in S3-curl as the Eucalyptus CLC instead of Amazon S3. In our installation, this endpoint will be b02b10. All other authentication keys should already be defined in the user's .eucarc file as shown above.

4 Usability

This section covers the general usability of Eucalyptus from the point of view of both cluster administrator and regular users. The functionalities are classified into three categories, **Image and Storage Management**, **Cluster and VM Management** and **User Management**. We will also comment on the security features, community support and bugs detected in the software.

4.1 Image and Storage Management

Walrus is a storage server employed by Eucalyptus to stored EMIs. It emulates the functionality and interface of Amazon S3, using a single level storage hierarchy, putting "objects" into "buckets". Other than functioning as an image repository, Eucalyptus users can also use Walrus as a generic storage medium to store raw data. However, unlike S3, which supports access through API programming for multiple languages, Walrus only allows access via curl and S3-curl. Currently, Walrus must be installed on the CLC. By default, the underlying storage device for Walrus will be the local hard disk. However due to storage requirements, it is more practical to employ a network device exported from a large storage pool, for example a SAN. Technically, this is feasible as we can simply point the Walrus directory to a network exported device. However, we did not test this particular set up; we used the local hard drive on the CLC as the storage backend for Walrus. As such, we do not guarantee that Walrus will function correctly with devices other than a local disk. A distributed Walrus service is also being considered by the developers for a future version of Eucalyptus.

The process to create and upload an image into the EMI format is similar to image creation for the AMI format; using the Amazon EC2 AMI tools to bundle kernel and ramdisk images with operating system (OS) images. Support for converting an AMI to an EMI is planned for the next release of Eucalyptus. Like an AMI, before bundling an OS image into an EMI, the image itself needs to conform to 2 specific requirements:

1. The image must be Xen compatible. This implies that the user should first test the image by ensuring that it can be manually started using Xen.

2. Eucalyptus will mount the image as partitions, whole disk images do not work. A specific partition table is also expected. If a different partition layout is desired, the Eucalyptus source must be changed to reflect the new partition table.

Once these requirements are satisfied, a user first bundles an image (optionally with a kernel and ramdisk) into a EMI object using `ec2-bundle-image`. Then the images is uploaded to Walrus via the `ec2-upload-image` command. Once the uploading is done, the user will register the image into Eucalyptus with `ec2-register`. Both administrators and regular users can create and upload images. By default, the images uploaded by any user are publicly viewable and startable with `ec2-describe-images` and `ec2-run-instances`. Often a user may choose to restrict the starting of his or her image by other users. This can be achieve by setting the access control on that particular EMI using `ec2-modify-image-attribute`. An administrator can also toggle image visibility through the Eucalyptus web portal (Section 4.3).

All EMIs uploaded to Walrus are considered "base" VM images. This means that these images are never directly used to instantiate VMs. Before a VM is started (this starting process will be elaborated in Section 4.2), the CLC first determines if the destination NC has a local cache copy of the image requested for the particular VM instance. If the image was not previously cached, the CLC will initiate a process to extract the base image from Walrus and push it to the NC, where it will be cached. Once the NC verifies the integrity of the cache copy, it will make another copy from the cache before starting a VM locally with the copy. From our evaluation, a minimum of 8 GB of temporary files were generated on the CLC when distributing a single 4 GB image to a NC with a cold cache. A NC with a hot cache uses 4 GB of disk space for each additional instance started from the cached image. We will quantify the time taking to start VM instances in the next section.

To summarize, in the worst case, an instance using a 4 GB image require an initial investment of at least 20 GB of disk space globally. This enormous space requirement prohibited our Eucalyptus test environment from storing a large repository of VM images; we constantly run out of disk space when starting multiple instances simultaneously. Finally, we deviated from the initial Eucalyptus design by using an NFS exported directory from a large storage pool for a shared cached across all the NCs. This method will no doubt create delays in image I/O when contention among the NCs to access the shared cache is high. However it is the solution we employed to overcome the space requirement in our test environment.

The Eucalyptus developers will need to resolve these image storage and distribution issues before large-scale deployment of Eucalyptus can match those offered by Amazon EC2.

4.2 Cluster and VM Management

The Eucalyptus web portal allows an administrator limited control over cluster and VM configuration. This includes setting the network location for the Walrus service, the local path to Walrus on the CLC and the network location of the zone in which VMs will start.

The **Zone** system in Eucalyptus is a similar concept borrowed from Amazon EC2. While in EC2, a zone means a specific geographical region, in Eucalyptus, it denotes a particular cluster of machines controlled under a CC. As mentioned in Section 2.1, the Eucalyptus design incorporate multiple CCs reporting to a CLC. Therefore, in theory, multiple zones can be enabled in a Eucalyptus environment. However this particular feature is not currently implemented. This limitation is reflected in the web portal where an administrator can only specify the network location of a single cluster/zone.

From the web portal, the administrator can also specify pre-defined resource description for VM types. For example, VM type `m1.small` consumes 1 CPU and 512 MB of RAM while `c1.xlarge` consumes 4 CPU and 4 GB of RAM. A total of 5 VM types can be defined. As VMs start in the cluster, the administrator can monitor the total resources consumed by executing the `ec2-describe-availability-zones verbose` command. This command will output the number of free "slices" left in the zone to start instances of each type. However, it does not tell the administrator which VM is running on which NC. The lack of detail in monitoring information on VM deployment can potentially hinder an administrator from effectively managing a large Eucalyptus cluster.

From a user's perspective, VM control as well as monitoring resource availability are limited to command-line tools. Before a user starts an instance, he or she will execute the `ec2-describe-availability-zones` command (the `verbose` option is not available to regular users). If the zone is online, the command will output the name of the zone, otherwise it will return an error message denoting the unavailability of any resources. Unlike an administrator, the user does not know exactly how many more instances he or she can start in a zone, only if the zone is online or not.

Next, the user will execute `ec2-describe-images` to find out the EMIs visible to him or her. To start a VM, the user will specify the type of VM, the number of instances and the EMI ID from which the instances will start through the `ec2-start-instances` command. At this point, the CC will determine if there are enough resources in the zone to start the instances. If there are, it will select a NC that can accommodate the resource requests and perform the image distribution process as mentioned in Section 4.1. The selection of the candidate NC is based on a greedy or round robin deployment policy. This policy can be specified by the administrator in the Eucalyptus configuration file. If no suitable NCs are found, the `ec2-start-instances` command will return a failed message to the user denoting an unavailability of resources. In our preliminary measurements specific to the setup in our test environment, the provisioning time[1] for an instance from a 4 GB image to a NC with a cold cache is approximately 5 mins 30 seconds. The provisioning time is reduced to approximate 1 min 50 seconds when the instance is deployed onto a NC with a hot cache[2]. We speculate that the provisioning time will increase significantly when multiple `ec2-start-instances` command is executed concurrently. This is due to the multiple copies of images and temporary files created during the image distribution process (Section 4.1).

During the life time of an instance, a user can run the `ec2-describe-instances` command to poll the current status of the VM. This command will also show the IP address(es) and type of all VM instances owned by the user. If the network is in MANAGED mode, a user can also requests for public IP addresses and allocate them to their VMs using the `ec2-allocate-address` and `ec2-associate-address` commands. In MANAGED mode, the user can also define ingress rules using `ec2-authorize` to limit connectivities to particular VMs they own.

To terminate a VM, the user issues an `ec2-terminate-instances` command. Once a VM is terminated, all changes made to the VM during its life time is lost. In Amazon EC2, an image saving mechanism (`ec2-bundle-vol`) is used to bundle the current state of a VM during execution into a new AMI, therefore

[1]The provisioning time is defined to be the submission of the `ec2-start-instances` command to `xend` starting the VM. This does not include the time the VM takes to boot after being started by `xend`. The boot time of a VM can range from a few seconds to minutes depending on the number of services to be started in the VM.

[2]Recall that the image cache in our Eucalyptus environment is an NFS exported directory shared among all the NCs

preserving all changes made in the VM. This bundling mechanism is not available in Eucalyptus but is being planned in the next stable release.

In addition to saving a VM image, storing other persistent data is a crucial requirement. Eucalyptus lacks in this aspect. Unlike Amazon's Elastic Block Store (EBS) which allows an EC2 user to attach external persistent storage to running VM instances, a Eucalyptus user will need to find other ways of exporting their persistent data out from the instance. This can be achieved by either manually copying the data out using `scp`, or moving them to a network directory attached in the instance, or uploading them to external storage services like Walrus (we demonstrate this storage methodology in Section 5) and S3.

4.3 User Management

Eucalyptus has been designed for its user management system to be fully web-based. A user first requests access to the Eucalyptus installation through its secure web portal by filling out a web form describing his or her profile. This request will generate an email to the administrator who will in turn log in to the web portal to grant the user access. Along with the creation of a new user, X.509 credentials and secret keys will be generated for the user to authenticate with the Eucalyptus services. A user will download these keys through the web portal and set up their local `.eucarc` file with the appropriate paths to the local installation of the Amazon API and AMI tools. Through the web portal, an administrator can also update user profiles as well as delete or disable user accounts.

Users can request for a password reset by confirming their profile email address. This system has a potential security flaw as email highjacking can easily compromise the entire user authentication scheme. A high jacker can reset the password through email and gain access to the user's account to download all credential information. Alas, this type of password reset scheme is common among most web portals, including Amazon EC2.

4.4 Support

The Eucalyptus developers provide comprehensive support service on their wiki. In addition to clearly structured installation and configuration documentations, an online forum is also available. While evaluating Eucalyptus, we also took part in active discussions on the forum. The developers and forum moderators are keen in answering questions about installation, usage, debugging and also their current development progress. The forum also serves as a medium for users to request for features they wish to see in upcoming releases of Eucalyptus.

4.5 Bugs and Limitations

In this section, we describe some of the bugs encountered in our installation and summarizes the limitations of Eucalyptus from our experience with the software.

4.5.1 Bugs

Throughout our evaluation effort, we found a number bugs with Eucalyptus v1.4. Apart from the design and usability issues that were highlighted in the previous sections, these bugs are a prohibiter from employing Eucalyptus as a full scale cloud solution to any production-type environment. Below we enumerate the major bugs detected.

1. In the STATIC network mode, after the CLC restarts, it loses all information about previous MAC or IP address assignments. A user that calls `ec2-describe-instances` after the CLC restarts will see the IP address field of all his or her running instances as `0.0.0.0`. At this point, if more VM instances are started, they will be assigned MAC and IP address combination that were previously assigned to running VMs and consequently resulting in conflicting IP address in the Eucalyptus cluster.

2. Starting multiple instances of different VM types will eventually render the system unresponsive, prohibiting it from creating new instances. The error message returned from the CLC during this situation is: `Server: SERVICE: FinishedVerify PROBLEM: null MSG-TYPE: RunInstancesType`. The solution to this is to restrict instances to only one VM type. A restart of Eucalyptus is required at this point to resume functionality.

3. Continuous SOAP calls (approximately 1 every 5 seconds) will eventually render the system unresponsive within 1 day. We detected this bug during the development of our demonstrator (Section 5) where we made `ec2-describe-instances` calls to the CLC every few seconds to poll the status of running VMs. After a day, the calls will all timeout with no errors or informative messages in the Eucalyptus logs. A restart of Eucalyptus is required at this point to resume functionality.

All the above mentioned bugs have been reported to the Eucalyptus developers.

4.5.2 Limitations

There are various limitations to Eucalyptus v1.4 due to its designs and development maturity. We have discussed most of these limitations in different sections throughout the document, here we fully enumerate them.

1. Eucalyptus v1.4 supports only one cluster of physical machines (Section 2.1).
2. Support Xen VMs only (Section 2.1).
3. A more comprehensive API should be developed to interface with Walrus(Section 4.1).
4. Storing and distributing EMIs requires a very large amount of storage space. In our case, this severely limited the number of base images we can store in Walrus (Section 4.1).
5. Both users and administrator do not have good information about the resource availability in the cluster (Section 4.2).
6. Images takes a long time to be provisioned. From our test, a 4 GB image takes approximately 5 min 30 seconds to provision to a cold cache and 1 min 50 seconds to provision to a hot cache. This time will significantly increase when multiple instances are provisioned simultaneously. Also note that the provisioning time does not include the time taken to boot the VM (Section 4.2).
7. Images cannot be saved. Once a VM is terminated, all changes made within the VM are lost (Section 4.2).

Some of the limitations described above are confined to v1.4. As we shall see in Section 6, new features in the first release candidate of Eucalyptus v1.5 addresses some of these limitations.

5 Demonstrator

We had previously developed an online animation rendering service as a demonstrator to illustrate potential real-world usage of cloud technology for social networking applications. Initially, this rendering service was used to showcase our own cloud solution, **ASPEN/** (Section 7.4). For this evaluation, we have ported the demonstrator to Eucalyptus in order to highlight how one can employ the EC2 interface and Eucalyptus instances to deploy web applications.

5.1 Rendering Service Environment

Figure 3 illustrates the rendering service environment. It consists of 2 components: 1) the service frontend as an interface to rendering users, and 2) a service backend that communicates with Eucalyptus to dynamically provision rendering nodes (VMs). A screen shot of the service frontend is shown in Figure 4.

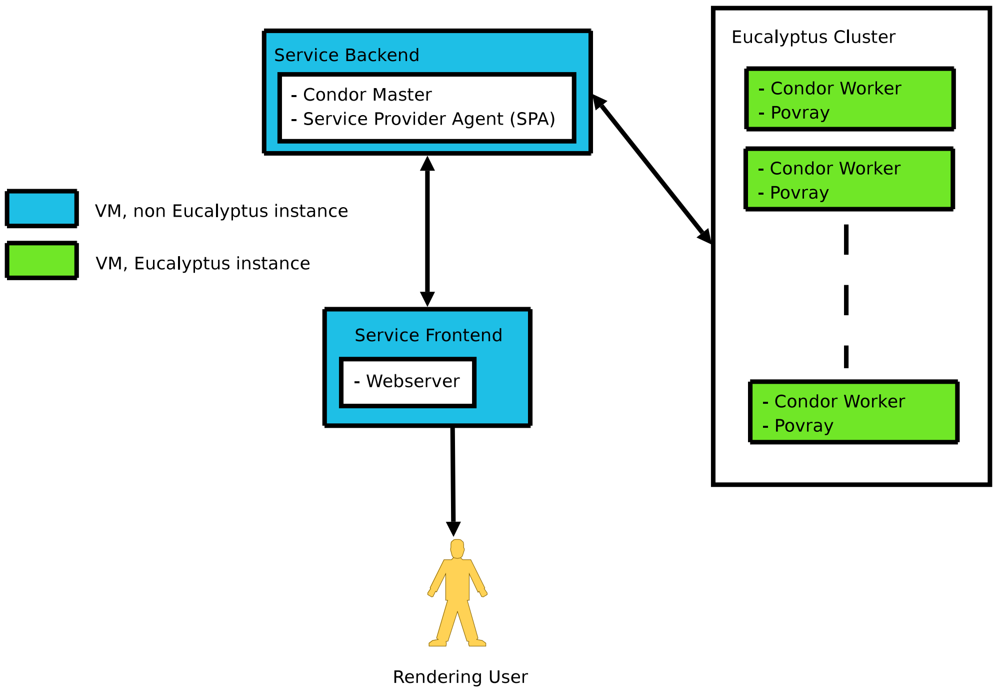
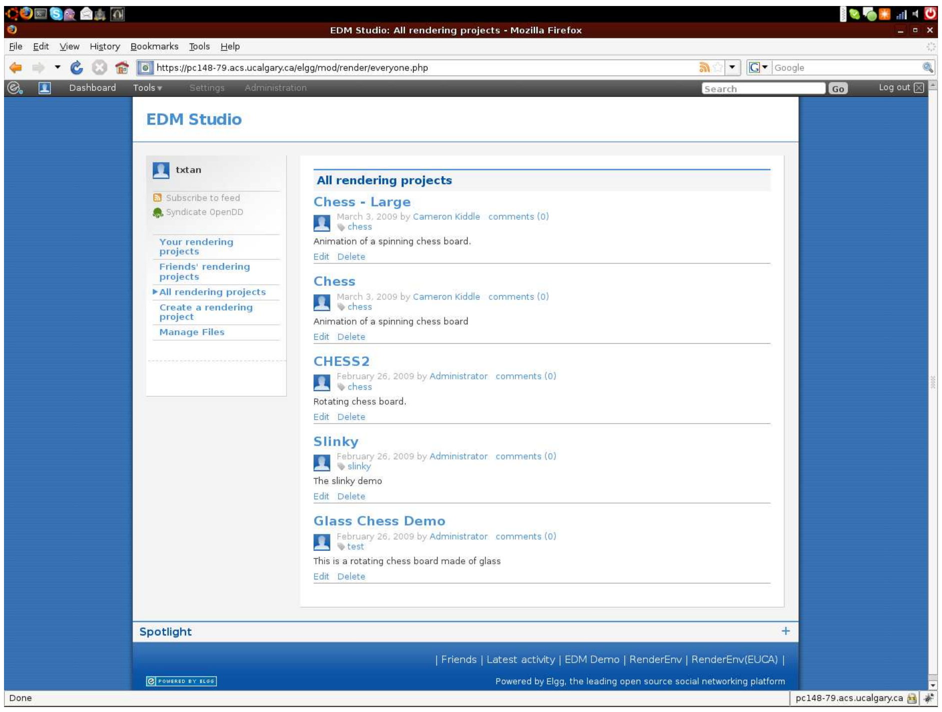


Figure 3: Rendering Service Environment


EDM Studio: All rendering projects - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://pc148-79.acs.ucalgary.ca/elgg/mod/render/everyone.php
Google
Dashboard Tools Settings Administration
Search Go
Log out
EDM Studio
txtan
Subscribe to feed
Syndicate OpenDD
Your rendering projects
Friends' rendering projects
All rendering projects
Create a rendering project
Manage Files
Spotlight
| Friends | Latest activity | EDM Demo | RenderEnv | RenderEnv(EUCA) |
POWERED BY ELGG
Done
pc148-79.acs.ucalgary.ca

All rendering projects

Chess - Large
March 3, 2009 by Cameron Kiddle comments (0)
chess
Animation of a spinning chess board.
Edit Delete

Chess
March 3, 2009 by Cameron Kiddle comments (0)
chess
Animation of a spinning chess board
Edit Delete

CHESS2
February 26, 2009 by Administrator comments (0)
chess
Rotating chess board.
Edit Delete

Slinky
February 26, 2009 by Administrator comments (0)
slinky
The slinky demo
Edit Delete

Glass Chess Demo
February 26, 2009 by Administrator comments (0)
test
This is a rotating chess board made of glass
Edit Delete

Powered by Elgg, the leading open source social networking platform

Figure 4: Rendering service frontend in Elgg

5.1.1 Service Frontend

Elgg [22] is an open source social networking platform supporting many aspects of building and customizing a social networking site. The Grid Research Center has used Elgg extensively to built social networking sites catered to research communities. Therefore the rendering service frontend is also built in the Elgg environment to leverage previous knowledge the group has in developing web applications. For this demonstrator, we use the original frontend from our previous rendering service making changes only to the backend where we will be communicating with Eucalyptus instead of ASPEN and using Walrus as storage for the rendering results.

The service frontend or Elgg is served via an Apache webserver. This webserver is a VM. However it is not a Eucalyptus instance. We choose to host this long-lived VM outside of the Eucalyptus environment because Eucalyptus does not currently support persistent images, therefore if the webserver VM is terminated, all information and changes made within the VM (and to its image) will be lost. This is an unfortunate limitation of the current version of Eucalyptus that restricts it from being production-ready. A similar limitation can be also observed in Amazon EC2. Regardless of the image saving mechanism, since Amazon makes no guarantees to the uptime of a VM instance, if an EC2 instance crashes or terminates before the user saved the image, all changes will be lost unless written to EBS.

A second reason why we host this VM outside Eucalyptus is because we need it to have a permanent IP. Due to the bug described in Section 4.5.1, when Eucalyptus restarts, it loses all IP information associated with already running VMs and assigns them again to new VMs. This behaviour is not acceptable for a long-lived VM like the web server.

The frontend interacts with rendering users, allowing them to upload rendering files and specify rendering descriptors. The frontend then passes the descriptors to the service backend for processing. It also displays rendered frames and movies to the user.

5.1.2 Service Backend

The service backend is the brain of the rendering service. It runs in a long-lived non-Eucalyptus VM. We choose to host this VM outside of Eucalyptus for the same reasons as in the service frontend, 1) we need a persistent image, 2) we want the VM to have a permanent IP.

The job of the backend is to create rendering jobs from rendering descriptors and submit the jobs into a Condor queue. We used Condor [6], a high throughput computing scheduler and job resource manager, to manage rendering jobs and maintain the availability of rendering nodes to run render jobs. The rendering nodes are Eucalyptus instances started from a custom EMI installed with the Condor software and the animation software Povray [21]. A monitoring agent called the Service Provider Agent (SPA) [8] monitors the Condor queue. As new rendering jobs arrive, it requests Eucalyptus to provision rendering nodes to run the jobs. It also requests Eucalyptus to terminate rendering nodes when no jobs are found in the Condor queue. In other words, the SPA serves as an entity to communicate resource requirements to a cloud provider (Eucalyptus in the current implementation) to achieve a dynamic resource provisioning environment to fully profit from the cloud computing paradigm.

5.2 Render Workflow

In Figure 5, we see an example of the rendering process. First, a rendering user uploads his or her Povray render description files, (Figure 6). These files are uploaded directly from the frontend interface to Walrus. The user then specifies some runtime rendering parameters to the frontend (Figure 7). The rendering descriptors are forwarded to the backend which creates and submits Condor Povray rendering jobs into the Condor Pool. One render job corresponds to one frame to be rendered. As new jobs appear in the queue, the SPA requests Eucalyptus to allocate rendering VMs to run the jobs. The SPA will only request up to a pre-defined number of instances. In the current demonstrator, this value is set to 8 (restricted by the 2 virtualization machines we have in our cluster). This implies that as long as there are render jobs in the queue, the SPA will request VMs from Eucalyptus, up to a maximum of 8 VMs.

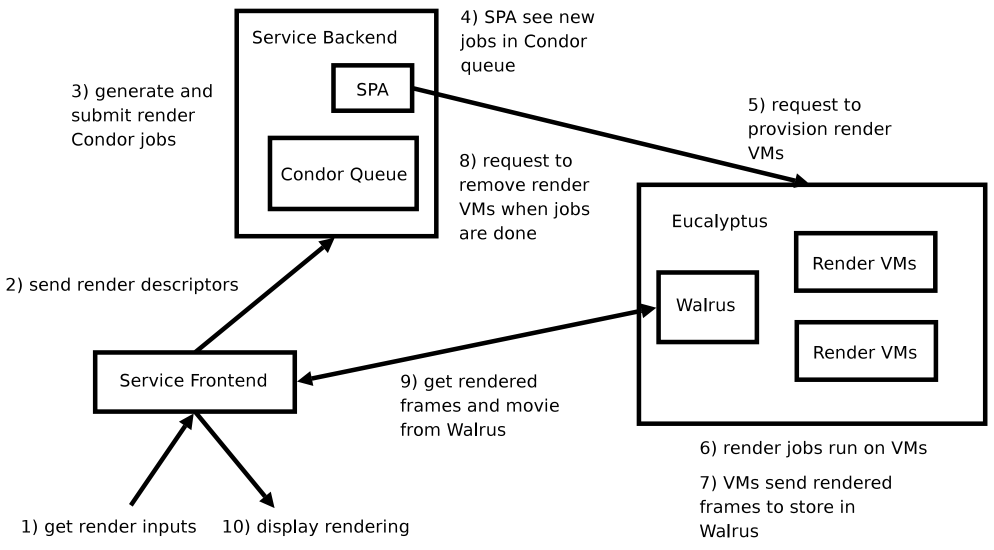

Service Backend
SPA
Condor Queue
4) SPA see new jobs in Condor queue
3) generate and submit render Condor jobs
5) request to provision render VMs
8) request to remove render VMs when jobs are done
Eucalyptus
Render VMs
Walrus
Render VMs
2) send render descriptors
Service Frontend
9) get rendered frames and movie from Walrus
6) render jobs run on VMs
7) VMs send rendered frames to store in Walrus
1) get render inputs
10) display rendering

Figure 5: Rendering Workflow

After the rendering VMs start, they automatically register themselves as Condor workers to the Condor Master. Subsequently, the render jobs in the queue will be launched on to the rendering VMs. Each job downloads from Walrus the rendering files particular to its rendering session. As a job completes its rendering, the frame rendered will be uploaded automatically to Walrus before the job terminates. This step is needed because the render VMs (started as Eucalyptus instances) do not have access to local persistent storage, therefore the frames have to be transfered out of the VMs if they want to be accessed at a later time.

As the number of jobs finishes and the queue size decreases, the SPA will request for Eucalyptus to deallocate the instances. For example, if the number of queue jobs reduces from 8 to 7, 1 VM will be deallocated by Eucalyptus from the maximum of 8 running VMs. This dynamic provisioning scheme ensures that no VMs are idling at any point in time. When the entire rendering is completed, a final job will be submitted into Condor to assemble the rendered frames into a movie. This movie will also be uploaded into Walrus. The frontend will download the frames and movies from Walrus to be displayed to the user at a later time (Figure 8).

6 Eucalyptus Version 1.5rc1

During the writing of this document, the first release candidate for Eucalyptus v1.5 was announced. This version introduces changes to old features and added a significant amount of new features. Although we did not install this version, from the official change log, we can see that some of the limitations described in Section 4.5.2 have been lifted. However, we do not know if the bugs reported in Section 4.5.1 have been fixed. Below is the official change log verbatim from the Eucalyptus support website.

6.1 Version 1.5rc1 Change Log

- Elastic Block Store (EBS) support (volumes & snapshots)
- Walrus improvements
  - Support for groups in ACLS

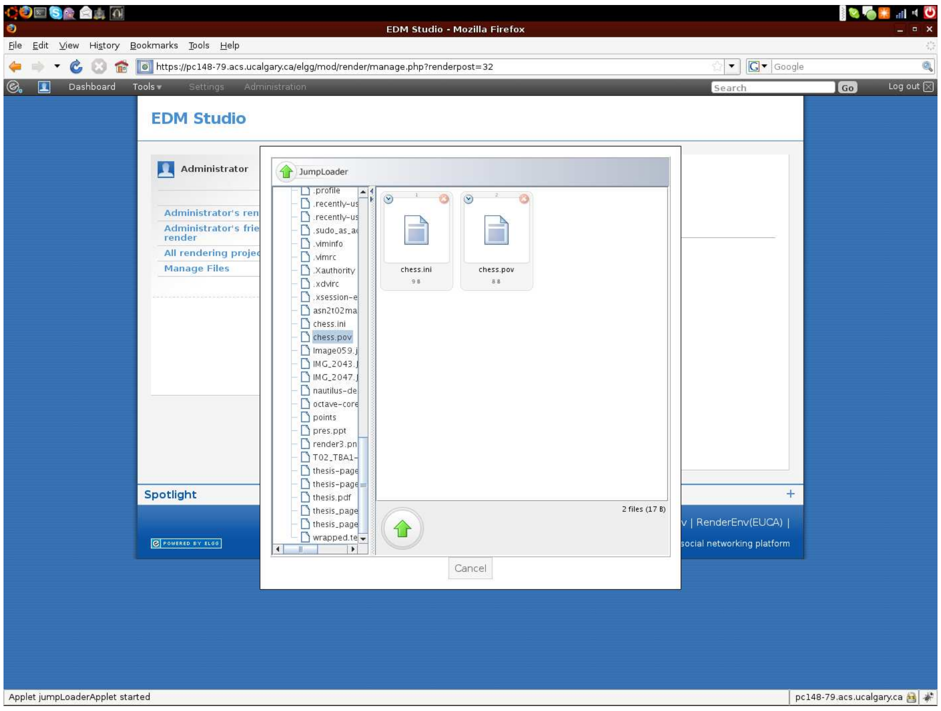
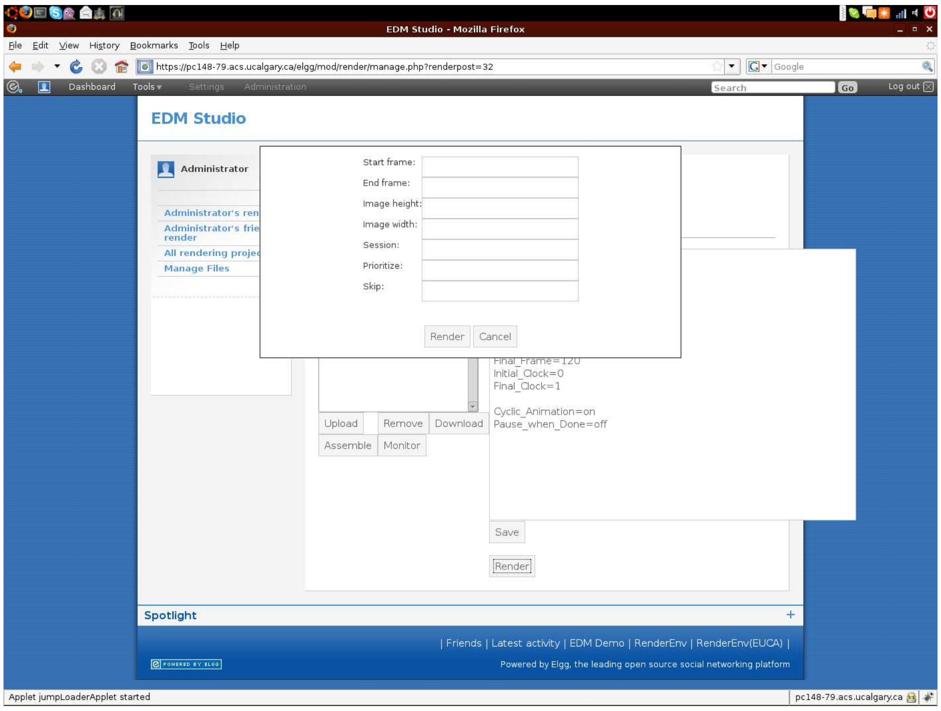


Figure 6: Uploading Povray rendering description files


EDM Studio - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://pc148-79.acs.ucalgary.ca/elgg/mod/render/manage.php?renderpost=32
Dashboard Tools Settings Administration
Search Go Log out
EDM Studio
Administrator
Administrator's ren
Administrator's frie render
All rendering projec
Manage Files
Render Cancel
Upload Remove Download
Assemble Monitor
Save
Render
Spotlight
| Friends | Latest activity | EDM Demo | RenderEnv | RenderEnv(EUCA) |
Powered by Elgg, the leading open source social networking platform
Applet jumpLoaderApplet started
pc148-79.acs.ucalgary.ca

Start frame:	
End frame:	
Image height:	
Image width:	
Session:	
Prioritize:	
Skip:	

```
Final_Frame=120
Initial_Clock=0
Final_Clock=1

Cyclic_Animation=on
Pause_when_Done=off
```

Figure 7: Rendering parameters submission form

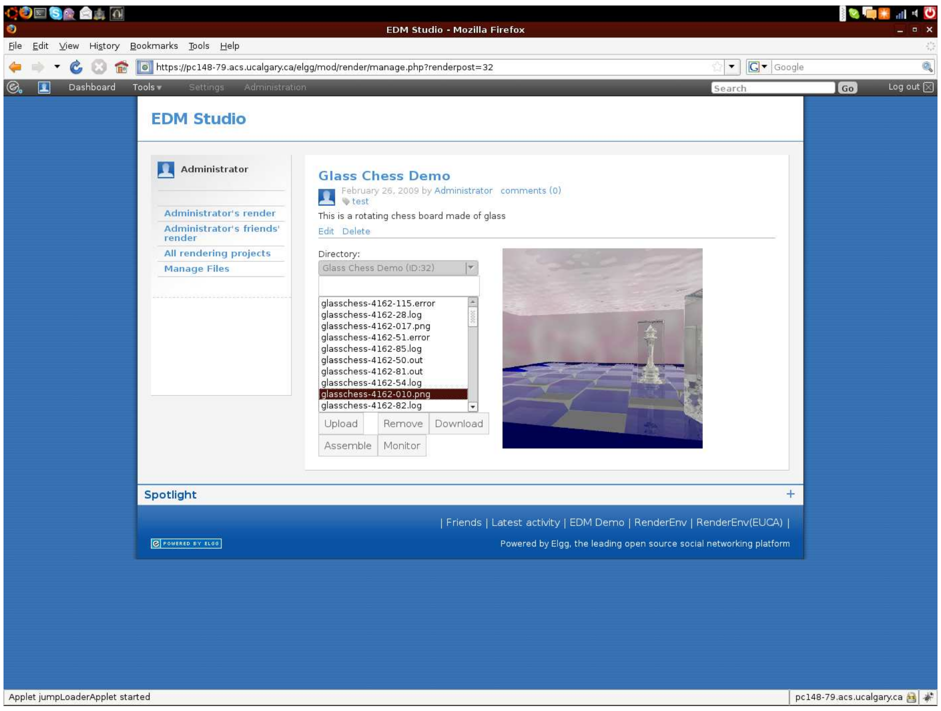

EDM Studio - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://pc148-79.acs.ucalgary.ca/elgg/mod/render/manage.php?renderpost=32
Dashboard Tools Settings Administration
Search Go Log out
EDM Studio
Administrator
Administrator's render
Administrator's friends' render
All rendering projects
Manage Files
Glass Chess Demo
February 26, 2009 by Administrator comments (0)
test
This is a rotating chess board made of glass
Edit Delete
Directory:
Glass Chess Demo (ID:32)
glasschess-4162-115.error
glasschess-4162-28.log
glasschess-4162-017.png
glasschess-4162-51.error
glasschess-4162-85.log
glasschess-4162-50.out
glasschess-4162-81.out
glasschess-4162-54.log
glasschess-4162-010.png
glasschess-4162-82.log
Upload Remove Download
Assemble Monitor
Spotlight
| Friends | Latest activity | EDM Demo | RenderEnv | RenderEnv(EUCA) |
Powered by Elgg, the leading open source social networking platform
Applet jumpLoaderApplet started
pc148-79.acs.ucalgary.ca

Figure 8: Viewing rendering frames

  - Fixed issues with meta data support
  - Web browser form-based uploads via HTTP POST
  - Object copying
  - Query string authentication
  - Support for arbitrary key names
  - Compressed image downloads and fixes to image caching
  - Reduced memory requirement

- Network improvement: new MANAGED-NOVLAN mode

- Node-side improvements:

  - Support for the KVM hypervisor
  - Compression & retries on Walrus downloads
  - Reworked caching (now with configurable limit)

- Web UI improvements:

  - Cloud registration with Rightscale (from admin's 'Credentials' tab)
  - New configuration options for Walrus
  - Better screening of usernames
  - Fixed account confirmation glitches

- Building and installation improvements

- Better Java installation checking
- New command-line administration: euca_conf -addcluster ... -addnode ...
- Non-root user deployment of Eucalyptus
- Binary packages for more distributions (Ubuntu et al)

From these changes, we see that a new method of compressing images and image caching is introduced. This has the potential to decrease both the storage requirements and distribution time of EMIs which may in turn result in faster provisioning time for instances. We also note that an implementation of EBS has been introduced, thus allowing Eucalyptus users to attach and detach external block devices to store persistent data from within the VM. However, there is currently still no support for saving images. Support for KVM is also enabled in this version and a new network mode that does not use `vlan` is implemented.

7 Related Work

Eucalyptus is just one of the many cloud computing platforms available on the market today. In this section, we present a brief survey of other cloud contributions from both commercial and academic organizations. We will focus on the comparison of key features and functionality of research-based platforms, giving minimal treatment to the proprietary solutions.

7.1 Amazon Web Services

Amazon is considered a fore-runner in offering a reliable and simple solution to access the cloud. Amazon Web Services (AWS) [24], as the name implies, is a suite of web services providing on demand computing power, storage and networking for businesses to grow their computing requirements elastically in the cloud. We have already referred to some of these services throughout the document. Specifically, we described how the Simple Storage Service (S3) and Elastic Cloud (EC2) is emulated within Eucalyptus.

AWS is a proprietary product. Due to its early involvement, it is considered the de facto standard in cloud computing. Most other emerging open source cloud platforms provide some functionality to integrate with AWS. For example, both Open Nebula (Section 7.2) and Nimbus (Section 7.3) supports starting of Amazon instances within their infrastructure. Eucalyptus on the other hand based their entire design on emulating Amazon EC2 and S3.

7.2 Open Nebula

Open Nebula [18] is an open source cloud platform developed and maintained at the Universidad Complutense de Madrid. It is part of Reservoir [23], a major EU-funded research project for resources and services virtualization. Open Nebula is based on a highly modular design featuring components that perform scheduling, VM management, image management and network management. Internally it uses Sun Grid Engine (SGE) [11] for resource management and scheduling. Users interact with Open Nebula through a command line interface (CLI). VM instances are created by specifying their properties via "VM templates". Currently, users can create both Xen and KVM VMs. They can also start EC2 instances via the Open Nebula interface by using a EC2-defined VM template.

The image management component in Open Nebula is similar to Eucalyptus. VM images are first "cloned" from a base image. Like Eucalyptus, the cloning process is basically executing a `cp` command on the raw image file. The cloned images are then copied using ssh or shared via NFS onto the virtualization nodes. However, in Open Nebula, images are not deleted after an instance terminates. This preserves the state of a VM even after it shuts down.

The time taken to provision an instance is on the same order as Eucalyptus since they are both using similar image management strategies. An API is also available to allow Open Nebula to talk to external storage services. This potentially means that a more efficient method can be used for image distribution.

Another component is the scheduling module. Internally, Open Nebula uses a simple queueing system[3] to prioritize VM creations. However the default scheduling behaviour can be changed by integrating with a more sophisticated scheduler. The Haizea Lease Manager [25] is one scheduler that has been successfully integrated with Open Nebula via its scheduler API. Using Haizea, advanced reservation and preemption can be implemented.

7.3 Nimbus

Nimbus [19], previously called Virtual Workspace Service, is an open source Globus [2] product for creation of virtual clusters. It is similar in many aspects to Open Nebula for creation and management of Xen VM instances. Current ongoing efforts are to include support for both KVM and VMWare within Nimbus. The main difference comparing Nimbus to Open Nebula is that Nimbus uses grid credentials to authenticate user requests. A user can also start EC2 instances via the same set of credentials making it easier than Open Nebula to interact with EC2. Image management is similar to Open Nebula and Eucalyptus, raw file copies are a common methodology among these platforms. The state of a VM can be "saved" by making a copy of the current image. There is also the notion of an image repository where each user can upload VM images to his or her individual repository.

7.4 ASPEN

The Automated Service Provisioning ENvironment or ASPEN [8] is a research project of the Grid Research Centre. We developed this cloud platform to provide a dynamic provisioning environment for both batch and interactive workloads. We have used ASPEN extensively to provision virtual environments for both commercial and academic projects. ASPEN currently supports Xen VMs, but there are plans to support other virtualization technology like KVM in the near future.

ASPEN is similar in many ways to the other cloud platforms discussed before. It supports dynamic VM creation, VM resizing in terms of CPU and memory, and persistent storage. However ASPEN differs in two major aspects. First, ASPEN introduced a novel approach to image management. Using Sun's ZFS as the underlying file system of an ASPEN image server, fast VM image cloning and snapshotting functionalities can be implemented. VM images are created using ZFS's Copy-On-Write (CoW) mechanism, saving time and space requirements for the provisioning of VM instances. ASPEN also supports more image formats than other platforms. These include file based, blocked based and even network rooted images.

The second advantage that ASPEN has over existing cloud platform is its sophisticated VM scheduling technology. By integrating with the Moab Workload Manager [17], ASPEN can utilize advanced reservations, preemption, backfill and a highly customizable priority system for flexible scheduling of VM instances based on different user and group policies. It can also harness the resizing, suspension and checkpointing capabilities of VMs to automatically optimize VMs to physical machine deployment. For example, automatically migrating VMs for load balancing or packing VMs across fewer physical machines to reduce energy, cooling and other operational costs [26].

7.5 Feature Chart

In this section, we summarize the features of each cloud technology. Open Nebula can be considered among all, to be a very comprehensive cloud platform by supporting most aspects of cloud computing requirements. However it is also known to take upwards of minutes to provision a single Open Nebula instance. Globus Nimbus allows easy VM creation of both Nimbus and EC2 instances using a common set of grid credentials. ASPEN provides a highly sophisticated VM scheduling system customizable to meet site specific needs. It also employs advance CoW and snapshotting capabilities for fast image cloning. An image cloning operation in ASPEN is in the order of seconds compared to minutes by the other cloud platforms.

[3]It is unclear to what extend SGE's scheduling capabilities are incorporated into Open Nebula.

Except for Eucalyptus v1.4, all these platforms allow saving of images. However as we saw in Section 6, Eucalyptus is addressing crucial issues on persistent data storage. This means that as of v1.5, Eucalyptus will be in a better position as a cloud platform for deployment in production environments. Table 1 compares the main features of each open source cloud solution described.

7.6 Other Cloud Platforms

All the cloud technologies covered above are offering services that builds on demand and dynamic resource provisioning infrastructures. This computing paradigm is called Infrastructure-as-a-Service (IaaS). There are many other platforms that provide IaaS. For example, two recent startups, Abiquo and Enomaly Inc offers elastic computing and infrastructure management services respectively through their open source cloud platforms, AbiCloud [1] and Enomaly [12]. Other companies that have a longer history and are offering closed source solutions are Platform Computing with their Virtual Machine Orchestrator [27], VMWare with Virtual Center [7] and 3Tera with AppLogic [4].

Apart from IaaS, Platform-as-a-Service (PaaS) offers an integration computing and development environment driven by cloud infrastructures. Companies like Google offers PaaS through its Google App Engine [10]. Customers using Google App Engine can build and maintain scalable web applications in the Google hosting environment. Microsoft's Azure [5] is a similar service that promises scalable and dynamic hosting of internet applications. Force.com [14] is another PaaS provider offering an integrated web development platform driven by the cloud.

8 Conclusion

We have investigated the Eucalyptus cloud computing platform as a general solution to deploying cloud infrastructures. We documented the design, installation and usability of the version 1.4 platform. We have also developed a demonstrator that illustrates a real-world use case of the Eucalyptus cloud. The general impression of Eucalyptus is that it is easy to install and configure. It has a well-defined interface that is borrowed from Amazon EC2. The functionality that Eucalyptus supports matches closely with what EC2 is offering, however a lack of certain features and major bugs in the current version proves that Eucalyptus is not yet ready for deployment on production-type environments. The first release candidate for Eucalyptus version 1.5 was announced during the writing of this document. With this version and its new features, Eucalyptus is in a better position as a major competitor in the cloud market.

Cloud	VM Type	Interface	Image Management	Networking	VM Functions	Scheduling
Eucalyptus	Xen	CLI, web portal	local file copy, distribute images via SOAP	security groups, elastic IP	none	first-fit, round robin
Open Nebula	Xen, KVM	CLI	local file copy, distribute via ssh or NFS	none	save, pause, resize, live migrate	queueing system, advanced reservation, preemption
Nimbus	Xen	CLI	file copy, unknown distribution method	none	save	unknown
ASPEN	Xen	CLI	CoW, snapshots, distribute over NFS	none	save, pause, resize	advanced reservation, backfill, preemption, customizable priority queueing system, policy

Table 1: Feature comparison between different cloud platforms

References

[1] abiCloud. http://www.abiquo.com/en/products/abicloud.

[2] Globus Alliance. http://www.globus.org.

[3] Elastic Compute Cloud Amazon Web Services. http://aws.amazon.com/ec2.

[4] 3Tera AppLogic. http://www.3tera.com/applogic/.

[5] Microsoft Azure. http://www.microsoft.com/azure.

[6] Jim Basney, Miron Livny, and Todd Tannenbaum. High throughput computing with Condor. *HPCU news*, 1, 1997.

[7] VMware Virtual Center. http://www.vmware.com/products/vi/vc/.

[8] R. Curry, C. Kiddle, N. Markatchev, R. Simmonds, T. Tan, M. Arlitt, and B. Walker. ASPEN: an automated service provisioning environment for data centres. *Proceedings of the 15th HP Software University Association Workshop*, 2008.

[9] Eucalyptus Documentation. http://eucalyptus.cs.ucsb.edu/wiki/documentation.

[10] Google App Engine. http://code.google.com/appengine.

[11] Sun Grid Engine. http://http://gridengine.sunsource.net.

[12] Enomaly. http://www.enomaly.com/.

[13] N. Fallenbeck, H.J. Picht, M. Smith, and B. Freisleben. Xen and the art of cluster scheduling. *Proceedings of 1st International Workshop on Virtualization Technology in Distributed Computing*, 2006.

[14] Force.com. http://developer.force.com/.

[15] Eucalyptus in Karmic Koala. https://lists.ubuntu.com/archives/ubuntu-devel-announce/2009-february/000536.html.

[16] Kernel Virtual Machine. http://kvm.qumranet.com/kvmwiki.

[17] Moab Workload Manager. http://www.clusterresources.com/pages/products /moab-cluster-suite.php.

[18] Open Nebula. http://www.opennebula.org.

[19] Nimbus. http://workspace.globus.org.

[20] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus open-source cloud-computing system. *Proceedings of Cloud Computing and Its Application*, 2008.

[21] Persistence of Vision Raytracer. http://www.povray.org.

[22] Elgg Social Networking Platform. http://www.elgg.org.

[23] Resources and Services Virtualization without Barriers. http://www.reservoir-fp7.eu/.

[24] Amazon Web Services. http://aws.amazon.com.

[25] B Sotomayor, K Keahey, and I. Foster. Combining batch execution and leasing using virtual machines. *Proceedings of the ACM/IEEE International Symposium on High Performance Distributed Computing*, 2008.

[26] Tingxi Tan. Scheduling Virtual Machines in Data Centres. *Masters Thesis, University of Calgary*, 2009.

[27] Platform Computing VMO. http://www.platform.com/products/platform-vm-orchestrator/.

[28] VMWare. http://www.vmware.com/.