ACKNOWLEDGEMENT

It is a great pleasure to acknowledge the support and encouragement received in the successful completion of this project. It is a privilege to express my profound gratitude and indebtedness to my supervisors ***Prof. Arun Agarwal*** *and* ***Dr. Rajeev Wankar*** *for their valuable, inspiring and constant support throughout the progress of this project. Their suggestions helped me to improve my work a lot.*

I'm grateful to the Head of Department (DCIS), ***Prof****.* ***Arun Agarwal*** *for providing us both the freedom and an excellent lab facility where we could test our work effectively. I would like to extend my sincere thanks to* ***Dean*** *of School of Mathematics, Computers and Information Sciences for his valuable cooperation.*

I `m extremely thankful to ***A.I Lab*** *staff for their kind cooperation. A special word of appreciation to Babu rao, Rahul, Rajesh, Kiran and Sriharsha and also rest of my friends for their valuable suggestions and moral support.*

Finally I express my gratitude to all my parents for their love and affection, without which this work would not have been possible.

A V Naveen Kumar

Table of Content

List of Figures

List of Tables

Chapter 1

Introduction

As there is a significant increase in information and communication technology from past decade, there is a clear vision that computing will be one of the major utility. Many utilities have been into existence; latest one is the "***Cloud Computing***" [1]. Most of our data is stored on local networks with servers that may be clustered and sharing storage. Transferring this huge amount of data has many constraints. One of the constraint is the throughput in data transfer. This approach has had time to be developed into stable architecture, and provide decent redundancy when deployed right. As a newer emerging technology, cloud computing, has shown up demanding attention and is quickly changing the direction of the technology landscape and the way how things are to be done.

Hence in this thesis we define *cloud computing* and provide the architecture for creating IaaS clouds with open resources privately. We also present the performance evaluation for the data transfers is been done frequently in the cloud. In addition to this, we also provide complete detailed description on open source cloud architecture, resource allocation in the cloud and the basic service models in the cloud. Furthermore, we also evaluate the differences between various current sets of computing powers.

1.1 What is a Cloud?

``A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers."[1] --*Buyya*

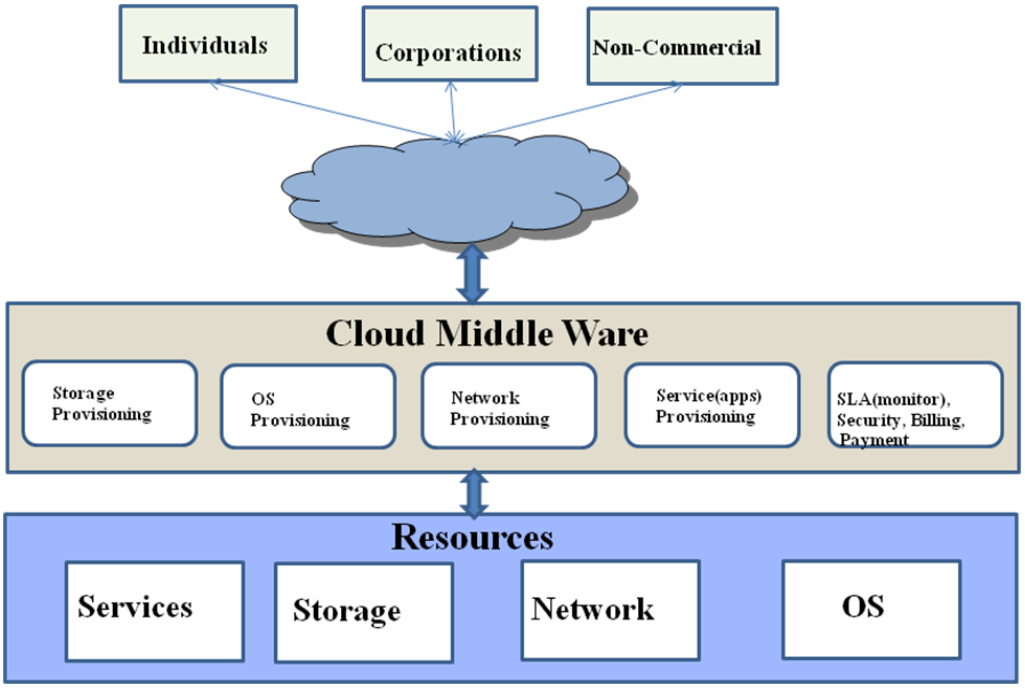


Figure 1.1 Complete cloud design

A cloud (Figure 1.1) is a generic term for "Internet", which simply refers to group of networks. Hence the cloud is a physical place, perhaps owned and controlled by some other entity, and it contains computing resources that are available on demand for a price.

1.2 Evolution to Cloud Computing

The *Figure 1.2* shows the evolution in computing power starting from Grid computing followed by utility computing, reaching to software as a service and presently holding a position in cloud computing. The emergence mainly involved in new advances in processors, virtualization technology, distributed storage, broadband Internet access, automated management and fast inexpensive servers have all combined to make cloud computing a compelling paradigm. This vast process power is usually got with distributed, large-scale server cluster and server virtualization software.

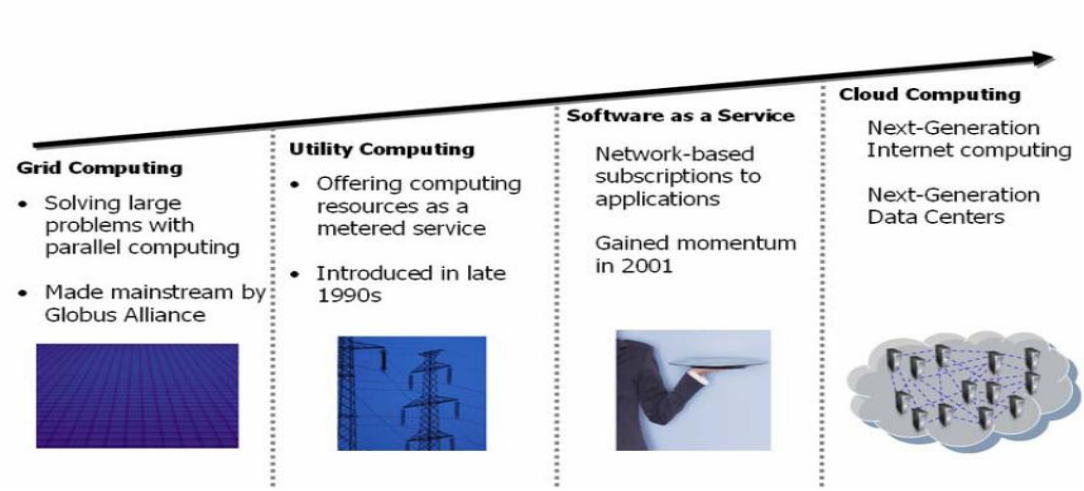


Figure 1.2 Evolution to cloud computing.

1.3 Why Cloud Computing?

Cloud computing has the potential to upend the software industry entirely, as applications are purchased, licensed and run over the network instead of a user's desktop. This shift will put data centers and their administrators at the center of the distributed network, as processing power, electricity, bandwidth and storage are all managed remotely. It affects not only business models, but the underlying architecture of how we develop, deploy, run and deliver applications.

1.4 Types of Clouds?

Organizations can choose to deploy applications on public, private, or hybrid clouds, each of which has its trade-offs. The terms *public*, *private*, and *hybrid* do not state location. While public clouds are typically "out there" on the Internet and private clouds are typically located on premises, a private cloud might be hosted at a collocation (side-by-side) facility as well.

Organizations may make a number of considerations with regard to which cloud computing model they choose to employ, and they might use more than one model to solve different problems. An application needed on a temporary basis might be best suited for deployment in a public cloud because it helps to avoid the need to purchase additional equipment to solve a temporary need. Likewise, a permanent application,

or one that has specific requirements on quality of service or location of data, might best be deployed in a private or hybrid cloud [4].

1.4.1 Public Cloud

Public clouds are run by third parties, and applications from different customers are likely to be mixed together on the cloud's servers, storage systems, and networks as shown in *Figure 1.3*. Public clouds are most often hosted away from customer premises, and they provide a way to reduce customer risk and cost by providing a flexible, even temporary extension to enterprise infrastructure. If a public cloud is implemented with performance, security, and data locality in mind, the existence of other applications running in the cloud should be transparent to both cloud architects and end users. Indeed, one of the benefits of public clouds is that they can be much larger than a company's private cloud might be, offering the ability to scale up and down on demand, and shifting infrastructure risks from the enterprise to the cloud provider, if even just temporarily.

Portions of a public cloud can be carved out for the exclusive use of a single client, creating a virtual private datacenter. Rather than being limited to deploying virtual machine images in a public cloud, a virtual private datacenter gives customers greater visibility into its infrastructure. Now anyone can manipulate not just virtual machine images, but also servers, storage systems, network devices, and network topology, deployed at a collocation facility. [4]

Creating a virtual private datacenter with all components located in the same facility helps to lessen the issue of data locality because bandwidth is plentiful and typically free when connecting resources within the same facility.

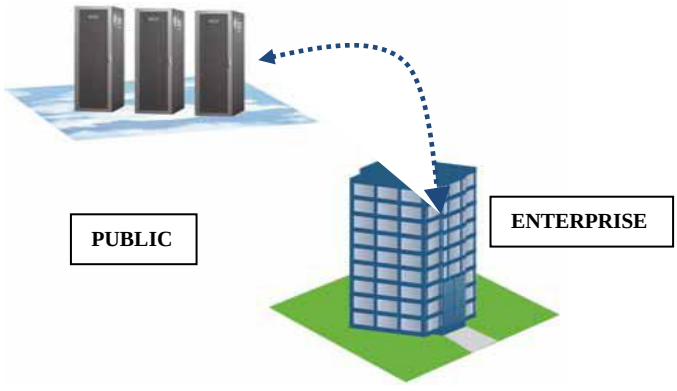


Figure 1.3 A public cloud provides services to multiple customers, and is typically

1.4.2 Private Cloud

Private clouds are built for the exclusive use of one client, providing the utmost control over data, security, and quality of service (*Figure 1.4*). The company owns the infrastructure and has control over how applications are deployed on it. Private clouds may be deployed in an enterprise datacenter, and they also may be deployed at a collocation facility.

Private clouds can be built and managed by a company's own IT organization or by a cloud provider. In this "hosted private" model, as we installed the Eucalyptus cloud privately within the university, we configure and operate the infrastructure to support a private cloud within the university datacenter.

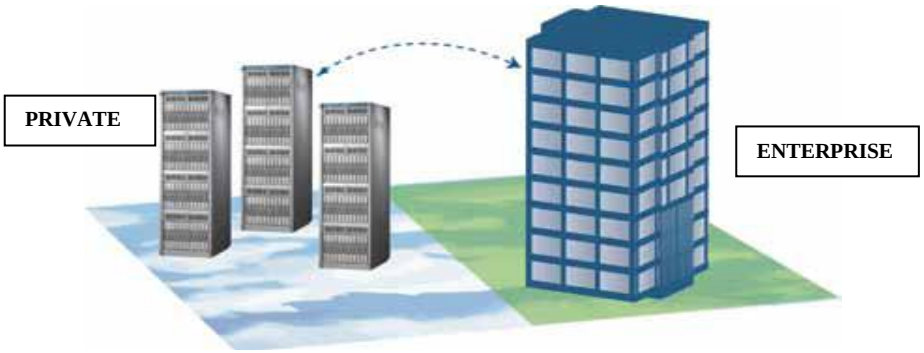


Figure 1.4 Private clouds may be hosted at a collocation facility or in an enterprise datacenter. They may be supported by the company, by a cloud provider, or by a third party such as an outsourcing firm.[4]

1.4.3 Hybrid Cloud

Hybrid clouds combine both public and private cloud models (*Figure 1.5*). They can help to provide cloud on-demand, externally provisioned scale. The ability to augment a private cloud with the resources of a public cloud can be used to maintain service levels in the face of rapid workload fluctuations. This is most often seen with the use of storage clouds to support Web 2.0 applications. A hybrid cloud also can be used to handle planned workload spikes. Sometimes called "surge computing," a public cloud can be used to perform periodic tasks that can be deployed easily on a public cloud.

Hybrid clouds introduce the complexity of determining how to distribute applications across both a public and private cloud. Among the issues that need to be considered is the relationship between data and processing resources. If the data is small, or the application is stateless, a hybrid cloud can be much more successful than if large

amounts of data must be transferred into a public cloud for a small amount of processing.

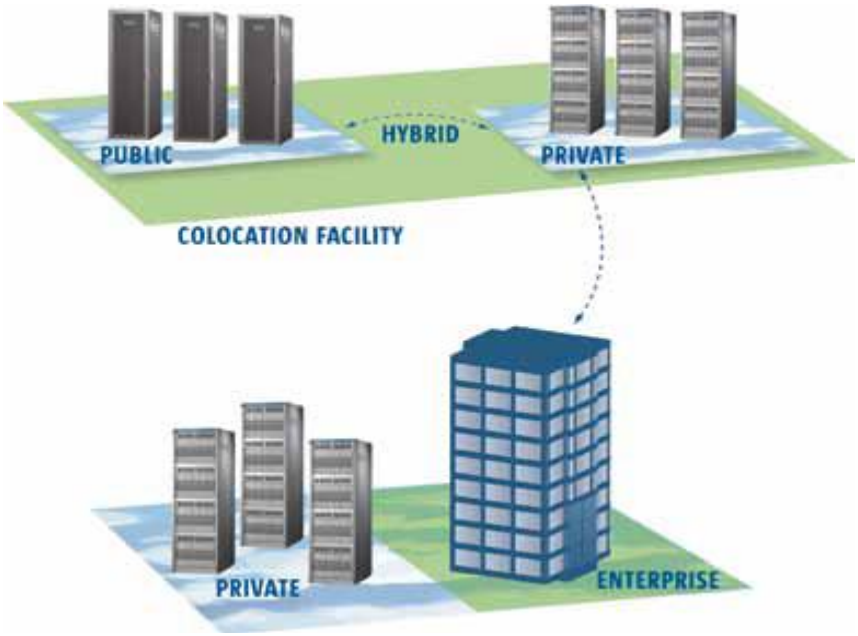


Figure 1.5.Hybrid clouds combine both public and private cloud models, and they can be particularly effective when both types of cloud are located in the same facility. [4]

1.5 Virtualization in clouds

Virtualization is a keystone design technique for all cloud architectures. In cloud computing it refers primarily to platform virtualization or the abstraction of physical resources from the people and applications using them. Virtualization allows servers, storage devices, and other hardware to be treated as a pool of resources rather than separate systems, so that these resources can be allocated on demand. In cloud computing, we're interested in techniques such as *paravirtualization*, which allows a single server to be treated as multiple virtual servers, and *clustering*, which allows multiple servers to be treated as a single server. However we have done full virtualization, where the operating systems are deployed directly on the hypervisor (XEN) for creating eucalyptus cloud. Here compute nodes are made on multiple machines, but for the user it looks like single machine with whole set of combined processors and storage. Some specific advantages, including:

Higher utilization rates — Prior to virtualization, server and storage utilization rates in enterprise datacenters typically averaged less than 50%. Through

virtualization, workloads can be encapsulated and transferred to idle or underused systems — which means existing systems can be consolidated, so purchases of additional server capacity can be delayed or avoided.

Resource consolidation — Virtualization allows for consolidation of multiple IT resources. Beyond server and storage consolidation, virtualization provides an opportunity to consolidate the systems architecture, application infrastructure, data and databases, interfaces, networks, desktops, and even business processes, resulting in cost savings and greater efficiency.

Lower power usage/costs — The electricity required to run enterprise-class datacenters is no longer available in unlimited supplies, and the cost is on an upward spiral. Using virtualization to consolidate makes it possible to cut total power consumption and save significant money.

Space savings — Server spread out remains a serious problem in most enterprise datacenters, but datacenter expansion is not always an option, with building costs averaging several thousand dollars per square foot. Virtualization can alleviate the strain by consolidating many virtual systems onto fewer physical systems, as we have done in the present project.

Disaster recovery/business continuity — Virtualization can increase overall service-level availability rates and provide new options for disaster recovery solutions.

1.6 Description of Work Done

Following is the description of the work carried out for the development of Eucalyptus Private Cloud at University of Hyderabad:

1. Set-up of Eucalyptus private cloud 1 and 2 on six HP dc7900 Core2Duo V-PRO Systems (64bit support) with Open Suse v11.2 (64bit) using XEN 3.4 is done.

2. Maintaining Cloud Controller, Cluster Controller, Node Controller and Storage for both clouds.

3. Deployed different Operating Systems including their kernels and Servers (basic version without GUI) like FEDORA 11 32bit & 64bit, CENTOS 5.3 32bit & 64bit, Ubuntu 9.04 32bit & 64bit, Debian5.0 32bit & 64bit and Karmic server 32bit & 64bit.

4. Able to run instances of different OS within the LAN.

5. Instances are been accessed through Command Line and through open source plug-ins (Hybridfox, s3fox etc) available.

6. About 10 users have been added to access the cloud.

7. Max of 4 instances of size 2GB and 5GB storage can be accessed at a time with their own IP address (Separate IP within the LAN is given to each instance). A maximum of 2-instance can be accessed with 10GB and 20GB storage. (Storage is nothing but the space the OS instance takes to run).

8. Successfully deployed Globus 4.2.1 in the clouds for using Grid FTP and UDT for the data transfer performances.

1.7 Organization of Thesis

This thesis has been organized giving complete information on evolution to clouds and use of virtualization in chapter 1. Followed by, discussing and differentiating various computing powers and also describing the service models in the cloud in chapter 2. In chapter 3 we gave complete information on what is eucalyptus cloud, its features & characteristics. Chapter 4 discusses about the data transfer protocols. Chapter 5 includes complete details of the framework and architecture of the system. Different ways to access the cloud and the performance evaluations of data transfer protocols. This thesis ends up with chapter 6, which includes conclusion and future work.

Chapter 2

Cloud Computing

2.1 Cluster Computing

``A cluster is a type of parallel and distributed system, which consists of a collection of inter-connected stand-alone computers working together as a single integrated computing resource.[1]" --Buyya.

Cluster computing is a form of computing in which a group of computers are linked together so that they can act like a single entity. There are a number of reasons for people to use cluster computers for computing tasks, ranging from an inability to afford a single computer with the computing capability of a cluster to a desire to ensure that a computing system is always available. One common reason to use cluster computing is a desire to create redundancy in a computer network to ensure that it will always be available and that it will not fail.

2.2 Grid computing

``A Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed `autonomous' resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements.[1]"

Grid computing is a term for either of two broad subcategories of distributed computing:

1. Data grids provide controlled sharing and management of large amounts of distributed data, often used in combination with computational grids.

2. The creation of a "virtual supercomputer" composed of a network of loosely-coupled computers, acting in concert to perform very large tasks. This

technology has been applied to computationally-intensive scientific, mathematical, and academic problems through volunteer computing, and it is used in commercial enterprises for such diverse applications as drug discovery, economic forecasting, seismic analysis, and back-office data processing in support of e-commerce and web services.

2.3 Utility Computing

Utility computing is a business model of providing computing resource, user get and uses the computing resource from service provider and pay for practically used resource. To say it simply, it is a price model based on resource usage quantity. The main benefit of utility computing is better economics. Corporate data centers are notoriously underutilized, with resources such as servers often idle 85 percent of the time. This is due to over provisioning. Buying more hardware than is needed on average in order to handle peak, to handle expected future loads and to prepare for unexpected courses in demand. Utility computing allows companies to only pay for the computing resources they need, when they need them.

2.4 Cloud Computing

2.4.1 What is Cloud Computing?

Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly made available on demand and released with least management effort or service provider interaction.

2.4.2 Cluster Vs Grid Vs Cloud Computing

A set of characteristics that helps distinguish cluster, Grid and Cloud computing systems is listed in *Table 2.1*. The resources in clusters are located in a single administrative domain and managed by a single entity whereas, in Grid systems, resources are geographically distributed across multiple administrative domains with their own management policies and goals.

Another key difference between cluster and Grid systems arises from the way application scheduling is performed. The schedulers in cluster systems focus on enhancing the overall system performance and utility as they are responsible for the whole system.

On the other hand, the schedulers in Grid systems called resource brokers, focusing on enhancing the performance of a specific application in such a way that its end-users' QoS requirements are met.

Cloud computing platforms possess characteristics of both clusters and Grids, with its own special attributes and capabilities such strong support for virtualization, dynamically composable services with Web Service interfaces, and strong support for creating 3rd party, value added services by building on Cloud compute, storage, and application services.

Thus, Clouds are promising to provide services to users without reference to the infrastructure on which these are hosted.

Characteristics	**Cluster**	**Grid**	**Cloud**
Resources Management	Centralized	Distributed	Centralized/ Distributed
Ownership	Single	Distributed	Single
Allocation/scheduling	Centralized	Distributed	Centralized/ Distributed
Size	100 connections	1000 connections	100s to 1000s connections
Service negotiation	Limited	SLA , Local between Enterprise grid and resources provider	SLA, Global between the cloud provider and users
Pricing of services	Limited, not open market	Dominated by public good or privately assigned Utility pricing	Discounted for larger customers

Potential for building 3rd party or value-added solutions	Limited due to rigid architecture	Limited due to strong orientation for scientific computing	High potential _ can create new services by dynamically provisioning of compute, storage, and application services and offer as their own isolated or composite Cloud services to users
Internetworking	Multi-clustering within an Organization	Limited adoption, but being explored through research efforts such as Gridbus InterGrid	High potential, third party solution providers can loosely tie together services of different Clouds
Node Operating System (OS)	One of the standard OSs (Linux, Windows)	Any standard OS (dominated by Unix)	A hypervisor (VM) on which multiple OSs run

Table 2.1 Showing key characteristics between cluster, grid and cloud systems [1]

2.4.3 Cloud Computing Vs Utility Computing

Utility computing is a business model; it is a type of price model to deliver application infrastructure resource. Cloud computing is a computing model, relates to the way we design, build, deploy and run applications that operate in a sharing resources and boasting the ability to dynamically grow, shrink and self-heal. Utility computing is often need a cloud computing infrastructure, but not must need.

2.4.4 Cloud Service Models

Cloud service models are broadly categorized into 3 parts. Infrastructure as a service, Platform as a service and Software as a service. In short IaaS, PaaS, SaaS.

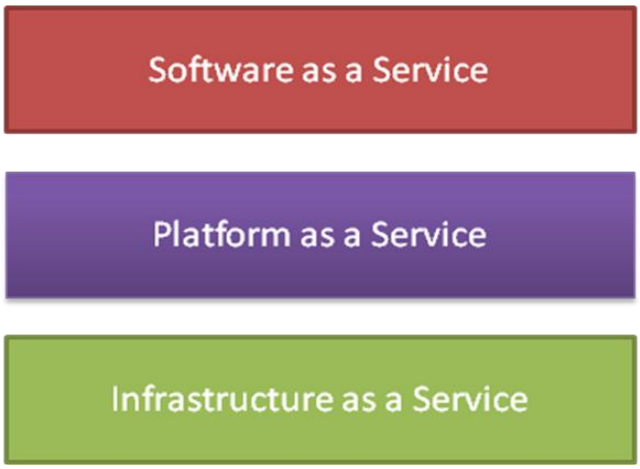


Figure 2.1 Cloud service models, SaaS, PaaS & Iaas

Software as a service is a way where we use provider's applications over a network. Platform as a Service is a way to deploy customer-created applications to a cloud, and finally Infrastructure as a Service is nothing but renting processing, storage, network capacity, and other fundamental computing resources. These 3 service models are described briefly as follows:

2.4.4.1 Software as a Service (SaaS)

SaaS is the ability to access software over the Internet as a service. An early approach to SaaS was the Application Service Provider (ASP). ASPs provide subscriptions to software that is hosted or delivered over the Internet. The ASP delivers the software and charges fees based on its use. In this way, you don't purchase the software but simply lease it on an as-needed basis.

Another perspective on SaaS is the use of software over the Internet that executes remotely. This software can be in the form of services used by a local application (defined as *Web services*) or a remote application observed through a Web browser. One example of a remote application service is Google Apps, which provides several enterprise applications through a standard Web browser. Remotely executing applications commonly rely on an application server to expose needed services. An application server is a software framework that exposes APIs for software services.

2.4.4.2 Platform as a Service (SaaS)

PaaS can be described as an entire virtualized platform that includes one or more servers (virtualized over the set of physical servers), operating systems, and specific applications (such as Apache and MySQL for Web-based applications). In some cases, these platforms can be predefined and selected; in others, you can provide a VM image that contains all the necessary user-specific applications.

Types:

- **Add-on development facilities:** These facilities allow customization of existing SaaS applications, and in some ways is the equivalent of macro language customization facilities provided with packaged software applications such as Lotus Notes, or Microsoft Word. Often these require PaaS developers and their users to purchase subscriptions to the co-resident SaaS application.

- **Stand alone development environments:** Stand-alone PaaS environments do not include technical, licensing or financial dependencies on specific SaaS applications or web services, and are intended to provide a generalized development environment.

- **Application delivery-only environments:** Some PaaS offerings lack development, debugging and test capabilities, and provide only hosting-level services such as security and on-demand scalability.

- **Open Platform as a Service:** Lets the developer use any programming language, any database, any operating system, any server, etc.

As we are not dealing with SaaS and Paas in this project we limit the discussion till here. We will see Iaas in detail next.

2.4.4.3 Infrastructure as a Service (SaaS)

IaaS is the delivery of computer infrastructure as a service. This layer differs

from PaaS in that the virtual hardware is provided without a software stack. Instead, we have to provide a VM image that is invoked on one or more virtualized servers. IaaS provides and maintains the underlying hardware, operating system and network Infrastructure resources and provides it in a virtualized, easy to manage commoditized way.

IaaS is the unrefined form of "C*omputing as a service*" or also called as "*Hardware as a service*" as in our cloud we are providing the hardware space for instance to run and perform other operations.

Here we have built a Eucalyptus cloud, where we provide VM (operating system as of now), so that the user can access the deployed OS images in the cloud. As a provider, we give hardware space and computing power to user. Here eucalyptus is an open source implementation of Amazon EC2 that is interface-compatible with the commercial service. Like EC2, Eucalyptus relies on Linux (Suse 11.2 here) with Xen (v 3.4 here) for operating system virtualization.

2.4.5 Essential Cloud Characteristics

Here we list some of the major essential characteristics of a cloud:

a) On-demand self-service

The self-service cloud is a powerful IT service delivery model that significantly reduces administrative overhead while driving end-user productivity and satisfaction. Services are delivered on demand or reserved in the future for end users in a completely self-service manner. As a part of this project we have developed on demand (based on availability of instance slot) service to access the cloud. Here the admin can assign slot based on user request. To make it more advanced cloud, the future work would be making reservations to the user, so that the cloud becomes more easy way to access.

b) Ubiquitous network access

Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).

c) Resource pooling

The provider's computing resources are pooled to serve all consumers using a multitenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. The customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction.

d) Rapid elasticity

Capabilities can be rapidly and elastically provisioned to quickly scale up, and rapidly released to quickly scale down based on demand of requirement.

e) Measured service

Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported providing transparency of the utilized service for both the provider and user.

2.4.6 Benefits and Risk of Cloud Computing

Here are some benefits of cloud computing:

- **Reduced Cost**

  Cloud technology is paid incrementally, saving organizations money.
- **Increased Storage**

  Organizations can store more data than on private computer systems.
- **Highly Automated**

  No longer do IT personnel need to worry about keeping software up to date.
- **Flexibility**

  Cloud computing offers much more flexibility than past computing methods.
- **More Mobility**

  Employees can access information wherever they are, rather than having to remain at their desks.

- **Allows IT to Shift Focus**

  No longer having to worry about constant server updates and other computing issues, government organizations will be free to concentrate on innovation.

Following are some risks in cloud computing

1. **Privileged user access.**

   Sensitive data processed outside the enterprise brings with it an inherent level of risk, because outsourced services bypass the "physical, logical and personnel controls" IT shops exert over in-house programs. Get as much information as you can about the people who manage your data. "Ask providers to supply specific information on the hiring and oversight of privileged administrators, and the controls over their access," --Gartner.

2. **Regulatory compliance.**

   Customers are ultimately responsible for the security and integrity of their own data, even when it is held by a service provider. Traditional service providers are subjected to external audits and security certifications. Cloud computing providers who refuse to undergo this analysis are "signaling that customers can only use them for the most trivial functions," --Gartner.

3. **Data location**.

   When we use the cloud, we probably won't know exactly where our data is hosted. In fact, we might not even know what country it will be stored in. Ask providers if they will commit to storing and processing data in specific jurisdictions, and whether they will make a contractual commitment to obey local privacy requirements on behalf of their customers, --Gartner.

4. **Data segregation.**

   Data in the cloud is typically in a shared environment alongside data from other customers. Encryption is effective but isn't a cure-all. The cloud provider should provide evidence that encryption schemes were designed and

tested by experienced specialists. "Encryption accidents can make data totally unusable, and even normal encryption can complicate availability," --Gartner

5. **Recovery.**

Even if you don't know where your data is, a cloud provider should tell you what will happen to your data and service in case of a disaster. "Any offering that does not replicate the data and application infrastructure across multiple sites is vulnerable to a total failure," Gartner says. Ask your provider if it has "the ability to do a complete restoration, and how long it will take."

6. **Investigative support.**

Investigating inappropriate or illegal activity may be impossible in cloud computing. Cloud services are especially difficult to investigate, because logging and data for multiple customers may be co-located and may also be spread across an ever-changing set of hosts and data centers. If you cannot get a contractual commitment to support specific forms of investigation, along with evidence that the vendor has already successfully supported such activities, then our only safe assumption is that investigation and discovery requests will be impossible." –Gartner

7. **Long-term viability.**

Ideally, your cloud computing provider will never go broke or get acquired and swallowed up by a larger company. But you must be sure your data will remain available even after such an event. "Ask potential providers how you would get your data back and if it would be in a format that you could import into a replacement application," --Gartner

Chapter 3

Eucalyptus Cloud

3.1 What is Eucalyptus Cloud?

Eucalyptus is an open source software framework for cloud computing that implements what is commonly referred to as Infrastructure as a Service (IaaS). The Eucalyptus Private Cloud (EPC), via the Eucalyptus software, provides users with the ability to run and control isolated collections of virtual machine instances with many EC2/Hybridfox-compatible tools.

Eucalyptus is abbreviated as*:*

Elastic **U**tility **C**omputing **A**rchitecture **L**inking **Y**our **P**rograms **T**o **U**seful **S**ystems

3.2 Eucalyptus Cloud Architecture

The architecture of the EUCALYPTUS system is simple, flexible and modular with a hierarchical design reflecting common resource environments found in many academic settings. In essence, the system allows users to start, control, access, and terminate entire virtual machines using an emulation of Amazon EC2's SOAP and "Query" interfaces. That is, users of EUCALYPTUS interact with the system using the exact same tools and interfaces that they use to interact with Amazon EC2. Currently, the eucalyptus cloud support VMs that run atop the Xen hypervisor (v 3.4 we used). It has implemented each high-level system component as a stand-alone Web service. This has the following benefits:

- Each Web service exposes a well-defined language-agnostic API in the form of a WSDL document containing both operations that the service can perform and input/output data structures.

- We can leverage existing Web-service features such as WSSecurity policies for secure communication between components. There are four high-level

components, each with its own Web-service interface, that comprise a EUCALYPTUS installation:

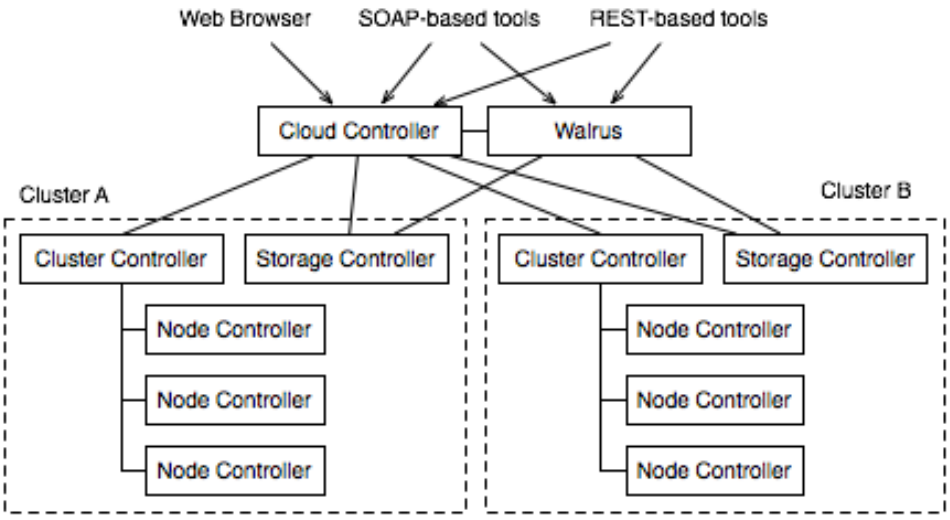


Figure 3.1 Eucalyptus cloud architecture [2]

- **Node Controller** controls the execution, inspection, and terminating of VM instances on the host where it runs.

- **Cluster Controller** gathers information about and schedules VM execution on specific node controllers, as well as manages virtual instance network.

- **Storage Controller** (Walrus) is a put/get storage service that implements Amazon's S3 interface, providing a mechanism for storing and accessing virtual machine images and user data.

- **Cloud Controller** is the entry-point into the cloud for users and administrators. It queries node managers for information about resources, makes high-level scheduling decisions, and implements them by making requests to cluster controllers. The relationships and deployment locations of each component within a typical small cluster setting are shown in *Figure 3.1*.

3.3 Components of Eucalyptus Cloud

3.3.1 Node Controller

A Node Controller (NC) executes on every node that is designated for hosting VM instances. An NC queries and controls the system software on its node (i.e., the host operating system and the hypervisor) in response to queries and control requests from its Cluster Controller. An NC makes queries to discover the node's physical resources – the number of cores, the size of memory, the available disk space – as well as to learn about the state of VM instances on the node (although an NC keeps track of the instances that it controls, instances may be started and stopped through mechanisms beyond NC's control). The information thus collected is propagated up to the Cluster Controller in responses to describe Resource and describe-Instances requests [2].

Cluster Controllers control VM instances on a node by making run- Instance and terminate-Instance requests to the node's NC. Upon verifying the authorization – e.g., only the owner of an instance or an administrator is allowed to terminate it – and after confirming resource availability, the NC executes the request with the assistance of the hypervisor. To start an instance, the NC makes a node-local copy of the instance image files (the kernel, the root file system, and the ramdisk image), either from a remote image repository or from the local cache, creates a new endpoint in the virtual network overlay, and instructs the hypervisor to boot the instance. To stop an instance, the NC instructs the hypervisor to terminate the VM, tears down the virtual network endpoint, and cleans up the files associated with the instance (the root file system is not preserved after the instance terminates) [2].

In short we can say that NC is responsible for Host, start up, inspection, and termination of instance (s).

3.3.2 Cluster Controller

The Cluster Controller (CC) generally executes on a cluster front-end machine, or any machine that has network connectivity to both the nodes running NCs

and to the machine running the Cloud Controller (CLC). Many of the CC's operations are similar to the NC's operations

but are generally plural instead of singular (e.g. runInstances, describeInstances, terminateInstances, describeResources). CC has three primary functions: schedule incoming instance run requests to specific NCs, control the instance virtual network overlay, and gather/report information about a set of NCs. When a CC receives a set of instances to run, it contacts each NC component through its describeResource operation and sends the runInstances request to the first NC that has enough free resources to host the instance. When a CC receives a describeResources request, it also receives a list of resource characteristics (cores, memory, and disk we can see these results at last of this thesis) describing the resource requirements needed by an instance (termed a VM "type"). With this information, the CC calculates how many simultaneous instances of the specific "type" can execute on its collection of NCs and reports that number back to the CLC. As we setup a low configuration cloud system. A maximum of 5 users can use the instances at a time[2].

In short we can say that CC is responsible for

- Gathering state information from its collection of NCs.
- Schedules (Round Robin) incoming VM instance execution request to individual NCs.

3.3.3 Storage Controller

EUCALYPTUS includes Walrus, a data storage service that controls standard web services technologies (Axis2, Mule) and is interface compatible with Amazon's Simple Storage Service (S3) [6]. Walrus implements the REST (via HTTP), sometimes termed the "Query" interface, as well as the SOAP interfaces that are compatible with S3. Walrus provides two types of functionality.

- Users that have access to EUCALYPTUS can use Walrus to stream data into/out of the cloud as well as from instances that they have started on nodes.

- And also Walrus acts as a storage service for VM images. Root file system as well as kernel and ramdisk images used to instantiate VMs on nodes can be

uploaded to Walrus and accessed from nodes. When the instances are made to run the temporary location for these instances would in the NC which we have chosen.

3.3.4 Cloud Controller

The underlying virtualized resources that comprise a EUCALYPTUS cloud are exposed and managed by, the Cloud Controller (CLC). The CLC is a collection of web services which are best grouped by their roles into three categories:

- **Resource Services** perform system-wide arbitration of resource allocations, let users manipulate properties of the virtual machines and networks, and monitor both system components and virtual resources.

- **Data Services** govern persistent user and system data and provide for a configurable user environment for formulating resource allocation request properties.

- **Interface Services** present user-visible interfaces, handling authentication & protocol translation, and expose system management tools providing.

In short we can say that CLC is:

- User visible entry point.
- Responsible for user request.
- Processing service-level agreements (SLAs)

3.4 Features of Eucalyptus Cloud

Following are some features which made eucalyptus cloud more usable.

- Deployment on multiple clusters.
- Deployment of components (Cloud controller, Walrus, Storage Controller, Cluster Controller) on different machines.
- Enhanced maintenance support: components are now "crash consistent," maintaining state across process restart or machine crash.

- Dynamic DNS support: instances and buckets can now be accessed via hostnames. As we have tried and succeeded using Hybridfox.
- Enhanced concurrency management: cloud requests are serviced asynchronously with minimal locking using eventual consistency for scale
- Support for open-source monitoring and health/status: Ganglia and Nagios interaction can also be added to clusters.
- Re-engineered front-end Web-services stack to be faster and more robust as we can compare v1.5 to v1.6.
- Networking improvements, including multi-cluster support. Can be added very easily
- Improved hypervisor support
- Improved startup
- Web UI improvements including ability to select from themes (at compile time). We have modified a theme changing the cloud name very easily.
- Building and installation improvements. Installation when compared to v1.5, v1.6 has been improved a lot.

Chapter 4

Data Transfer Protocols

4.1 File Transfer Protocol (FTP)

File Transfer Protocol (**FTP**) is a standard network protocol used to copy a file from one host to another over a TCP/IP-based network, such as the Internet. FTP is built on client-server architecture and utilizes separate control and data connections between the client and server applications which solve the problem of different end host configurations (i.e. Operating System, file names). FTP is used with user-based password authentication or with anonymous user access.

Standard connection model is shown below.

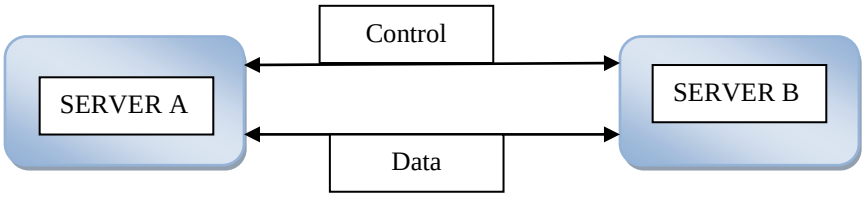


Figure 4.1 Basic connections.

4.1.1 Features & Characteristics for FTP

The objectives of FTP are

1. to promote sharing of files (computer programs and/or data),
2. to encourage indirect or implicit (via programs) use of remote computers,
3. to shield a user from variations in file storage systems among hosts, and
4. to transfer data reliably and efficiently.

FTP, though usable directly by a user at a terminal, is designed mainly for use by programs [8].

Here unlike FTP, which transfers data between client-server, we made an attempt to transfer data between servers. i.e.; from one cloud server to another cloud server.

4.2 Secure Copy Protocol (SCP)

Secure Copy or **SCP** is a means of securely transferring computer files between a local and a remote host or between two remote hosts. It is based on the Secure Shell (SSH) protocol.

The term SCP can refer to one of two related things, the **SCP protocol** or the **SCP program**.

The **SCP protocol** is a network protocol that supports file transfers. The SCP protocol, which runs on port 22, is based on the BSD RCP protocol which is channeled through the Secure Shell (SSH) protocol.

RCP stands for the Unix *'remote copy'* command. It is a command on the Unix operating systems that is used to remotely copy—to copy one or more files from one computer system to another.

SCP might not even be considered a protocol itself, but merely a combination of RCP and SSH. The RCP protocol performs the file transfer and the SSH protocol performs authentication and encryption. [9]

4.2.1 Features & Characteristics for SCP

1.Provides encryption and authentication through SSH protocol.

2.Reliable file transfer over TCP/IP

3.SCP protects the authenticity and confidentiality of the data in transfer.

Basic command to transfer data from one host to other:

$: scp <Data 1> <Data 2>...<Data n> username@ipaddress:path

Data1, Data2 etc is the data which has to be transferred from one machine to other.

To know things well, FTP and SCP are single stream data transfer protocols.

4.3 Grid File Transfer Protocol (Gird FTP)

GridFTP is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. The GridFTP protocol is based on FTP, the highly-popular Internet file transfer protocol. GridFTP Working Group organized under the Global Grid Forum coordinates developments of the server and client implementations of the protocol.

GridFTP is the protocol proposed for all data transfers in Grid environments. It extends the standard FTP protocol with facilities such as multistreamed transfer; auto tuning and Globus based security.

Data-intensive scientific and engineering applications require both the transfer of large amounts of data (terabytes) between geographically distributed storage systems and remote access to large datasets for user application. There are already a number of storage systems in use by the Grid community, each of which was designed to satisfy specific needs and requirements for storing, transferring and accessing large datasets.

Unfortunately, most of these storage systems use incompatible and often unpublished protocols and therefore require the use of their own client libraries for data access. These protocols and client libraries effectively partition the datasets available on the grid and applications that require access to data stored in different storage systems must use multiple access methods. To overcome these incompatible protocols, a universal grid data transfer and access protocol, *GridFTP* was developed.

GridFTP extends the standard FTP protocol and provides a superset of the features that are offered by the various grid storage systems currently in use. Storage providers gain a broader user base because their data is available to any client, while storage users gain access to a broader range of storage systems and data.

4.3.1 How can GridFTP be used for file transfer?

In order to use GridFTP for file transfer, one needs a GridFTP client program that provides the interface between the user and a GridFTP server. There are several clients available for GridFTP, one of which is *globus-url-copy*, a command line tool which can transfer files using the GridFTP protocol as well as other protocols such as

http, udt and ftp. *globus-url-copy* is distributed with the Globus Toolkit and usually available on machines that have the Globus Toolkit installed.

globus-url-copy syntax :

The basic syntax of the globus-url-copy command is:

globus-url-copy [options] sourceURL destinationURL

Where the arguments are described in Table 4.1

Argument	Description
[options]	The optional command line switches as described in sub section 2.1.2.
sourceURL	The URL of the file(s) to be copied. If it is a directory, it must end with a slash (/), and all files within that directory will be copied.
destURL	The URL to which to copy the file(s). To copy several files to one destination URL, destURL must be a directory and be terminated with a slash (/).

Table 4.1: Arguments of globus-url-copy

4.3.2 Features of GridFTP

Grid Security Infrastructure (GSI)

Robust and flexible authentication, integrity, and confidentiality features are critical when transferring or accessing files.

Parallel data transfer

On wide-area links, the use of multiple TCP streams in parallel (even between the same source and destination) improves the aggregate bandwidth over using a single TCP stream. GridFTP supports parallel data transfer through FTP command extensions and data channel extensions.

Striped data transfer

Data may be striped or interleaved across multiple servers. GridFTP includes extensions that initiate striped transfers, which use multiple TCP

streams to transfer data that is partitioned among multiple servers. Striped transfers provide further bandwidth improvements over those achieved with parallel transfers. We have defined GridFTP protocol extensions that support striped data transfers.

Partial file transfer

Many applications require the transfer of portions rather than complete files. Standard FTP supports the transfer of complete files or the transfer of the remainder of a file starting at a particular offset. GridFTP introduces new FTP commands to support transfers of subsets or regions of a file.

Support for reliable and restartable data transfers

Reliable transfer is essential for many applications and GridFTP incorporates fault tolerant features to handle transient network failures, server outages, etc. The FTP standard includes a basic feature for restarting failed transfers these are not widely implemented. The GridFTP protocol exploits these features and extends them to cover the new data channel protocol.

4.4 Udp Based Data Transfer Protocol (UDT)

UDT [10] is an application-level data transport protocol that uses UDP to transfer bulk data, while implementing its own reliability and congestion control mechanisms. UDT is built on top of UDP. UDT achieves good performance on high-bandwidth, high delay networks in which TCP has significant limitations. UDT uses UDP packets to transfer data and retransmit the lost packets to guarantee reliability. UDT's congestion control algorithm combines rate based and window based approaches to tune the inter-packet time and the window size, respectively. The congestion control parameters (window size and inter-packet time) are updated dynamically by a bandwidth estimation technique. UDT is popular among the

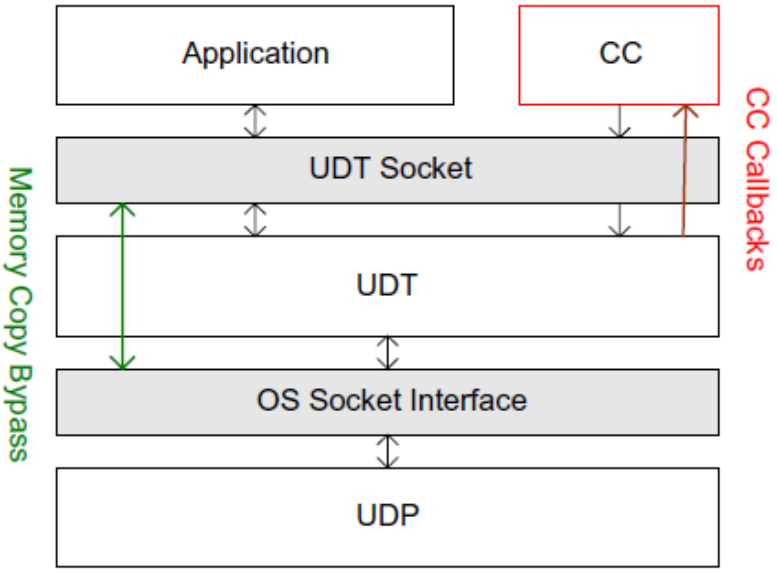


Figure 4.2 UDT architceture

application level solutions to address TCP limitations and it has been shown to be friendly to short-lived TCP flows [10].

In order to support convenient connection, UDT provides the rendezvous connection setup mode, in which there is no server or client, and two users can connect to each other directly. Inside the UDT implementation of the rendezvous connection setup, each UDT socket sends a connection request to its peer side, and whoever receives the request will then send back a response and set up the connection. If one of the connected UDT entities is being closed, it will send a shutdown message to the peer side. The peer side, after receiving this message, will also be closed. This shutdown message, delivered using UDP, is only sent once and not guaranteed to be received. If the message is not received, the peer side will be closed by a timeout mechanism (through the keep-alive packets). Keep-alive packets are generated periodically if there is no other data or control packets sent to the peer side.[10]

4.4.1 Globus XIO Driver for UDT

GridFTP uses the *Globus Extensible Input Output* (XIO) [3] interface to invoke network I/O operations. The Globus XIO framework presents a single, standard open/close, read/write interface to many different protocol implementations, including TCP, UDP, HTTP, and UDT. The protocol implementations are called drivers. Once created, a driver can be dynamically loaded and stacked by any Globus

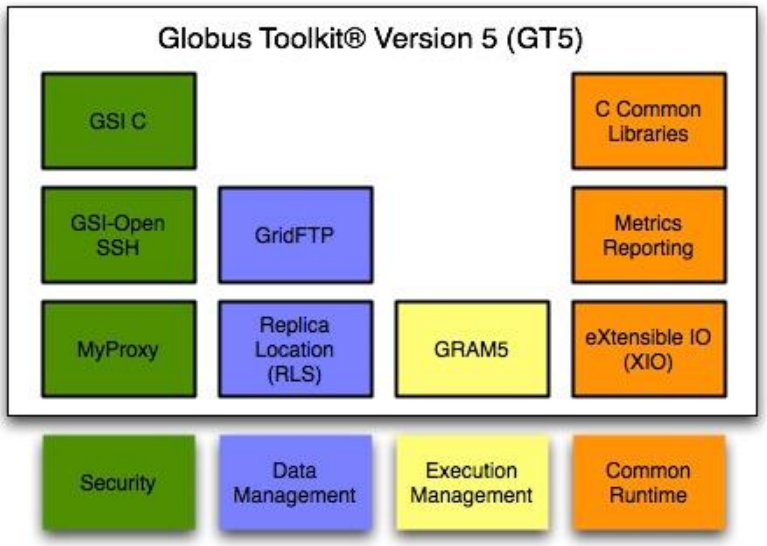


Figure 4.3 Globus Toolkit

XIO application. Many drivers have been created by using the native Globus XIO assistance APIs, including TCP, UDP, HTTP, File, Mode E, Telnet, Queuing, Ordering, GSI, and Multicast Transport. For other evolving protocols, if an implementation already exists, requiring the developers to implement their protocols by using native Globus XIO APIs would necessitate considerable effort.

A simple extension to the existing Globus has been added for UDT to work in Globus.

4.4.2 Characteristics for UDT

There are several sets of characteristics for UDT:

- UDT works in application layer. [12]
- It is a multi stream transport protocol
- A UDT entity can detect a broken connection if it does not receive any packets in a certain predetermined time
- UDT uses timer-based selective acknowledgment.
- UDT is bi-directional.
- UDT provides a convenient rendezvous connection setup to traverse firewalls. In rendezvous setup, both peers connect to each other's known port at the same time.

Chapter 5

System Design

5.1 Proposed Framework

The framework which is been proposed for data transfer is shown below.

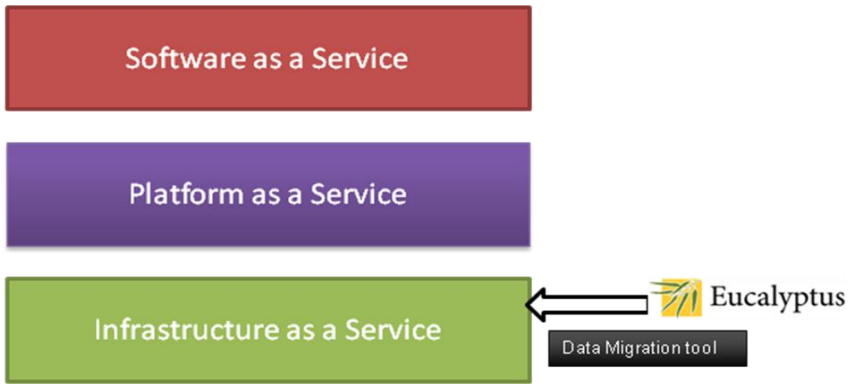


Figure 5.1 Framework proposed

As we can see the data migration tool is placed in the IaaS slot. As the data transfer from one cloud to other cloud is maintain by administrator. Data migraition tools is not a sevice, admin just provides a interface for this. Hence it fits at IaaS.

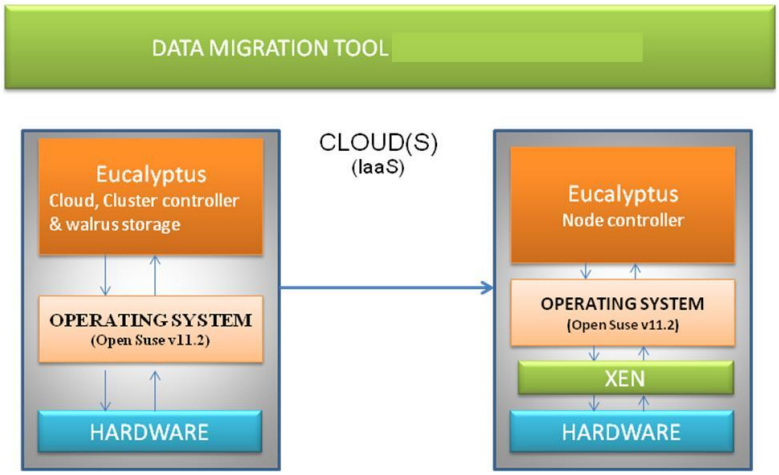


Figure 5.2 Framework and slot for data migration tool

M1

M2

5.2 Architecture for Eucalyptus Private Cloud

Figure 5.2 shows the internal architecture of eucalyptus cloud. As we can see Machine 1 (M1), we have configured the linux-suse operating system and on top of it we have Eucalyptus cloud and cluster controllers. And on Machine 2 (M2), we have an Hypervisor XEN, which resides over hard disk and carries the linux-suse operating system and on top of it we have configured the node controllers for the cloud.

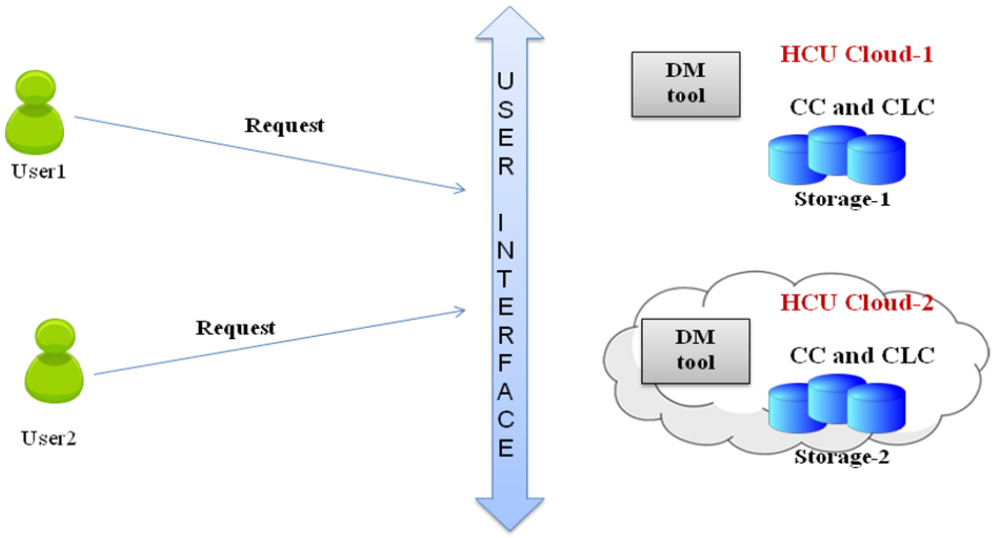


Figure 5.3 Proposed Architecture

As you can see the user will request the cloud admin for the data transfer, Admin transfers data from one cloud to other based on user request.

5.3 Scope

The scope of the project is as follows:

- Limited, within LAN.
- This cloud is a private cloud as it is made available with-in the campus only.
- A maximum of 5 users can access the instances which are available.
- Architecture is limited to two CLC`s, CC`s and two NCs, for the clouds.
- OS instances can be accessed using open source plug-in Hybridfox.
- Basic kernel OS is accessed in the cloud, by the user.

5.4 Software Requirement Specification

The following software packages are required to successfully implement the cloud and share the resources:

- Open Suse 11.2
- Apache2
- Apache2-prefork
- Java-1_6_0-openjdk
- Java-1_6_0-openjdk-devel
- Libvirt
- Curl
- Vlan
- Dhcp-server bridge-utils
- Xen
- Axis2 ,Axis2c,
- Hibernate 3.2.2,
- HSQLDB 1.8.0,
- jetty 6.1.9,
- JiBX,
- Mule 2.0.1, Rampart version 1.3,
- socat-1.6.0,VDE version 2.2.0-pre2

And to implement GridFTP the following packages are required:

- Globus 4.2.1
- OpenSSL
- Java 1.6
- Apache ant 1.6.4
- GCC compiler
- Tar, sed, make
- Perl
- PostgreSQL
- Libiodbc2 ,Libiodbc2-dev

Hardware Requirements:

The following are the hard ware requirements which are entitled to build the cloud:

- 4 GB RAM (on all 6 machines)
- Core2 vPRO processor (on all 6 machines)
- 250 GB hard disk (on all 6 machines)

5.5 Prerequisites and How Eucalyptus Private Cloud Developed

If you start with a standard Open Suse v11.2 installation, you will satisfy all prerequisites with the following steps:

1. **Apache2**

   The **Apache HTTP Server**, commonly referred to as **Apache** is web server software

2. **Apache2-prefork**

   This Multi-Processing Module (MPM) implements a non-threaded, pre-forking web server that handles requests in a manner similar to Apache 1.3. It is appropriate for sites that need to avoid threading for compatibility with non-thread-safe libraries. It is also the best MPM for isolating each request, so that a problem with a single request will not affect any other.

3. **Java-1_6_0-openjdk**

   The runtime environment (JRE). It's necessary for running Java programs.

4. **Java-1_6_0-openjdk-devel**

   The development kit (JDK). This is necessary for building of Java applications and also for JSP servlet containers, like Apache Tomcat.

5. **Libvirt**

   It is an open source API and management tool for managing platform virtualization. It is used to manage Linux KVM and Xen virtual machines through graphical interfaces such as Virtual Machine Manager and higher level tools. libvirt itself is a C library, but it has bindings in other languages, notably in Python and Perl.

6. **Curl**

   cURL is a computer software project providing a library and command-line tool for transferring data using various protocols. The cURL project produces two products, **libcurl** and **cURL.**

7. Vlan

A **virtual LAN**, commonly known as a **VLAN**, is a group of hosts with a common set of requirements that communicate as if they were attached to the same broadcast domain, regardless of their physical location. A VLAN has the same attributes as a physical LAN, but it allows for end stations to be grouped together even if they are not located on the same network switch. Network reconfiguration can be done through software instead of physically relocating devices.

8. Dhcp-server

The **Dynamic Host Configuration Protocol** (**DHCP**) is a computer networking protocol used by hosts (*DHCP clients*) to retrieve IP address assignments and other configuration information.

9. Bridge-utils

Utilities for configuring the Linux 2.4 bridge. This package contains utilities for configuring the Linux ethernet bridge. The Linux ethernet bridge can be used for connecting multiple ethernet devices together. The connecting is fully transparent: hosts connected to one ethernet device see hosts connected to the other ethernet devices directly.

10. Xen

Xen is an open-source virtualization solution. The Xen hypervisor acts as a thin layer between the hardware and the operating system, allowing multiple virtual servers to run simultaneously on a single physical server. Each virtual server acts independently of the others, with its own allocated area of RAM and virtual disks.

Many instructions in this guide refer to a *single-cluster installation*, in which all components except NC are co-located on one machine, which we refer to as **front-end**. All other machines, running only NCs, will be referred to as **nodes**. Out of these, 1 to 9 are installed on front end and 1, 5, 7 and 10 are to be installed in back end.

As seen from figure 5.2, we need to make CLC and CC into one machine and NC onto the other machine. Firstly we have to download all the packages which are related to Eucalyptus cloud[1]. Then proceed as follows

```
tar zxvf eucalyptus-$VERSION-*.tar.gz
cd eucalyptus-$VERSION-*
```

In the examples below we use `x86_64`, which should be replaced with `i386` or `i586` on 32-bit architectures. Install the third-party dependency RPMs on the front end:

```
cd eucalyptus-$VERSION*-rpm-deps-x86_64
rpm -Uvh aoetools-21-1.el4.x86_64.rpm \
        euca-axis2c-1.6.0-1.x86_64.rpm \
        euca-rampartc-1.3.0-1.x86_64.rpm \
        vblade-14-1mdv2008.1.x86_64.rpm \
        groovy-1.6.5-1.noarch.rpm \
        vtun-3.0.1-1.x86_64.rpm
cd ..
```

Install the -cloud, -walrus, -cc and -sc RPMs on the front end:

```
cd eucalyptus-$VERSION*-rpm-deps-x86_64
rpm -Uvh aoetools-25-2.49.x86_64.rpm \
        euca-axis2c-1.6.0-1.x86_64.rpm \
        euca-rampartc-1.3.0-1.x86_64.rpm \
        vblade-15-2.49.x86_64.rpm
cd ..
```

On the compute nodes, install the node controller RPM with dependencies:

```
rpm -Uvh eucalyptus-$VERSION-*.x86_64.rpm \
        eucalyptus-gl-$VERSION-*.x86_64.rpm \
        eucalyptus-nc-$VERSION-*.x86_64.rpm
```

[1]

Now start up your Eucalyptus services. On the front-end:

```
/etc/init.d/eucalyptus-cloud start
/etc/init.d/eucalyptus-cc start
```

On the node:

```
/etc/init.d/eucalyptus-nc start
```

5.5.1 First-time Setup

Below document further describes the steps for activating and possibly further configuring Eucalyptus after the software has been installed on all nodes.

Registering Eucalyptus Components

Here we need to register some more components in the cloud so that the cloud functionalities are upto the mark.

$EUCALYPTUS defines the directory path where the eucalyptus cloud is installed, here it's / (root)

```
$EUCALYPTUS/usr/sbin/euca_conf --register-walrus <front end IP address>

$EUCALYPTUS/usr/sbin/euca_conf --register-cluster <clustername> <front end IP address>

$EUCALYPTUS/usr/sbin/euca_conf --register-sc <clustername> <front end IP address>
```

We need to register Walrus, Cluster and storage cluster using the above commands, passing an argument "*Front IP Address*" which is actually the IP address of the cloud.

Finally, you need to register nodes with the front end. To do so, run the following command on the front end,

```
$EUCALYPTUS/usr/sbin/euca_conf --register-nodes "<Node 0 IP address> <Node 1 IP address> ... <Node N IP address>"
```

5.5.2 First-time Configuration

Point your browser to, https://front-end-ip:8443

Front-end-ip here is 172.16.88.4 for cloud 1 and 172.16.92.192 cloud 2

Since Eucalyptus is using a self-signed certificate, your browser is likely to prompt you to accept the certificate. On some machines it may take few minutes after the starting of the Cloud Controller for the URL to be responsive the first time you run Eucalyptus. You will be prompted for a user and password both of which are set to `admin` initially.

Upon logging in the first time you will be asked to

1. Change the admin password,
2. Set the admin's email address, and
3. Confirm the IP of the Cloud Controller host.

After clicking 'Submit', you will see the 'Configuration' tab. Since you've used `euca_conf` to register Walrus and a cluster, they will be listed along with a few configurable parameters. Look over the parameters to see if any need adjustment. For more information, see the Management section.

To use the system with client tools, you must obtain user credentials. From the 'Credentials' tab, Eucalyptus users can obtain two types of credentials: *x509 certificates* and query interface credentials. Use the 'Download Credentials' button to download a zip-file with both or click on the 'Show Keys' to see the query interface credentials. You will be able to use your credentials with Euca2ools, Amazon EC2 tools and third-party tools like rightscale.com. Create a directory to store your credentials, unpack the zip-file into it, and source the included 'eucarc':

```
mkdir $HOME/.euca
unzip euca2-admin-x509.zip -d $HOME/.euca
. $HOME/.euca/eucarc
```

As each time you need to set the environmental variables. Just copy the ‘eucarc’ file in /etc/bash.bashrc file.

Make sure the IP address of the cloud and nodes are static and are properly maintained.

5.5.3 Eucalyptus Network Configuration

Eucalyptus versions 1.5 and higher include a highly configurable VM networking subsystem that can be adapted to a variety of network environments. There are four high level networking "modes", each with its own set of configuration parameters, features, benefits and in some cases restrictions placed on your local network setup. The administrator must select one of these four modes before starting Eucalyptus on the front-end and nodes via modification of the 'eucalyptus.conf' configuration file on each machine running a Eucalyptus component. A brief description of each mode follows:

5.5.3.1 SYSTEM Mode

This is the simplest networking mode, but also offers the smallest number of networking features. In this mode, Eucalyptus simply assigns a random MAC address to the VM instance before booting and attaches the VM instance's ethernet device to the physical ethernet through the node's local Xen bridge. VM instances typically obtain an IP address using DHCP, the same way any non-VM machine using DHCP would obtain an address. Note that in this mode, the Eucalyptus administrator (or the administrator that manages the network to which Eucalyptus components are attached) must set up a DHCP server that has a dynamic pool of IP addresses to hand out as VMs boot.

5.5.3.2 STATIC Mode

This mode offers the Eucalyptus administrator more control over VM IP address assignment. Here, the administrator configures Eucalyptus with a 'map' of MAC address/IP Address pairs. When a VM is instantiated, Eucalyptus sets up a static entry within a Eucalyptus controlled DHCP server, takes the next free MAC/IP pair, assigns it to a VM, and attaches the VMs ethernet device to the physical ethernet through the Xen bridge on the nodes (in a manner similar to SYSTEM mode). This mode is useful for administrators who have a pool of MAC/IP addresses that they wish to always assign to their VMs.

5.5.3.3 MANAGED Mode

This mode is the most featureful of the three modes, but also carries with it the most potential constraints on the setup of the Eucalyptus administrator's network. In MANAGED mode, the Eucalyptus administrator defines a large network (usually private, unroutable) from which VM instances will draw their IP addresses.

Here we use a STATIC mode configuration which is good for the basic cloud.

5.5.4 Managing Eucalyptus Images

First, be sure to source your 'eucarc' file before running the commands below. Note that all users may upload and register images (depending on access granted to them by the Eucalyptus administrator), but only the admin user may ever upload/register kernels or ramdisks.

Second, the instructions below rely on the euca2ools command-line tools distributed by the Eucalyptus Team. Please, install them if you haven't done so already.

5.5.4.1 Adding Images

To enable a VM image as an executable entity, a user/admin must add a root disk image, a kernel/ramdisk pair (ramdisk may be optional) to Walrus and register the uploaded data with Eucalyptus. Each is added to Walrus and registered with Eucalyptus separately, using three EC2 commands. The following example uses the test image that we provide. Unpack it to any directory:

Add the kernel to Walrus, and register it with Eucalyptus (**WARNING**: your bucket names must not end with a slash!):

We now have three things to upload and register, the OS .img file, Ramdisk file and kernel file. Here we shown using ubuntu 9.04 i386 package.

Uploading Image file

```
$euca-bundle-image -i euca-ubuntu-9.04-i386/ubuntu.9-04.x86.img -u 000100729354 -d ubuntu-9.04-i386-image -k .euca/euca2-admin-fff16120-pk.pem -c .euca/euca2-admin-fff16120-cert.pem --ec2cert .euca/cloud-cert.pem

$euca-upload-bundle -b ubuntu-9.04-i386-image -m ubuntu-9.04-i386-image/ubuntu.9-04.x86.img.manifest.xml

$euca-register ubuntu-9.04-i386-image/ubuntu.9-04.x86.img.manifest.xml
```

Uploading Kernel Image file

```
$euca-bundle-image -i euca-ubuntu-9.04-i386/xen-kernel/vmlinuz-2.6.24-19-xen -u 000100729354 -d ubuntu-9.04-i386-kernel -k .euca/euca2-admin-fff16120-pk.pem -c .euca/euca2-admin-fff16120-cert.pem --ec2cert .euca/cloud-cert.pem --kernel true

$euca-upload-bundle -b ubuntu-9.04-i386-kernel -m ubuntu-9.04-i386-kernel/vmlinuz-2.6.24-19-xen.manifest.xml

$euca-register ubuntu-9.04-i386-kernel/vmlinuz-2.6.24-19-xen.manifest.xml
```

Uploading Ramdisk (initrd) Image file

```
$euca-bundle-image -i euca-ubuntu-9.04-i386/xen-kernel/initrd.img-2.6.24-19-xen -u 000100729354 -d ubuntu-9.04-i386-ramdisk -k .euca/euca2-admin-fff16120-pk.pem -c .euca/euca2-admin-fff16120-cert.pem --ec2cert .euca/cloud-cert.pem

$euca-upload-bundle -b ubuntu-9.04-i386-ramdisk -m ubuntu-9.04-i386-ramdisk/initrd.img-2.6.24-19-xen.manifest.xml

$euca-register ubuntu-9.04-i386-ramdisk/initrd.img-2.6.24-19-xen.manifest.xml
```

Associating kernels and ramdisks with instances

There are three ways that one can associate a kernel (and ramdisk) with a VM instance.

1. A user may associate a specific kernel/ramdisk identifier with an image at the 'euca-bundle-image' step

```
euca-bundle-image -i <vm image file> --kernel <eki-XXXXXXXX> --ramdisk <eri-XXXXXXXX>
```

2. A user may choose a specific kernel/ramdisk at instance run time as an option to 'euca-run-instances'

```
euca-run-instances --kernel <eki-XXXXXXXX> --ramdisk <eri-XXXXXXXX> <emi-XXXXXXXX>
```

3. The administrator can set 'default' registered kernel/ramdisk identifiers that will be used if a kernel/ramdisk is unspecified by either of the above options. This is accomplished by logging in to the administrative interface (https://your.cloud.server:8443), clicking on the 'Configuration' tab and adding an <eki-xxxxxxxx> and optionally an <eri-xxxxxxxx> as the defaults kernel/ramdisk to be used.

Deleting Images

In order to delete an image, you must first de-register the image:

```
euca-deregister <emi-XXXXXXXX>
```

5.5.4.2 User Management

A user friendly environment is made with GUI of eucalyptus cloud. Simple steps need to follow just to register into the cloud. User goes to the main page and simply registers his details with the "*apply for account"* button. And start

downloading the credentials and can access the resources which are available in the cloud.

5.6 Resources available in developed Private Cloud

Resources that are been added to the private cloud includes the deployment of Operating system and server images. They are listed below:

- Debian-5.0-x86_64 & i386
- Ubuntu 9.04-x86_64 & i386
- Fedora 11 i386
- Fedora 10-x86_64
- Centos-5.3--x86_64 & i386
- Karmic Server -x86_64 & i386

As these images are of basic configuration, we can get a terminal to access the OS associated with the images. Further we describe how to access these resources.

5.6.1 How resources invoke in the cloud

What actually the Operating System images contain:-

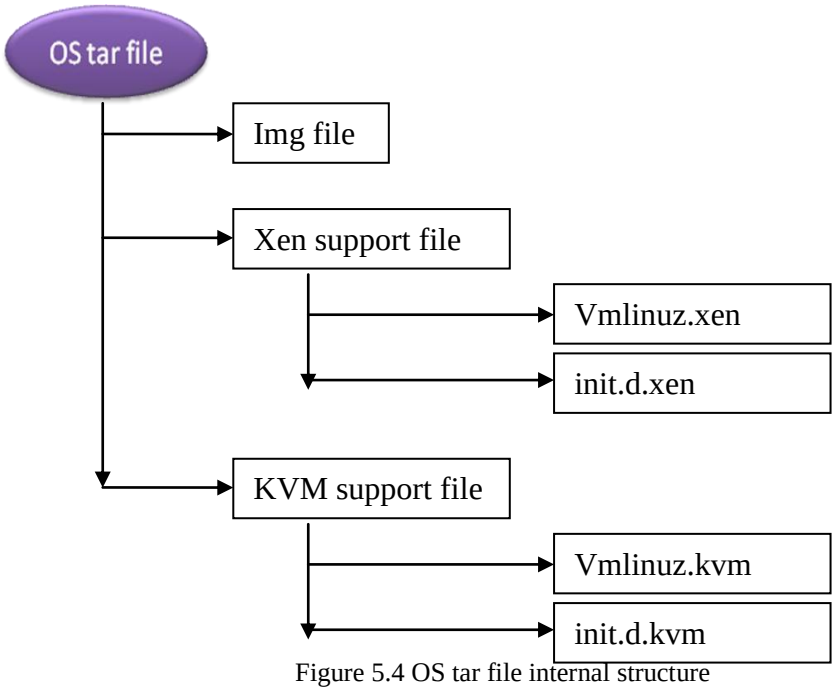


Figure 5.4 OS tar file internal structure

5.6.2 Some more information on Eucalyptus cloud resources in details:-

1. **What is actually an img file used for. What actually it contains?**

   Image is a single file or storage device.

2. **What is the VmLinuz file and init.d file which come along with any operating system image?**

   Vmlinuz is a compressed Linux kernel, and it is bootable. And init.d is the ramdisk file

3. **How actually these get related when they are uploaded in the Walrus?**

   The manifest file of the emi, will hold the information of the associated kernel and the ramdisk.

4. **How actually these files work in the User work space?**

   Init.d file is loaded→control is passes to kernel image (vmlinuz) →kernel is initialized→boot loader checks the system hardware→ mounts the root device →loads the necessary kernel modules. →The first user-space program (init) starts→high-level system initialization is performed.

5. **How these manifest files help to create the environment for the user?**

   Manifest file has the details of list of the image parts with their checksums. Including details of the associated with ramdisk and kernel.

5.7 Accessing Available Resources with Open Source Tools

The OS images which are available on the cloud can be accessed in two ways.

- Using command line arguments. (Euca2ools)
- Using open source GUI application. (Hybridfox)

Steps to how to access these are shown in the screenshots section of this thesis.

5.8 Experiment setup for Data Transfer in Cloud

Experiment setup includes the following setup which is done by us. Figure 5.5 Shows the complete experiment set up.

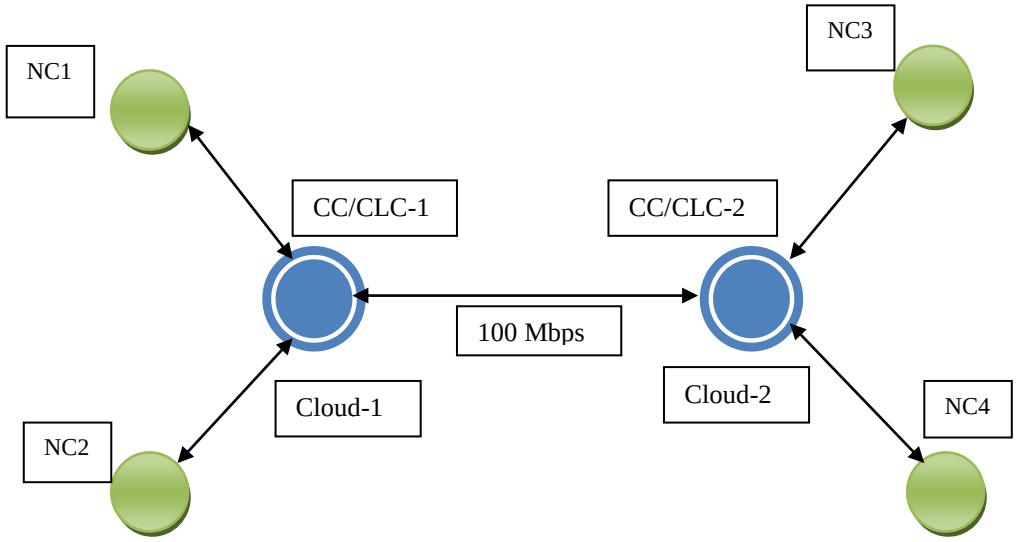


Figure 5.5 experiment setup

We have set up the cloud as shown above. Two CLCs and CCs for each cloud and NC1, NC2, NC3, NC4 for these clouds. A dedicated 100 Mbps LAN connection is maintained between these clouds to perform the experiment.

Here we have partially implemented the tool to check the performances of SCP, FTP, GridFTP and UDT by transferring 1GB data from one cloud to other.

5.9 Performance Evaluation for Different Data Transfer Protocols

5.9.1 Performance Evaluation using GridFTP over UDT and TCP

Throughput evaluation is has to be done using GridFTP and UDT. Unlike FTP and SCP, which are single stream transfer protocols. GridFTP and UDT are multi stream transfer protocols.

Chapter 6

Conclusion & Future Work

6.1 Conclusion

We present the results of how resources like operating systems and servers are been used in the private cloud by users. We have shown couple of methods in this scenario.

We also presented a data migration tool, and tried to show how GridFTP over UDT and GridFTP over TCP will produce the throughputs. We have partially implemented GridFTP over UDT and TCP. Overall, although UDT uses more system resources than TCP, it may achieve significant higher throughput compared to TCP and reduce its rate to accommodate the computing TCP flows. [3]

6.2 Future Work

As the resources, like OS`s here, has basic kernel configuration in the cloud as of now. One can even add the OS with GUI. This can be done by increasing the computing power of the CLC. Furthermore applications like appscale apps can be added to the cloud and can be cataloged into the resource list. As the number of users are low i.e.; five now. One can increase this number, by increasing the computing power of the cloud. Further improving the cloud, as number of users increase with increase in resources. To schedule the slots a brokerage application can also be added with the help of load balancer in the cloud. So that the slots can be assigned to the end user.

On the other hand, as we have developed a data migration tool for SCP, GridFTP over TCP and UDT for file transfer application. Further, one can try to further improve the back end of this tool, so that the data transfer between clouds completes so that performance can be evaluated.

Chapter 7

Implementation Details

7.1 Introduction

This part shows the set of outputs what we have achieved during the process of building cloud, maintaining the cloud, accessing the cloud for the resources. Set of screenshots are been put in to get a clear view of what actually we did in this project.

For data transfer applications, we use SCP, GridFTP over TCP and GridFTP over UDT. We are trying to incorporate these data transfer tool in our migration tool. For GridFTP over TCP or UDT we need to generate the CA certificate between two servers, so that the transfer will be successful. A modified GridFTP is below, which makes use of UDT instead of TCP as follows:

1. Build and install UDT

   globus$ make udt ("make gridftp udt" if gridftp is not built and installedalready)

   globus$ make install

2. Configure GridFTP server

   If you the GridFTP server from xinetd, add '-dc-whitelist udt,gsi,tcp' to 'server_args' in /etc/xinetd.d/gsiftp

   Alternatively, you can use the file $GLOBUS_LOCATION/etc/gridftp.conf to configure this. Add the following to that file:

   dc-whitelist udt,gsi,tcp

   If you run the server from commandline:

   GLOBUS_LOCATION/sbin/globus-gridftp-server -dc-whitelist udt,gsi,tcp

3. Run globus-url-copy with new command line option "-udt"

7.2 Basic Assumptions

Basic assumption what we have taken here is that the end user has all the basic requirements in order to access the resources in the cloud. Basic requirement by the end user is:

- The basic requirement is, the user should apply for an account in the cloud and have the credentials ready.
- If the user wishes to use the resources using command line, one should have the euca2ools 1.2 package to access.
- Should know how to use Hybridfox or any other open source tool, Elasticfox, S3fox etc.
- The user should aware that the OS resource which is been provided, is the basic kernel OS. And should satisfy with the functionalities it has.
- For data transfer, user must know the command lines used to transfer the data from one cloud to other.
- For using the tool, as we have partially implemented. We kept the tool very simple, it`s easy to use.

7.3 Outputs (screenshots)

Figure 7.1 front end https://172.16.92.128:8443

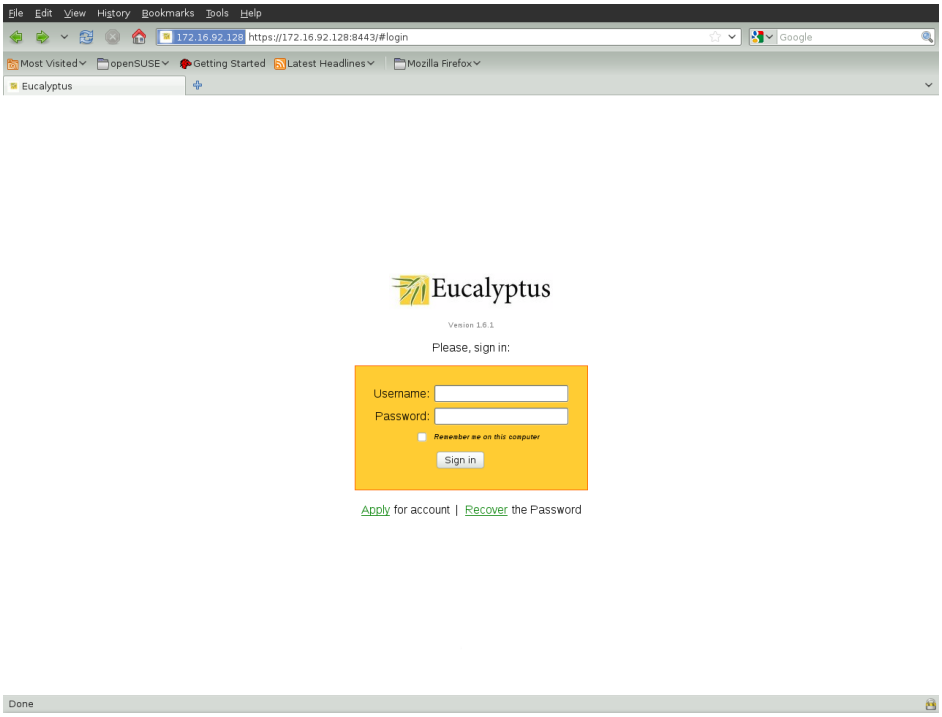


Figure 7.2 User registration

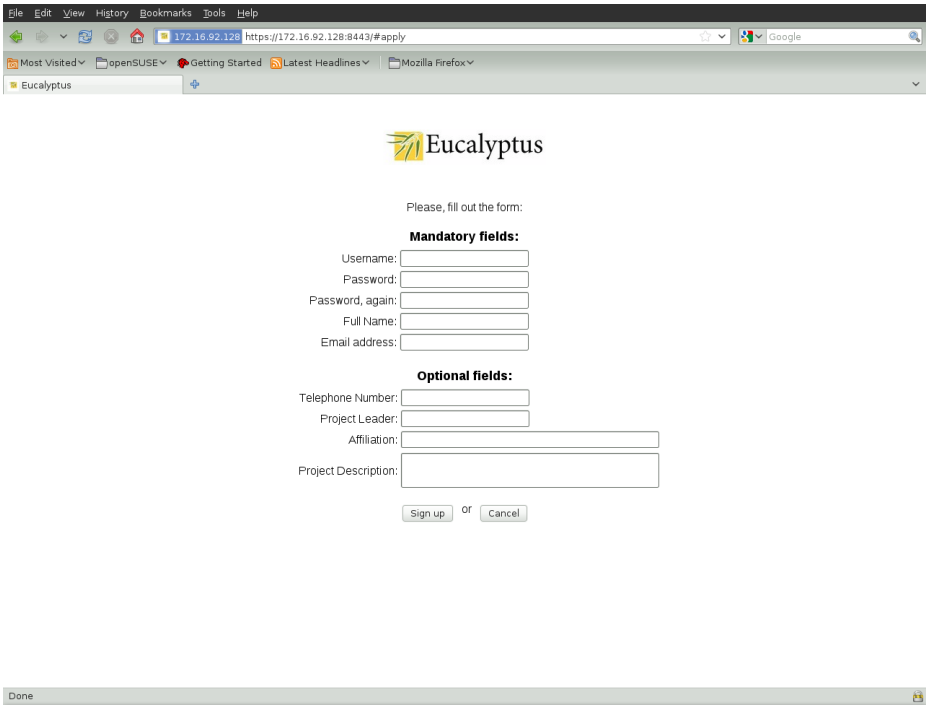


Figure 7.3 Request to approve page

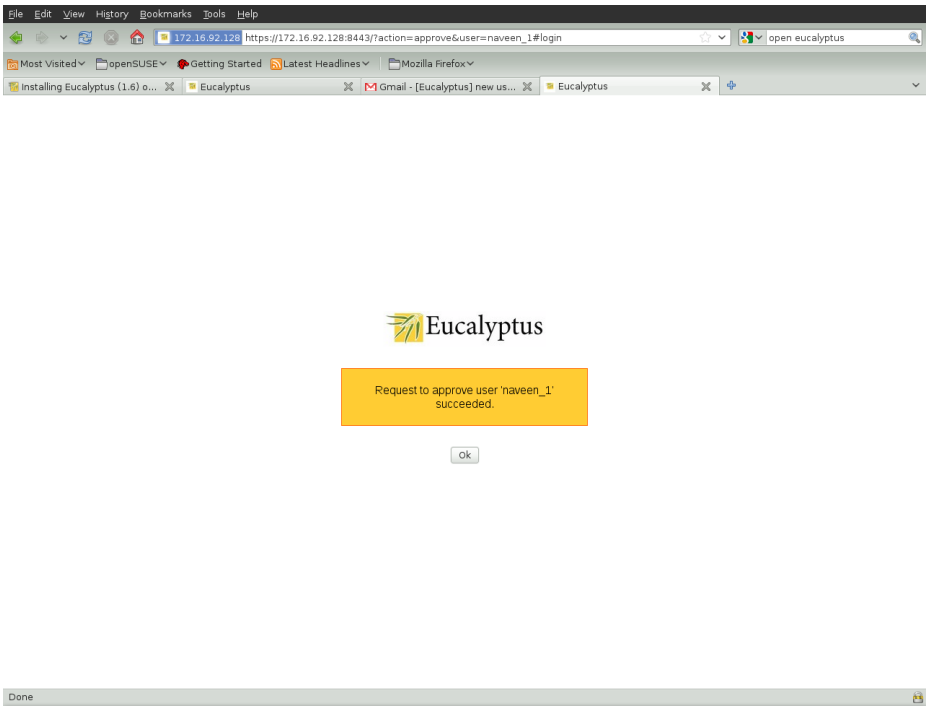


Figure 7.4 Activating account through email.

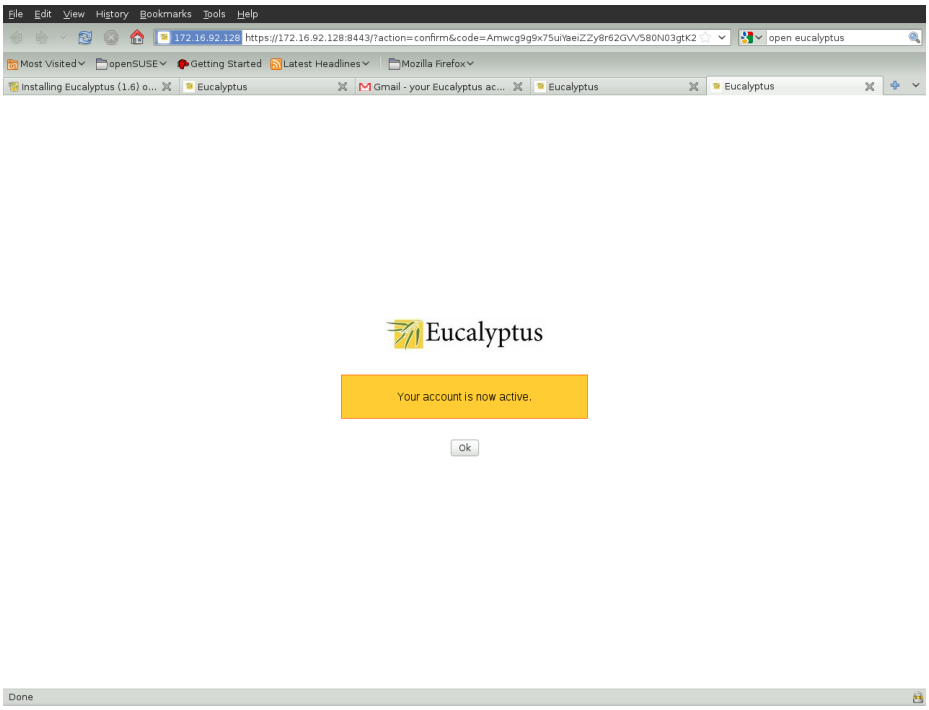


Figure 7.5 Account successful login.

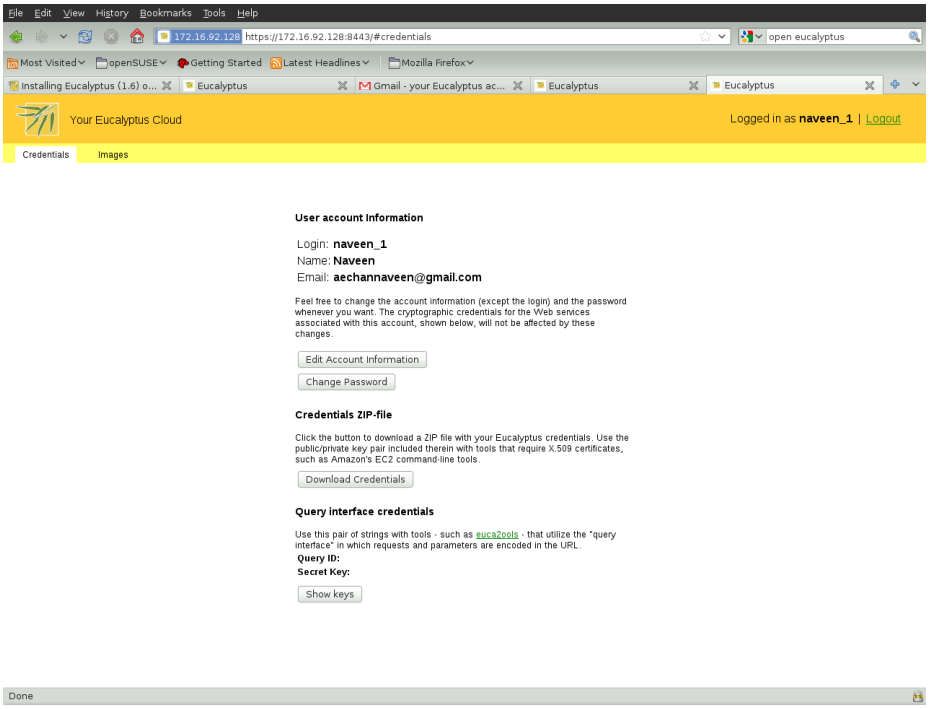


Figure 7.6 Deployed images by admin, for all users in cloud.

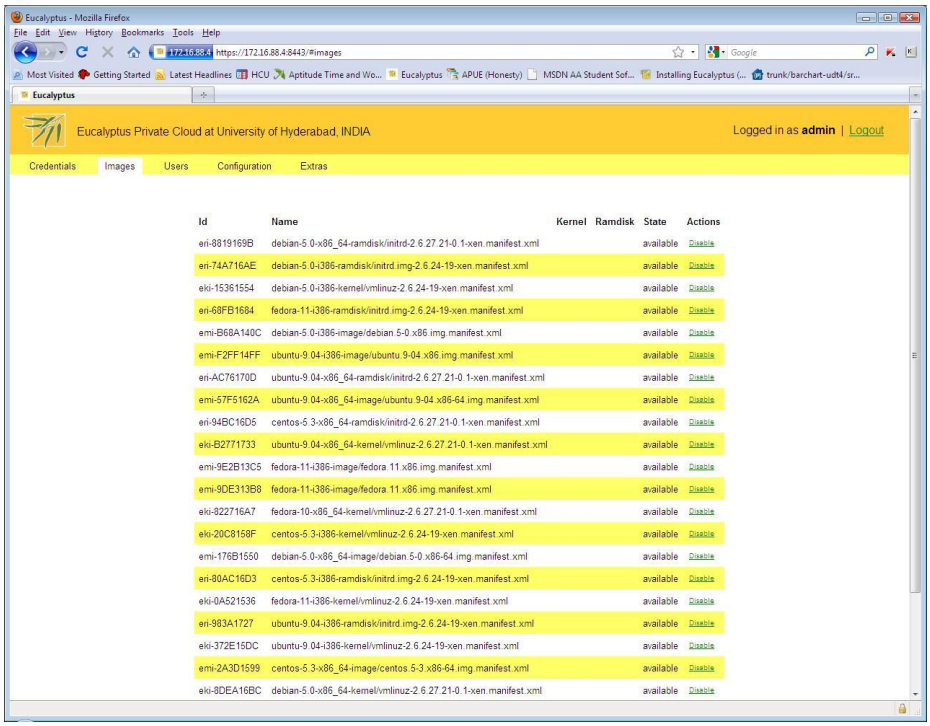


Figure 7.7 Set of commands used in Eucalyptus Cloud

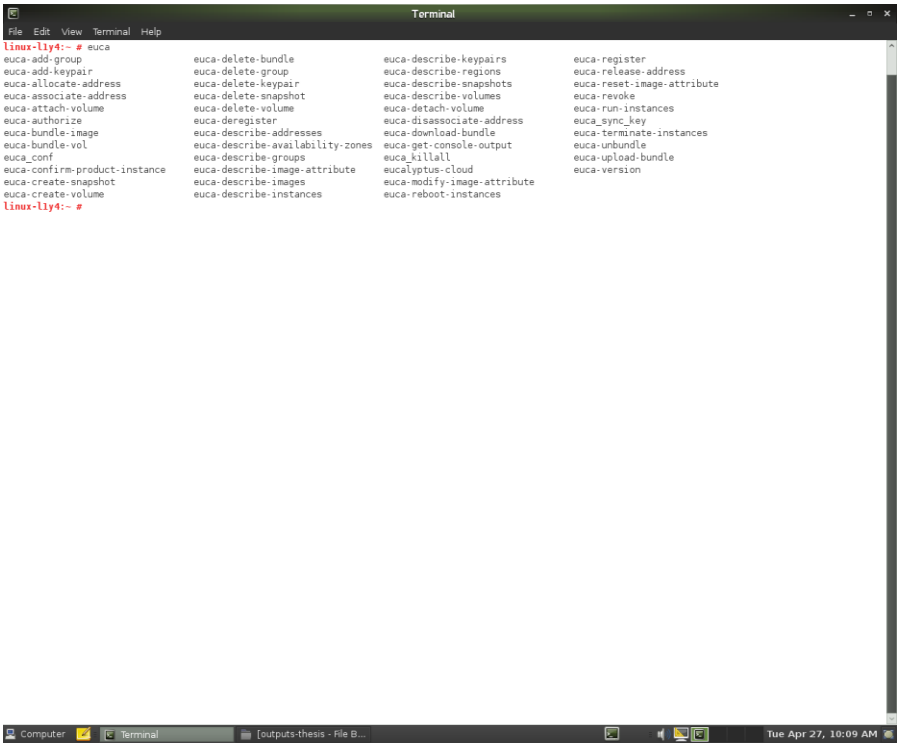


Figure 7.8 Availability Zones, we can see the cluster where the cloud is deployed

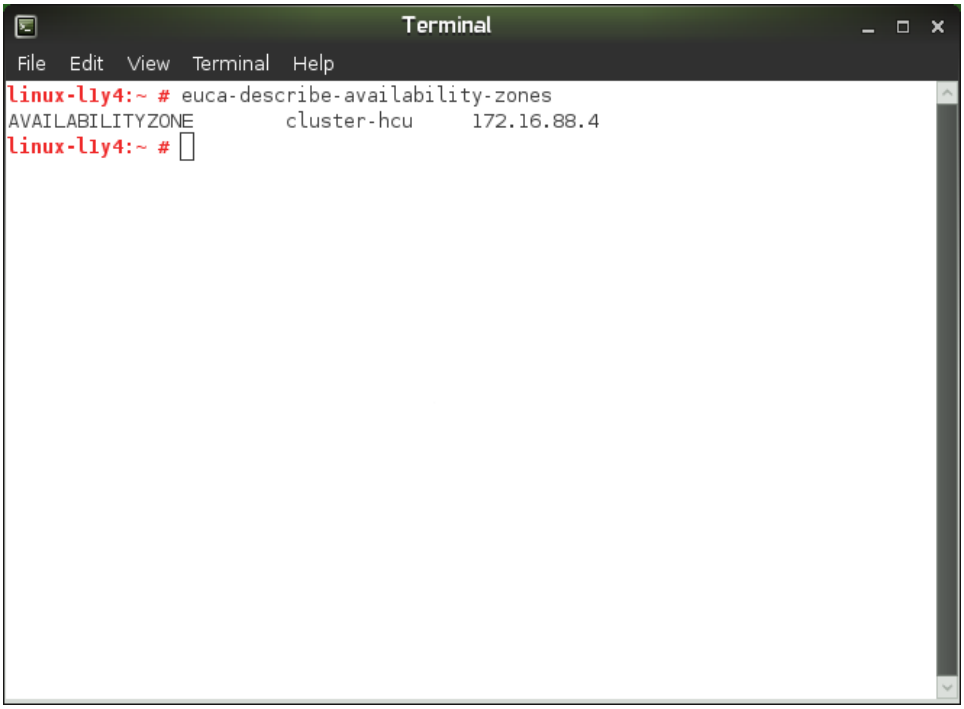


Figure 7.9 we can see the availability zones in brief. The following image shows the list of computing powers which are available.

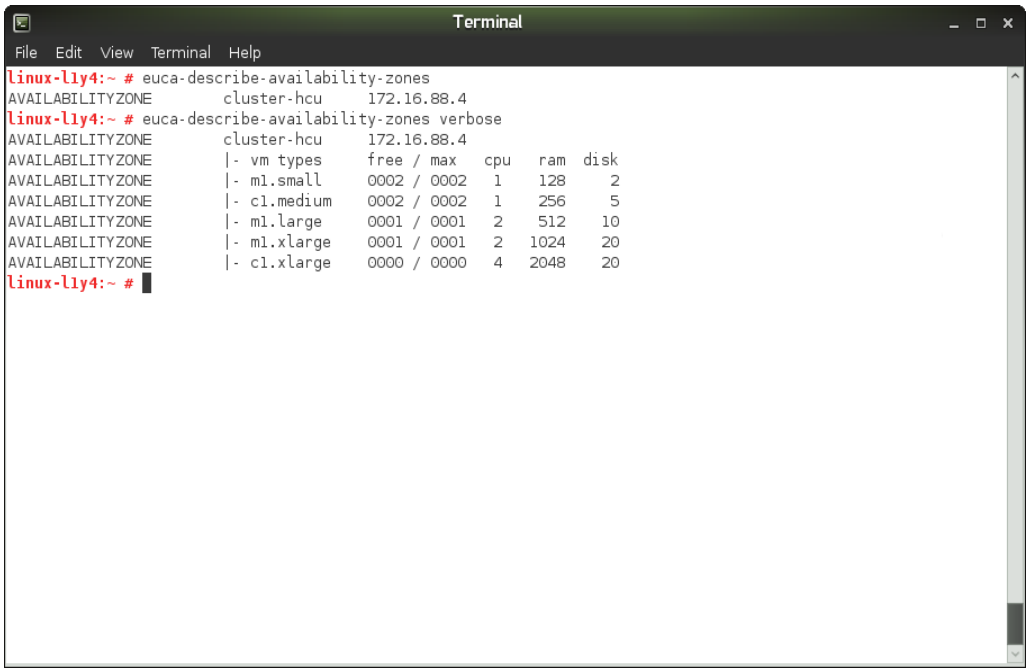


Figure 7.10 Key pair generation. Key 1and key2 are added in the cloud.

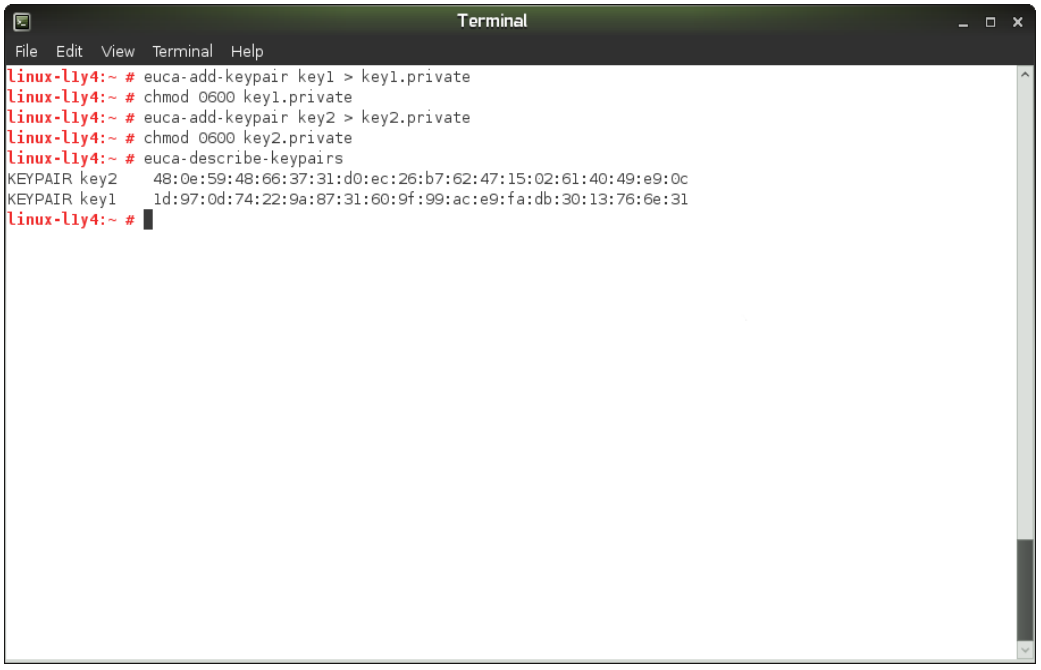


Figure 7.11 Different groups available in the cloud

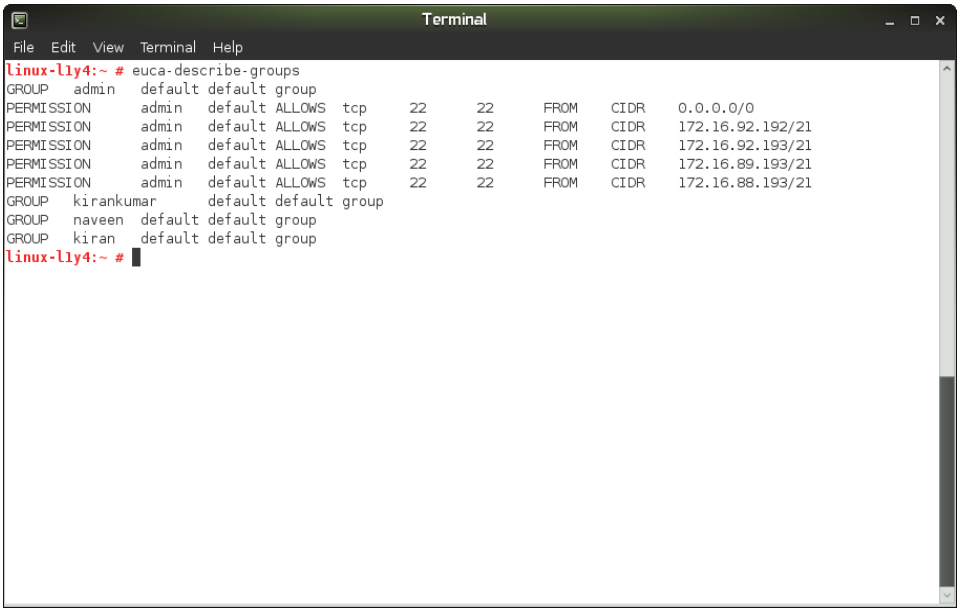


Figure 7.12 List of Image which are deployed in the cloud

Figure 7.13 List of instances that are been invoked in the cloud to run

List of screenshots using Hybridfox

Figure 7.14 Front end of Hybridfox

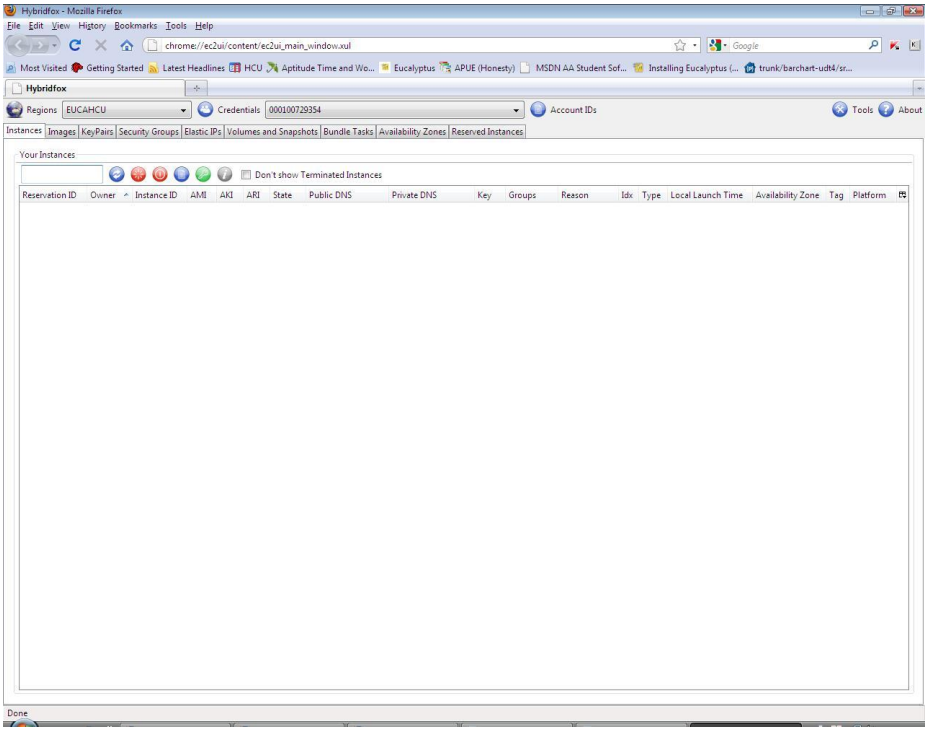


Figure 7.15 Managing regions.

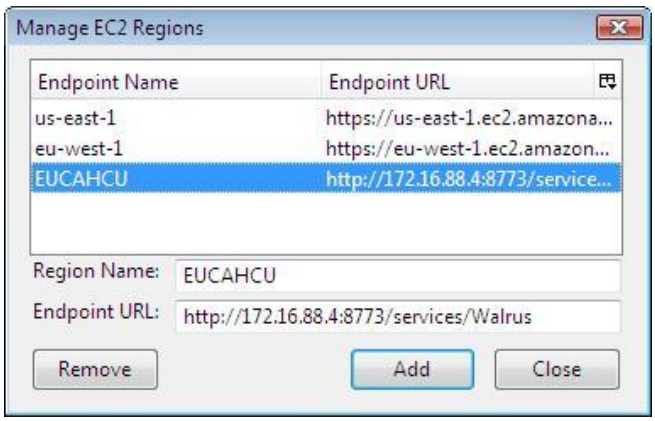


Figure 7.16 Managing Credentials

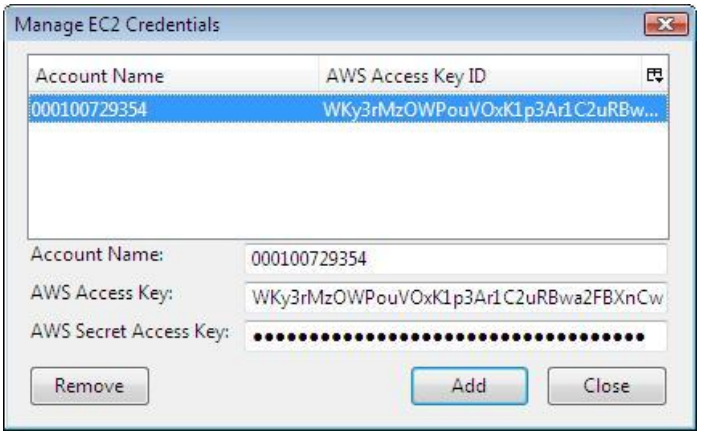


Figure 7.17 Managing Account ID

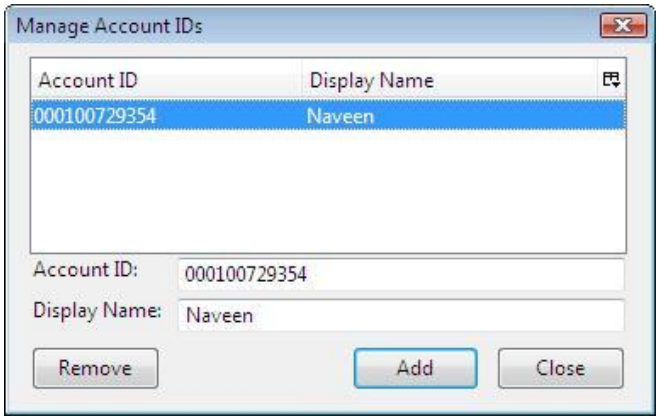


Figure 7.18 List of Images available in Cloud region HCU

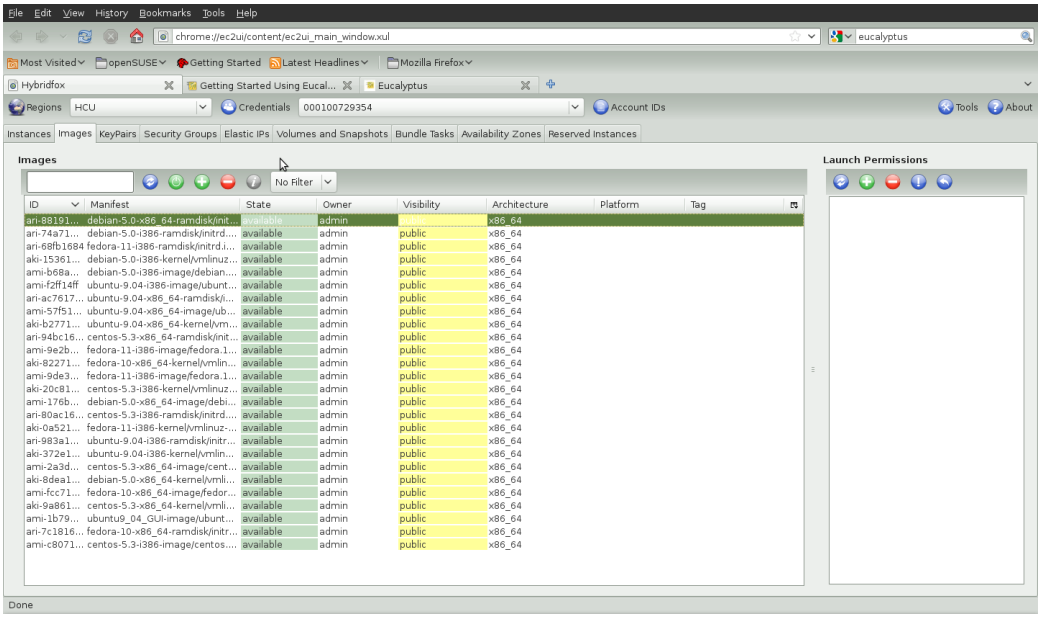


Figure 7.19 Generating the key pair

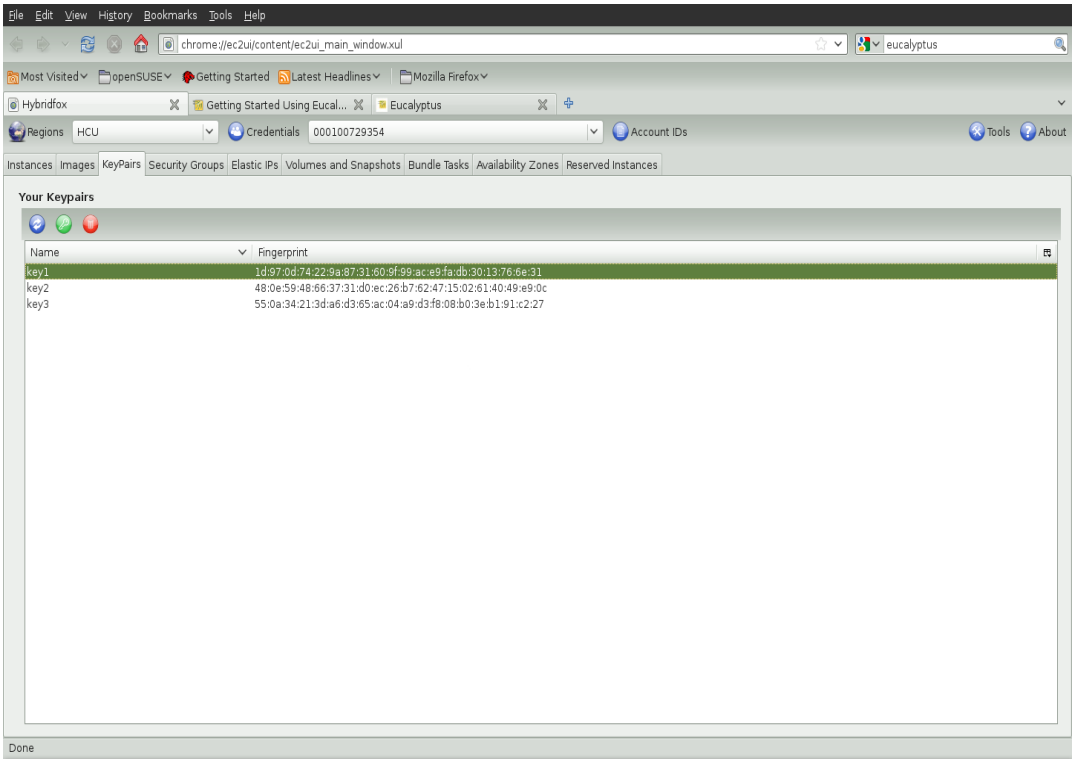


Figure 7.20 Different instances showing different status in cloud:

Pink: Terminates Instance

Green: Running Instance

Yellow: Pending Instance

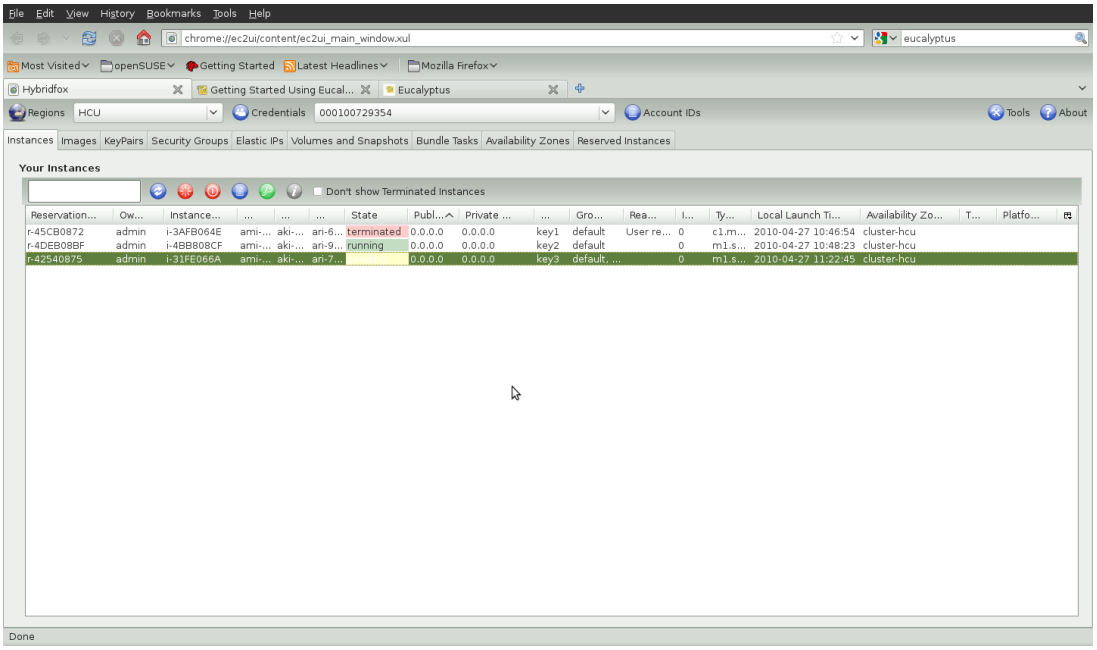


Figure 7.21 Single OS instance running in the Hyrbridfox.

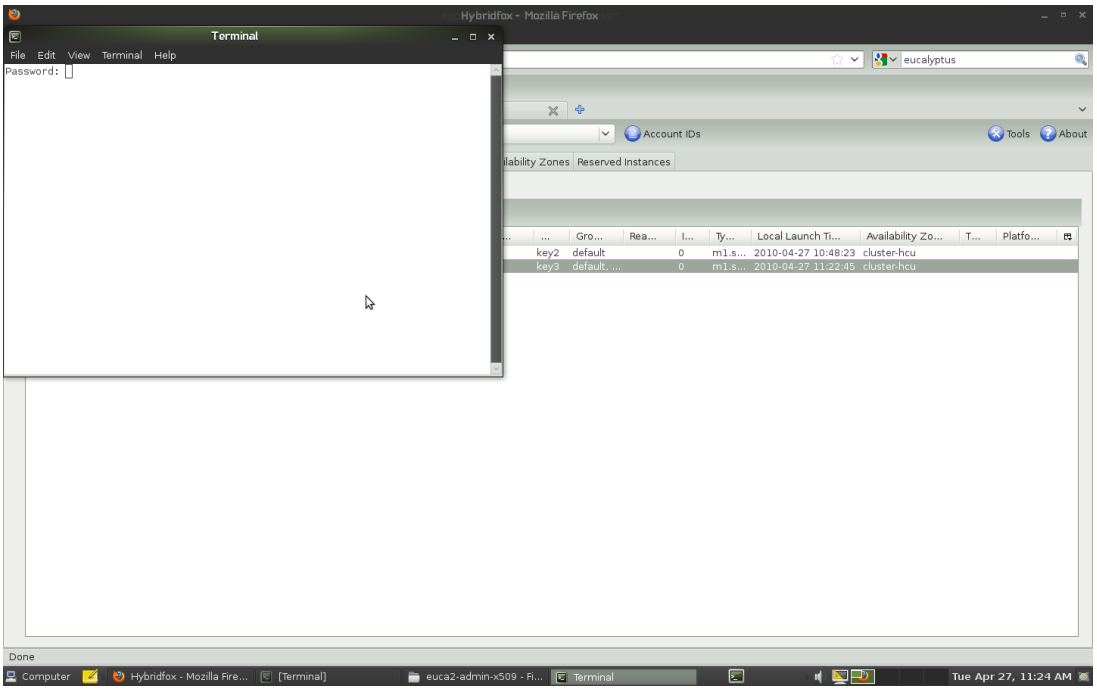


Figure 7.22 Multiple OS instances running in the Hyrbridfox

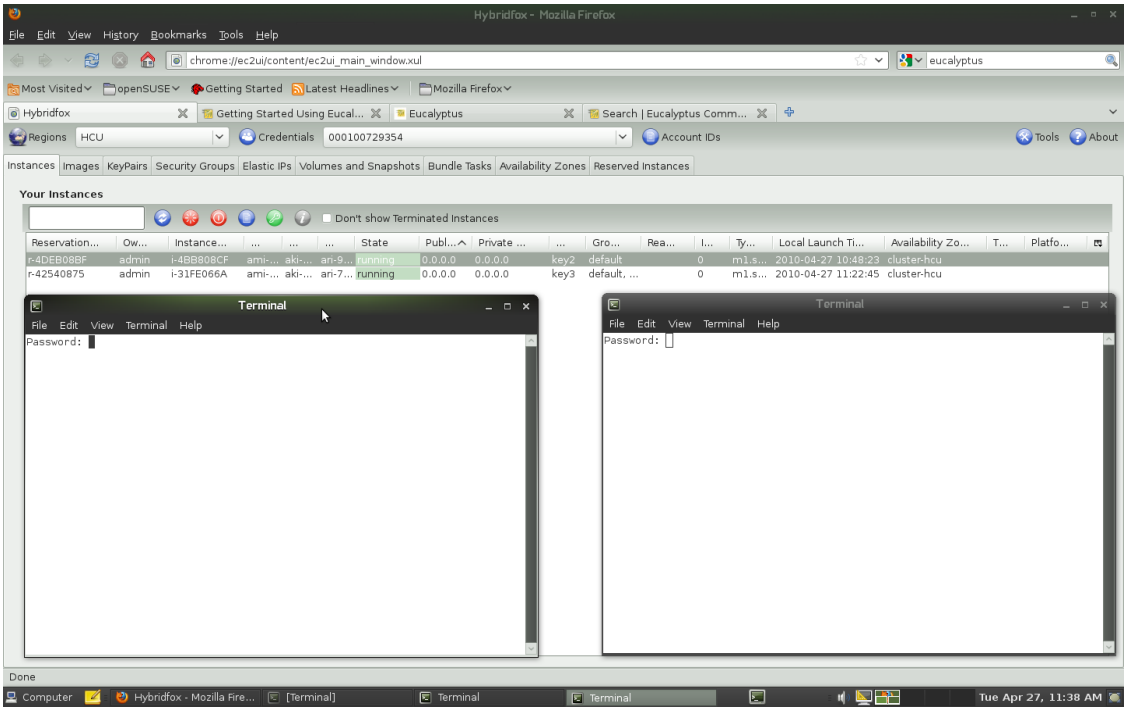


Figure 7.23 Data migration tool

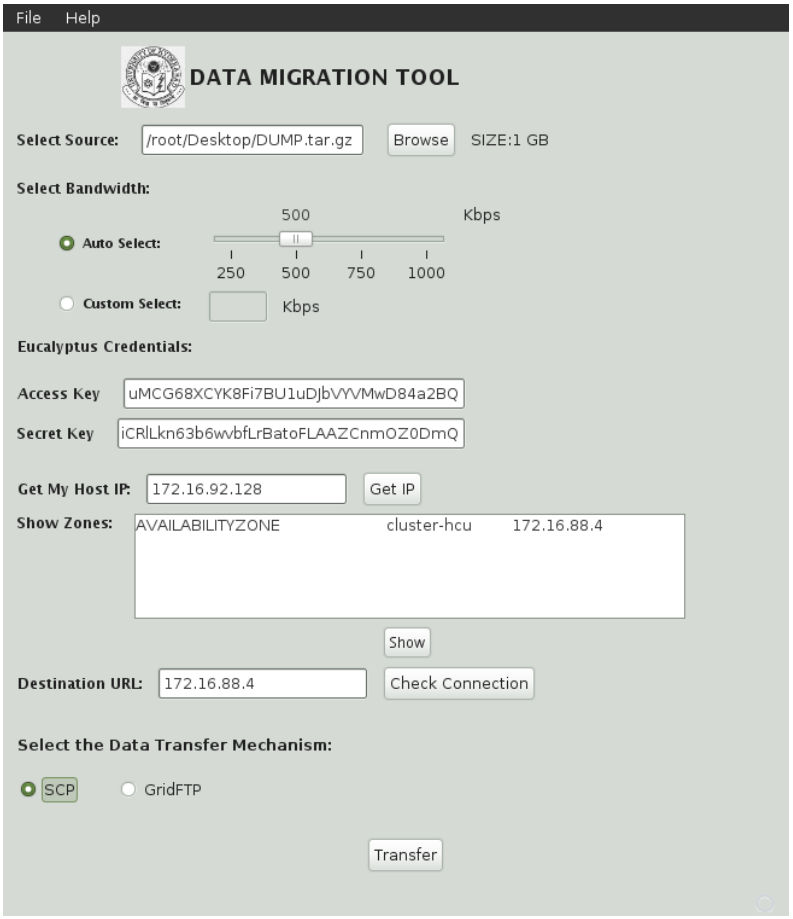


Figure 7.24 Connecting to cloud

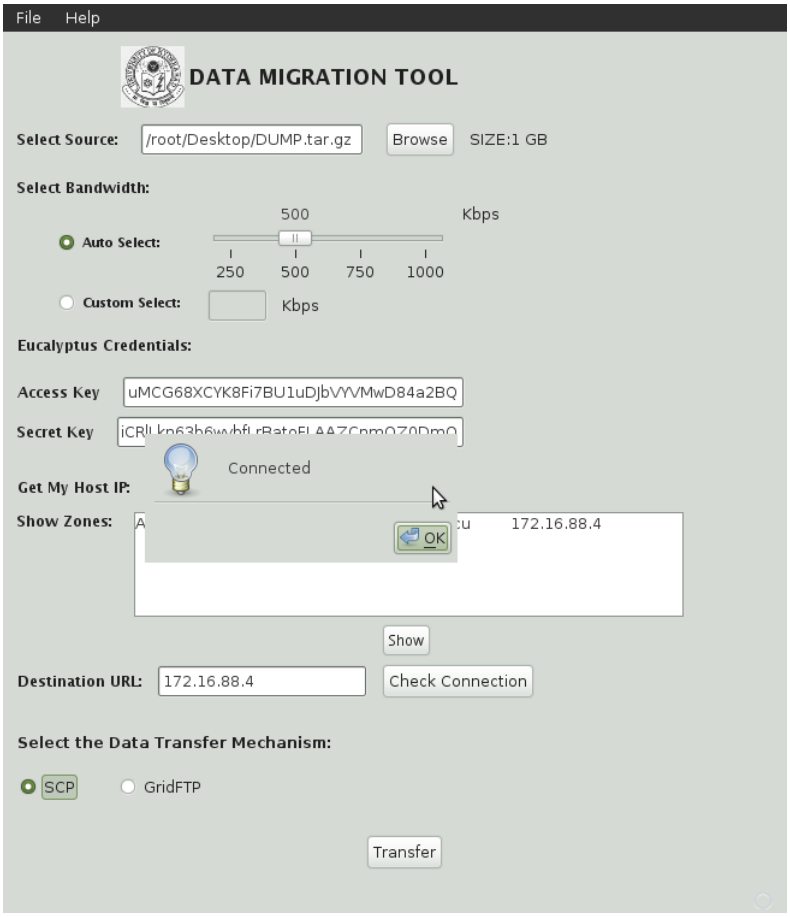


Figure 7.25 SCP tools for transfer

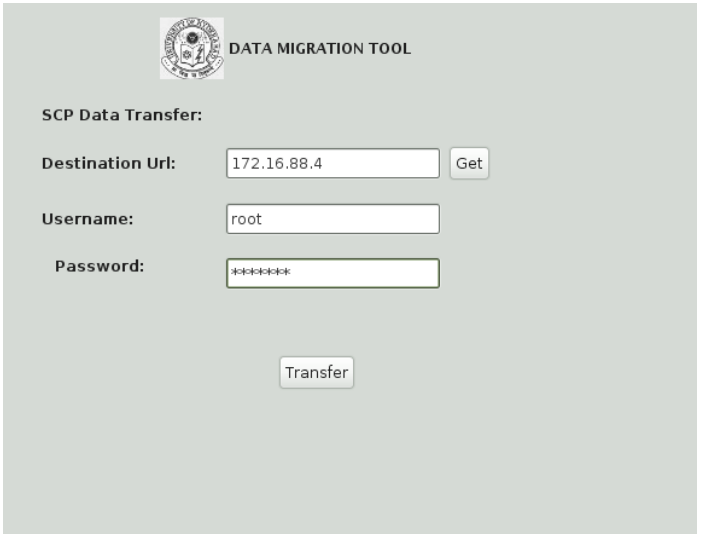
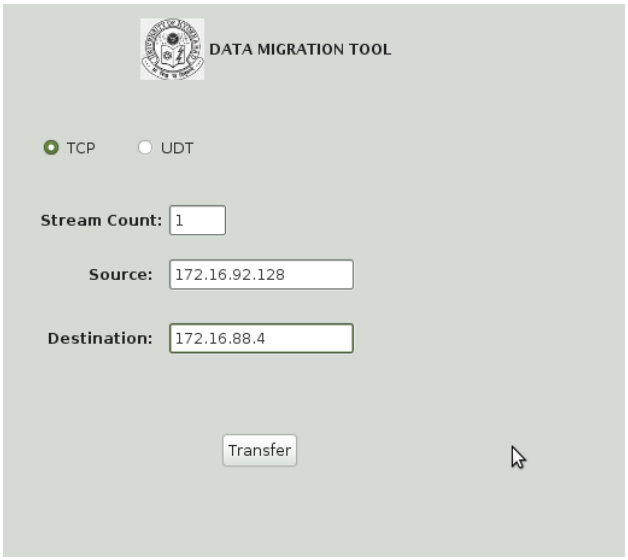


Figure 7.26 GridFTP in data migration tool



References

[1] Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Rajkumar Buyyaa, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic 2008.

[2] Eucalyptus: an open-source cloud computing infrastructure, by D Nurmi Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, Santa Barbara, CA 93106; Eucalyptus Systems Inc., 130 Castilian Dr., Goleta, CA 93117. For eucalyptus 2009.

[3] UDT as an Alternative Transport Protocol for GridFTP, John Bresnahan, Michael Link, Rajkumar Kettimuthu and Ian Foster, Proceedings of the 7th International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT 2009), Tokyo, Japan, May 2009.

[4] Introduction to Cloud Computing architecture, Sun microsystems White Paper 1st Edition, June 2009.

[5] Introduction to Cloud Computing, IBM Center for Bussiness of government, Vivek kundra, Fedral CIO, 2009.

[6] Amazon simple storage service api (2006-03-01) –http://docs. amazonweb services.com/AmazonS3/2006-03-01/

[7] Introduction to Cloud Computing, Liming Liu, 2008.

[8] RFC959 - File Transfer Protocol, J.Postel, J.Reynolds, ISI, 1985.

[9] SCP notes: blogs.sun.com/janp/entry/how_the_scp_protocol_works.

[10] Y. Gu and R. L. Grossman, "UDT: UDP-based data transfer for high-speed wide area networks,"Comput. Networks 51, 7 (May. 2007), 1777–1799. DOI= http://dx.doi.org/10.1016/j.comnet.2006.11.00.

[11] The Globus eXtensible Input/Output System (XIO): A protocol independent IO system for the Grid. William Allcock, John Bresnahan, Rajkumar Kettimuthu and Joseph Link, Mathematics and Computer Science Division, Argonne National Laboratory

[12] UDT: UDP-based data transfer for high-speed wide area networks. Yunhong Gu, Robert L. Grossman, 2006

FORUMS & SITES

[13] Cloud Vs Grid: http://markusklems.wordpress.com/2008/06/19/cloud-vs-grid/

[14] UDT: http://udt.sourceforge.net/

[15] Eucalyptus: http://open.eucalyptus.com/

[16] GridFTP: http://www.globus.org/toolkit/docs/4.2/4.2.1/data/gridftp/admin

[17] Eucalyptus forum: http://forum.eucalyptus.com/forum/

[18] Globus installation: http://www.globus.org/.

[19] Amazon cloud services: http://aws.amazon.com/ec2/#top

[20] Amazon web service & cloud service http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=87

[21] FTP: http://geekswithblogs.net/Lance/archive/2008/05/15/fxp---ftp-transfers-from- server- to-server.aspx

References

[1] Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Rajkumar Buyyaa, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic 2008.

[2] Eucalyptus: an open-source cloud computing infrastructure, by D Nurmi Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, Santa Barbara, CA 93106; Eucalyptus Systems Inc., 130 Castilian Dr., Goleta, CA 93117. For eucalyptus 2009.

[3] UDT as an Alternative Transport Protocol for GridFTP, John Bresnahan, Michael Link, Rajkumar Kettimuthu and Ian Foster, Proceedings of the 7th International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT 2009), Tokyo, Japan, May 2009.

[4] Introduction to Cloud Computing architecture, Sun microsystems White Paper 1st Edition, June 2009.

[5] Introduction to Cloud Computing, IBM Center for Bussiness of government, Vivek kundra, Fedral CIO, 2009.

[6] Amazon simple storage service api (2006-03-01) –http://docs. amazonwebservices.com/AmazonS3/2006-03-01/

[7] Introduction to Cloud Computing, Liming Liu, 2008.

[8] RFC959 - File Transfer Protocol, J.Postel,J.Reynolds,ISI,1985.

[9] SCP notes: blogs.sun.com/janp/entry/how_the_scp_protocol_works.

[10] Y. Gu and R. L. Grossman, “UDT: UDP-based data transfer for high-speed wide area networks,”Comput. Networks 51, 7 (May. 2007), 1777–1799. DOI= http://dx.doi.org/10.1016/j.comnet.2006.11.00.

[11] The Globus eXtensible Input/Output System (XIO): A protocol independent IO system for the Grid. William Allcock, John Bresnahan, Rajkumar Kettimuthu and Joseph Link, Mathematics and Computer Science Division, Argonne National Laboratory

[12] UDT: UDP-based data transfer for high-speed wide area networks. Yunhong Gu, Robert L. Grossman, 2006

FORUMS & SITES

[13] Cloud Vs Grid: http://markusklems.wordpress.com/2008/06/19/cloud-vs-grid/

[14] UDT: http://udt.sourceforge.net/

[15] Eucalyptus: http://open.eucalyptus.com/

[16] GridFTP: http://www.globus.org/toolkit/docs/4.2/4.2.1/data/gridftp/admin

[17] Eucalyptus forum: http://forum.eucalyptus.com/forum/

[18] Globus installation: http://www.globus.org/.

[19] Amazon cloud services: http://aws.amazon.com/ec2/#top

[20] Amazon web service & cloud service http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=87

[21] FTP: http://geekswithblogs.net/Lance/archive/2008/05/15/fxp---ftp-transfers-from- server- to-server.aspx