Bachelor thesis in Electronics and Computer Engineering
School of Engineering III
Politecnico Di Torino
December 2012

Title:

Exploring the Boto Interface for Resource Management in Cloud Infrastructures

Jawad Munir (158385)
s158385@studenti.polito.it

"Computer Engineering is no more about computers than astronomy is about telescopes"
E. W. Dijkstra

Supervised by:
Professor BIANCO ANDREA.

Preface

I would like to thank the following persons that have contributed in one way or another to this thesis.

Professor BIANCO ANDREA

All the Professors who guided me in the completion of my work

----

Professor Cosimo Anglano
Professor Massimo Canonico

ABSTRACT

This thesis investigates Boto API. We begin the story by explaining the key concepts regarding the understanding of underlying technology of Cloud Computing. Boto is a python add-on or we can say a python library to interact with amazon web services for example elastic cloud compute (EC2) and Simple Storage services (S3). Now as in 2006 when amazon launches it web services and cloud compute, it wasn't open source and the underlying virtualization technology was kept hidden under the piracy Act. But with time other vendors and open source cloud platforms emerges and among them the most prominent were Eucalyptus, Nimbus, Openstack.

In this thesis we are going to investigates also open source cloud platform Nimbus. We can interact with nimbus by using Boto API. The remote resources were taken from FutureGrid Project which is the university sponsored platform for research and development of Cloud technology. This platform provides you with the opportunity to use nimbus as IaaS and test various services.

The practical part of the work was to design a Virtual Machine Controller using Boto + Python. Now it need to Automate few of the services for example start a virtual machine or switch off a live instance.

So the detail of the controller is covered in chapter 4 of the thesis, in which also has provided the reason for the need to automate these functions.

In the end a conclusion is given for all the work done.

Keywords

Boto, EC2 (Elastic Cloud Compute), S3 (Simple storage service), AS (Auto scale), Load balancer (LB), IaaS, PaaS, SaaS

Contents

Chapter 1

Introduction to Cloud Computing

1.1 History

The origin of the term *cloud computing* is little bit controversial, but it appears to derive from the practice of using drawings of stylized clouds to denote networks in diagrams of computing and communications systems. As you remember, we used to denote the internet with a cloud.

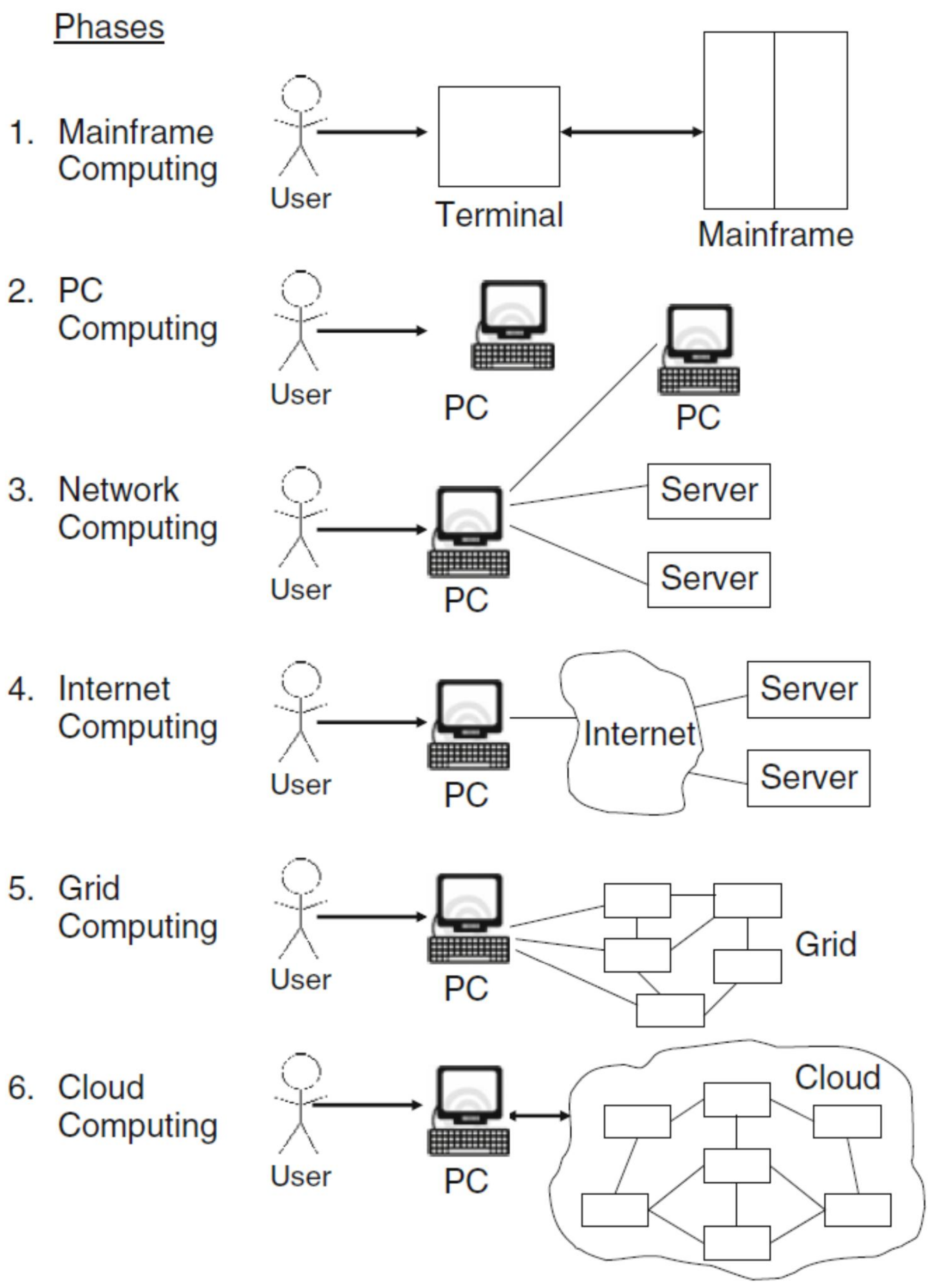
So back in the sixties and seventies you used to have something called mainframes and dumb terminals. so what that was is you have a mainframe and this was a very impressive computer this might have had 256K RAM, it was very powerful computer. Then connected to this mainframes from the secure socket, something called dumb terminal. So you could add you know anywhere between 5 to 100 dumb terminals.

These dumb terminals, they were simple little devices that allowed you to plug in a monitor a keyboard or a printer. These dumb terminals the only intelligence they had in them was the ability to connect to the mainframe. All the processing, everything happened on the mainframe.

Dumb terminal it basically sent all your little keystrokes to mainframe, the mainframe processed all the information and then sent the output to your monitor or the printer exciter.

So every one of this dumb terminals gets a little of (processing time) CPU time. So this was time-sharing. So with time engineers and researchers elaborated this concept of time-sharing and make large-scale computing power available to more users through time sharing. So this is the underlying concept behind Cloud Computing.

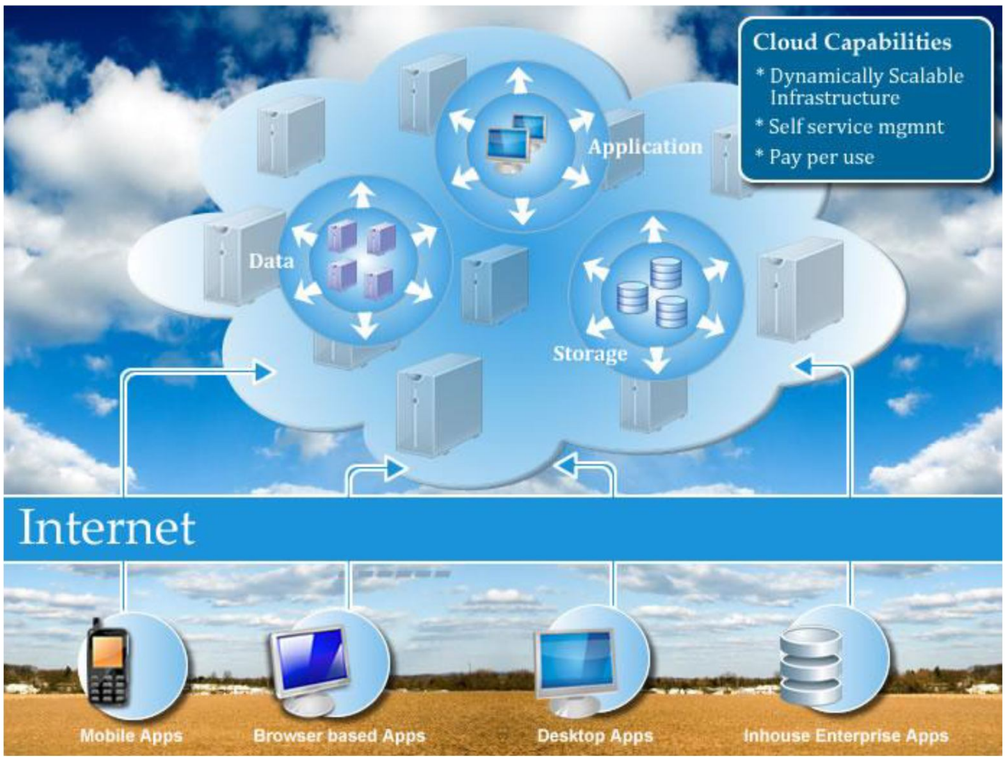
The following figure shows six computing paradigms – from mainframe computing to Internet computing, to grid computing and cloud computing (adapted from Voas and Zhang (2009)).



1.2 Definition of Cloud Computing

so we are going to be talking about what cloud computing is? Now cloud computing is the latest and greatest thing. And marketers for lots of big companies are all using cloud computing terms in their marketing campaigns to make them seem impressive. They can get clients and get customers and get money out of your pockets.

For examples, like Microsoft is running a new ad campaign with into the cloud where or somebody sits down at their computer and there's a whole flashy thing and then they go into the cloud, and it's really cool and fancy. The problem is most people don't really understand what's cloud computing? This cloud computing is in overall philosophy and design concept and it's much more complicated and yet much simpler than people give it credit.



1.2.1 Consumer Application Cloud Services

Cloud Computing is not a new thing. We all use consumer Application cloud services, for example twitter, Facebook, etc. You remember once upon a time consumer applications used to get installed on your computer, and you update it etc. So we are completely into cloud in consumer applications. But in business applications it happened in a different way. By business applications I

Mean financial applications, human resource (HR Management), CRM (customer relations management), web analytics. I am not going to discuss that. But what we don’t know that every business company which is gone public (i.e on internet), has been delivered as a Cloud Service.so we are into the cloud for more than 10 years. Now today when we all have such a buzz for cloud as for what happened at

Amazon in 2006. At Amazon they said, I am going to give my user computing resources (hardware and software) and storage as a Cloud service i.e on the network (typically the internet). Now the great thing in it, that now with for example 3000 dollars can buy me 10000 of computers (instances) for 30 minutes. Nobody has ever been given 10000 computers for 30 minutes, now this is magic and as smart and genius people out in world able to figure, what to do with such a great computational power , then we will be able to do things which were not economically possible before.

1.2.2 Cloud Computing versus Virtualization

A big mistake in our understanding of cloud computing is, when we conclude virtualization equals cloud computing. This is absolutely not true, virtual computing is a component of cloud computing but cloud computing in is far more than simple virtual computer.

The underline concept with cloud computing is that you are trying to separate the applications from the operating systems from the hardware that runs everything. What I mean by this?

In history if you remember if you have email service for your company. It meant, you had server hardware sitting somewhere in your company. For example if you had server hardware and then you installed windows-NT4, then on top of windows-NT4 operating system you installed Microsoft exchange server, to provide email service in your company and all that was a bundle of layers.

The problem the exchange server is, it depends upon everything below it. So if there is a problem in the operating system for example it gets a virus, or if a hard-drive gets clogged up with too much information, or if there is an error, your exchange server which means your email services all come to a halt. You no longer get your e-mail services because the operating system has failed for whatever reason. Again just like that with the server hardware if the CPU fan stops, or if your CPU burns out, or if the hard-drive fails or if the power supply fails. Your Email-exchange server goes down.

Idea of cloud computing is you are trying to disconnect application, operating system, hardware form, from themselves. But virtual computing is separating operating system from the hardware. We would come to the topic of virtualization again when we discuss technologies that are involved in cloud computing.

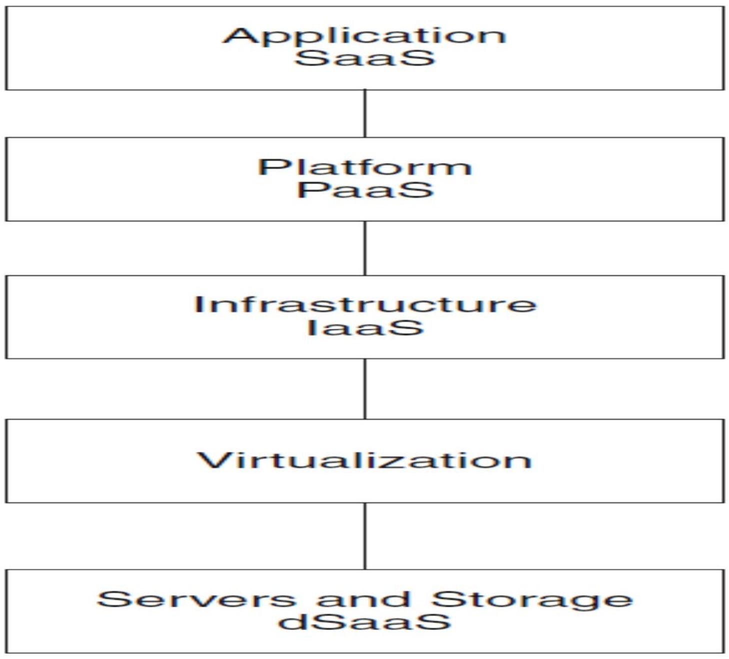
1.3 Service Models

Cloud providers offer their services according to three fundamental models. Infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) where IaaS is the most basic and each higher model abstracts from the details of the lower models.

http://en.wikipedia.org/wiki/Cloud_computing

Software as a service (SaaS)

In this model, a complete application is offered to the customer, as a service on demand. A single instance of the service runs on the cloud & multiple end users are serviced. On the customers‟ side, there is no need for upfront investment in servers or software licenses, while for the provider, the costs are lowered, since only a single application needs to be hosted & maintained. Today SaaS is offered by companies such as Google, Salesforce, Microsoft, Zoho, etc.



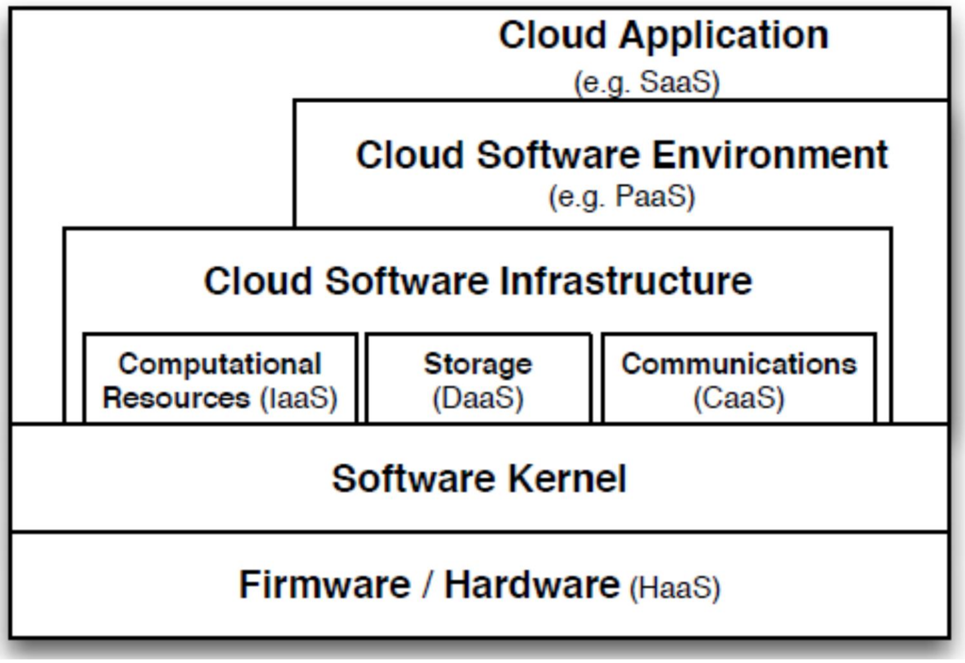
Layered architecture of Cloud Computing (adapted from Jones)

Platform as a service (PaaS)

Here, a layer of software or development environment is encapsulated & offered as a service, upon which other higher levels of service can be built. The customer has the freedom to build his own applications, which run on the provider's infrastructure. To meet manageability and scalability requirements of the applications, PaaS providers offer a predefined combination of OS and application servers, such as LAMP platform (Linux, Apache, MySQL and PHP), restricted J2EE, Ruby etc. Google's App Engine, Force.com, etc are some of the popular PaaS examples.

Infrastructure as a service (IaaS)

IaaS provides basic storage and computing capabilities as standardized services over the network. Servers, storage systems, networking equipment, data Centre space etc. are pooled and made available to handle workloads. The customer would typically deploy his own software on the infrastructure. Some common examples are Amazon, GoGrid, 3 Tera, etc.


Cloud Application
(e.g. SaaS)
Cloud Software Environment
(e.g. PaaS)
Cloud Software Infrastructure
Computational Resources (IaaS)
Storage (DaaS)
Communications (CaaS)
Software Kernel
Firmware / Hardware (HaaS)

Proposed Cloud Computing Ontology: depicted as five layers, with three constituents to the cloud infrastructure layer. The layered figure represents the inter-dependency and compos ability between the different layers in the cloud. (Proposed By Lamia Youseff, Maria Butrico, Dilma Da Silva University of California, Santa Barbara).

1.4 Deployment Models

Enterprises can choose to deploy applications on Public, Private or Hybrid clouds. Cloud Integrators can play a vital part in determining the right cloud path for each organization.

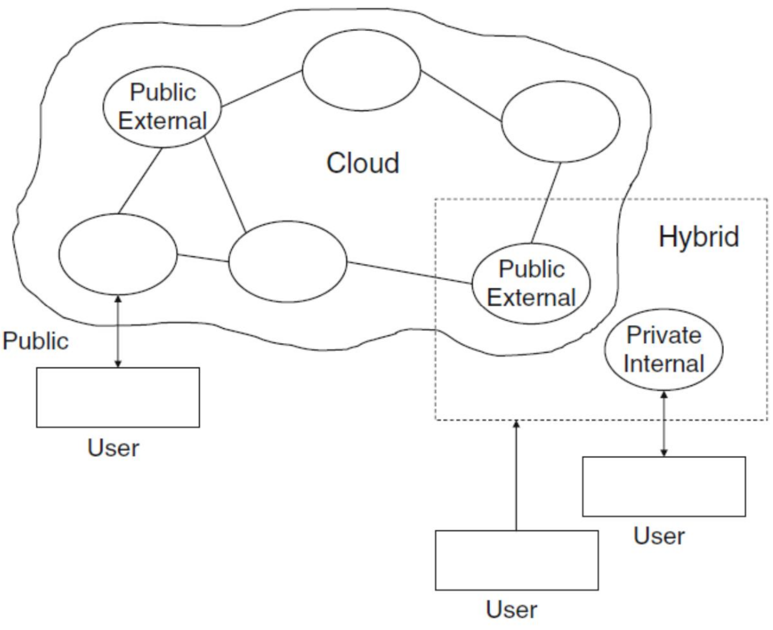
http://en.wikipedia.org/wiki/Cloud_computing

Public cloud

Public cloud applications, storage, and other resources are made available to the general public by a service provider. These services are free or offered on a pay-per-use model. Generally, public cloud service providers like Amazon AWS, Microsoft and Google own and operate the infrastructure and offer access only via Internet (direct connectivity is not offered).

Community cloud

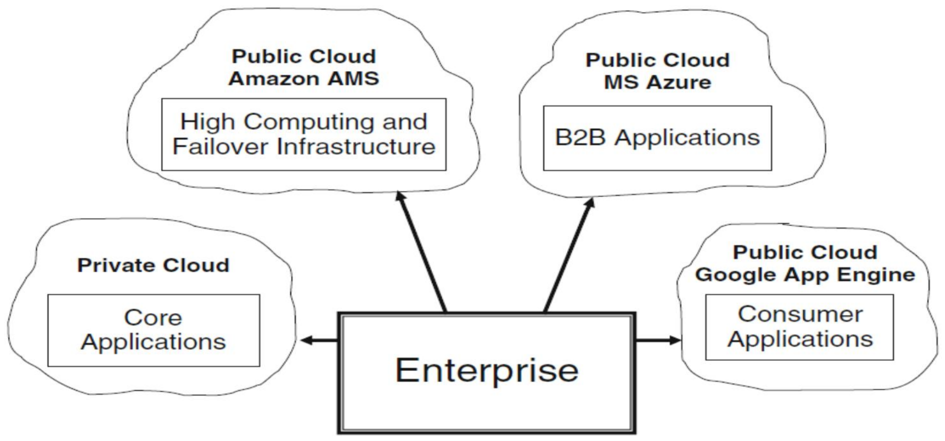
Community cloud shares infrastructure between several organizations from a specific community with common concerns (security, compliance, jurisdiction, etc.), whether managed internally or by a third-party and hosted internally or externally. The costs are spread over fewer users than a public cloud (but more than a private cloud), so only some of the cost savings potential of cloud computing are realized.



Hybrid cloud

Hybrid cloud is a composition of two or more clouds (private, community or public) that remain unique entities but are bound together, offering the benefits of multiple deployment models. By utilizing "hybrid cloud" architecture, companies and individuals are able to obtain degrees of fault tolerance combined with locally immediate usability without dependency on internet connectivity. Hybrid cloud architecture requires both on-premises resources and off-site (remote) server-based cloud infrastructure.

Hybrid clouds lack the flexibility, security and certainty of in-house applications. Hybrid cloud provides the flexibility of in house applications with the fault tolerance and scalability of cloud based services.



Private cloud

Private cloud is cloud infrastructure operated solely for a single organization, whether managed internally or by a third-party and hosted internally or externally. Undertaking a private cloud project requires a significant level and degree of engagement to virtualize the business environment, and it will require the organization to reevaluate decisions about existing resources. When it is done right, it can have a positive impact on a business, but every one of the steps in the project raises security issues that must be addressed in order to avoid serious vulnerabilities.

They have attracted criticism because users "still have to buy, build, and manage them" and thus do not benefit from less hands-on management, essentially "[lacking] the economic model that makes cloud computing such an intriguing concept".

1.5 The Inter-cloud

So inter-cloud this is interesting, now Google makes its own cloud, amazon its own, IBM makes cloud for other companies, so all these companies basically have same thing, as to give solutions to people who are remotely accessing the cloud. Now at some time it makes sense to move my data from cloud A to cloud B, Today we don’t have any standards or protocol where two clouds can interact with each other. Now this is like an internet problem days before. We don't know how inter-cloud protocols will standardize, but people can ask two or more clouds to interact with each other, to take advantage of compute power of each other.

Now one of the nice things about cloud, it has variable capacity, and it fairly natural to apply more computational capacity, if necessary for a while, but once you compute the task, you can re-allocate resources to others. Now the notion of mobile as a cloud-let is interesting, if it has same protocol as the cloud do for interacting with each other, so its like internet, internet don't care one person is micro-processor or and other is a super-computer from the standpoint of protocol everybody is same. So I think mobile should be cloud-let in the sense they should be able to interact clouds, the way any-other cloud interact with you.

And I hope five-years from now we would have inter-cloud standards, for inter-cloud interactions and mobile can be a part of it. So I define Inter-cloud is an interconnected global "cloud of clouds" , and an extension of the Internet "network of networks" on which it is based.

1.6 Technologies involved with cloud computing

Cloud Computing encompasses a lot more different types of technologies.

1.6.1 Web Applications

Web application is a simplest form of cloud computing. Web applications have been created using standard WWW technologies. So basically these applications you have been created using html5, JavaScript, php, etc.

Now why these are form of cloud computing, is because when you use these web applications you are actually accessing applications that are sitting in a someone else's (or a distant) server using the web browser that is installed on your computer, like Google docs is a classic web application.

You sit down at your computer, you open up a web-browser and then you go to the Google docs website, log into your account and then you can start writing documents. If basically as if the application was installed on your own computer.

Now office software that you were using is not installed on your computer, so if your computer crashes, or if your computer fails for whatever reason you can step to the side go to your neighbors computer get on and finish whatever paperwork that you were trying to do.

1.6.2 Database Clusters

Clustering or clusters of computers, it means when you set up multiple different servers so let's say we have 4 servers, these four servers all can have different hardware and also different operating system. But application installed on them is same, which connects them and replicate data. Mostly applications that can be clustered are database applications.

So databases if you are working in the windows world you may have heard of active directory clusters, or if you are dealing with web-programming you may have heard of my-sequel clusters, but I take as example my-sequel clusters. Now Servers with my-sequel database into a cluster, tell each server about the each other. What happens in this cluster is that replication occurs so that every single database, these servers, in four clusters contains the same information.

Now this is very important because what's goanna happen is when somebody let's say from the internet is coming to access database of, one of these clusters they will simply hit one of the server among the four clusters, the cluster figures out which server this person should be directed to.

So let's say a lot of people are trying to access a particular database on the internet and that database is a cluster of 4 servers, what's going to happen is, an incoming connection are going to come to first cluster, what first cluster is going to say is, I already has enough connections, I cannot deal with any more connections so when you come in from the internet, instead of hitting first server you will automatically get re-routed, they will be directed to a different server in the cluster this is something called load balancing. So basically if you want to give the best performance possible to your users that are coming in to use the database you use this load balancing and the cluster.

We also have the ability that if one of the servers fails then the cluster realizes the server fail and will not send connections to failed server. This is very important specially in the web world we just talk about web applications most web application store their data in databases nowadays generally my-sequel databases so you know when you create a document or you create a blog post, and if that is only one server and that one server fails you're done. But if you have a my-sequel cluster if one serve fails your blog stays up and running.

Clustering is a very important concept, when we are talking about cloud computing. So 90% stuff you do online has database. So it's better to store data in a clustered i.e cloud database environment.

1.6.3 Terminal Services

I already have talked about the main-frames and the dumb terminals before, now the terminal services is more or less the same concept but we now have new names and more characteristics associated.

Main-frame super computer is now "**Terminal services server**".
And our dumb clients are now named as "**thin clients**".

Now the idea is, a window opens up and then you will get your own little desktop environment with your applications whatever you're going to be using and you can go and you can work within that terminal services environment and just as if everything is installed on the computer in front of you. So with this terminal services server all the processing, the RAM, all the hard drive everything resides on this terminal services server you connect to the terminal services server with something called software thin client and then you can basically sit down at a desktop and basically if Microsoft office is installed you can write a word document.
The nice part is, to have a thin client to be working on a word document somebody else can have a thin client be working on excel document, another person can have something to be working on adobe Photoshop, another can be checking their email so basically all the processing for all the thin clients that are connected to the terminal services server happen on the terminal services server.

So basically you have two types of thin clients.

i) **Software thin clients.**
ii) **Hardware thin clients.**

This means that basically you either have a piece of hardware or you have a piece of software that is installed on your computer that can connect to this terminal services server.

Software thin clients

Software thin clients all this is a little application that you're going to install onto your computer example Trading application installed with windows computer, this application will connect to the terminal services server and then on your computer you will have a little window that opens up and gives you whatever applications that this services server is providing.

So again if you're dealing with a windows 2008 terminal services server you connect using a terminal services thin client to the server, when you click the icon of thin client, a window opens up and you're looking at something that looks like a normal window desktop. Great part about this case that desktop that you're looking at does not reside on your computer it resides on the terminal services server.

So if your computer died but your data doesn't go away, also your applications don't go away, you can simply go up to somebody else's thin client, access terminal services server and get back in, and finish doing your work.

Hardware thin clients

Basically all hardware thin clients are very small, very low powered computer devices basically all they do is they have the thin client application installed on them. So they're just powerful enough to be able to access the terminal services server, and no more powerful.

So with the hardware thin client you know you would not installed office onto it, you would not install adobe onto it. But the only thing it has installed is this thin client software so we can connect back to a terminal services server. The hardware is really small.

When you open up a thin client on your piece of hardware, you will see what looks to be a normal window desktop (off course if you are connected to normal windows8 terminal services server).

So you see there is nothing fancy about cloud computing it’s like the same as time-frame but a broader concept. The important thing to remember is, all the processing happens on the terminal services server none of the processing happens out on the thin clients.

1.6.4 Application Server

It is installed on the terminal services server. Terminal services server can also be used to run other applications, apart from just showing a desktop to the thin client.

So let's say if i wanted all the people in my office to have access to adobe Photoshop but i didn't actually want to install adobe Photoshop onto all the different computers on the network as for number of reasons for example licensing or I just simply don't want to install Photoshop and audit each and every computer.

So then people came up with Application servers. So with application servers you go back to the big terminal services server on top of that, you install your application server components. Again depending on what you want install, when you have thin clients connected, to terminal services server when they connect instead of accessing, a full windows environment when the window opens up all it will open up to is a Photoshop. So they can use a little thin client application connect to the terminal services server when they open it up they will be sitting and they will be looking at the application Photoshop, or a QuickBooks.

Beware: Application servers in these are not web-applications. Web-applications are applications that are created using standard web programming languages php, html, etc

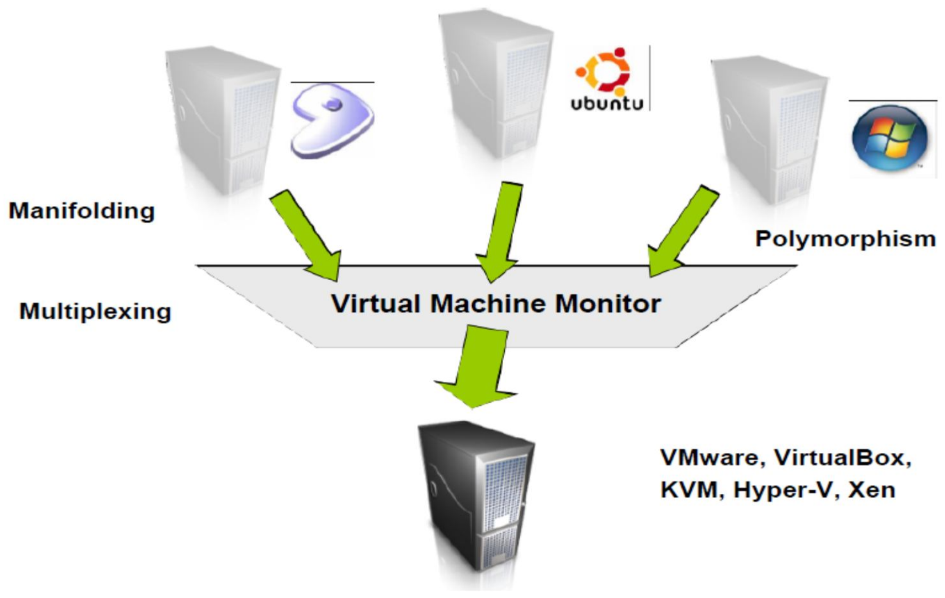
Application server is where real applications run. Applications like Photoshop, AutoCAD, Microsoft office, except that is actually installed on the server and then when the thin client connects basically the window opens up and employees sitting looking straight at the application they can access the application as if it is installed on their computers.

1.6.5 Virtualization

So now we get to topic that everybody, thinks of when they think about cloud computing and that is virtualization.

Virtual computers is basically you are able to separate the operating system from the hardware that runs the operating system, what this means that you can transfer the entire operating system with applications settings etc, from one piece of server hardware to another piece of server hardware and everything stays intact.

So in the bad old days before virtualization if i want to move/migrate my server operating system from one piece of physical hardware to another piece of physical hardware I would have to do a backup of all the data to make sure I got a hold of the data, and then install operating system with all the applications and it like a long process of 24 hours. Well now using virtualized environment, the operating system actually separated from hardware and you can just take the entire operating system again with all the application installed and everything etc, and you can just basically copy and pasting to a new peace of server hardware and be on your way. So migrations that may have taken in 24 hours in the past now declare how much data you're transferring might take an hour. So it happens through the use of virtualization software.



So virtualization software comes in two flavors.

1.6.5.1 client-installed virtualization software

Client installed means you had hardware and then you install a normal operating system on this piece of hardware. Then on top of the operating system that you just installed you install client virtualization software (VIRTUAL BOX)that then allows you to install an another operating system on top of the operating system that you already have. Note that every time we create new operating system in a virtual environment it's called an instance.

1.6.5.2 Hypervisor Software Virtualization

So with hypervisor (which controls supervisor is called a hypervisor) basically what happens is you have two pieces of software that you were going to need to deal with in this virtualized environment. They are much more powerful than virtual box.

The first is the hyper-visor and it basically kind of operating system that you install on the hardware that you want to run your virtual computers on. VMware is somebody that creates a lot of virtual computing stuff so their hyper-visor is something called ESXi.

You install ESXi onto the hardware where you want your virtual computers to run. And it returns you an IP address and computer name only.

Now you use management software. So this management software if we're talking about VMware world, they call it Vsphere.

This Vsphere, you install all into whatever computer you're going to use in order to administer your virtual computers. This management software gives the GUI to create/control the Virtual machine.

VMware they gave you a basic version of the ESXi and Vsphere for free, just to give you power to create a little virtual computers. But well if you want all of this fault tolerance built-in that's when you have to buy like the advanced addition for three thousand dollars and then they give you all of the additional features.

1.6.6 Hosted Instances

Once people figured out that you can separate the operating system from the underlying hardware. A lot of the big technology companies realized that they could sell instances of operating system up on their data centers in virtual environments at commodity prices.

So basically they would offer their clients, instead of buying server hardware on your own you can just create an instance of your operating system in our virtual environment and then use it for whatever you need.

Amazon was the first one. What they did is they said you only have to pay for what you use so let's say you need a small little server to do basic tasks for your company well, you can say i want to create a server that has 1GB of data storage, it's a Linux server, I need 512MB RAM, and any GB of transfer per month you can create an instance for your operating system with this configuration and then you only pay for it.

1.6.7 Hosted Solutions

Hosted solutions are applications or services that are hosted on Cloud.

So you know way back in the day the original host services really were things like Hotmail or Gmail. They gave interface to read your email, to give you an interface to be able write emails etc, so with that as a host service, one now as we go more and more into cloud computing world there are more and more of these hosted solutions out there.

When you ask you know what's kind of hosted solutions are out there well skies the limit.

Hosted solutions uses web applications, databases, email all that kind of stuff. For example look at sales forces contact management or sales management system. It is a hosted solution. Adobe is planning to move all of its products to being hosted solutions. So what adobe will do, is pay amazon to host its application and the user can access the adobe products (for example adobe Photoshop).

So instead of installing the software onto your computer it is all just hosted up on the internet. And you pay a monthly or a yearly fee to use it.

So all above technologies that I mentioned above are relevant with cloud computing. And one need to have a very good understanding of all these technologies in order to fully understand the underline concept of cloud computing.

1.7 Cloud Computing Benefits

Enterprises would need to align their applications, so as to exploit the architecture models that Cloud Computing offers. Some of the typical benefits are listed below:

1. **Reduced Cost**

   There are a number of reasons to attribute Cloud technology with lower costs. The billing model is pay as per usage; the infrastructure is not purchased thus lowering maintenance. Initial expense and recurring expenses are much lower than traditional computing.

2. **Increased Storage**

   With the massive Infrastructure that is offered by Cloud providers today, storage & maintenance of large volumes of data is a reality. Sudden workload spikes are also managed effectively & efficiently, since the cloud can scale dynamically.

3. **Flexibility**

   This is an extremely important characteristic. With enterprises having to adapt, even more rapidly, to changing business conditions, speed to deliver is critical. Cloud computing stresses on getting applications to market very quickly, by using the most appropriate building blocks necessary for deployment.

1.8 Cloud Computing Challenges

Despite its growing influence, concerns regarding cloud computing still remain. In our opinion, the benefits outweigh the drawbacks and the model is worth exploring. Some common challenges are:

1. Data Protection

Data Security is a crucial element that warrants scrutiny. Enterprises are reluctant to buy an assurance of business data security from vendors. They fear losing data to competition and the data confidentiality of consumers. In many instances, the actual storage location is not disclosed, adding onto the security concerns of enterprises. In the existing models, firewalls across data centers (owned by enterprises) protect this sensitive information. In the cloud model, Service providers are responsible for maintaining data security and enterprises would have to rely on them.

2. Data Recovery and Availability

All business applications have Service level agreements that are stringently followed. Operational teams play a key role in management of service level agreements and runtime governance of applications. In production environments, operational teams support

- Appropriate clustering and Fail over
- Data Replication
- System monitoring (Transactions monitoring, logs monitoring and others)
- Maintenance (Runtime Governance)
- Disaster recovery
- Capacity and performance management

If, any of the above mentioned services is under-served by a cloud provider, the damage & impact could be severe.

3. Management Capabilities

Despite there being multiple cloud providers, the management of platform and infrastructure is still in its infancy. Features like „Auto-scaling. for example, are a crucial requirement for many enterprises. There is huge potential to improve on the scalability and load balancing features provided today.

4. Regulatory and Compliance Restrictions

In some of the European countries, Government regulations do not allow customer's personal information and other sensitive information to be physically located outside the state or country. In order to meet such requirements, cloud providers need to setup a data center or a storage site exclusively within the country to comply with regulations. Having such an infrastructure may not always be feasible and is a big challenge for cloud providers.

1.9 Future of Cloud Computing

At present moment this industry is at its peak, and also in coming days it will grow. Right now we are in the early days of cloud computing, with many organizations taking their first, tentative steps. But five year on-ward cloud is going to be a major — and permanent — part of the enterprise computing infrastructure.

Eight years from now we are likely to see low-power processors crunching many workloads in the cloud, housed in highly automated data centers and supporting massively federated, scalable software architecture.

Alongside this increase in demand from enterprise, there will be development in the technologies that support clouds, with rapid increases in processing power making cloud projects even cheaper, while technologies currently limited to supercomputing will make it into the mainstream.

And of course, by five years, a generational shift will have occurred in organizations that means a new generation of CIOs will be in charge who have grown up using cloud-based tools, making them far more willing to adopt cloud on an enterprise scale.

With these developments you can imagine that cloud Computing is going to be hot industry for the next many years. As all software and Applications need to be written again and it should be in the cloud.

Chapter 2

Amazon Web Services and Boto API

2.1 Introduction to AWS

https://aws.amazon.com/

In 2006, Amazon Web Services (AWS) began offering IT infrastructure services to businesses in the form of web services -- now commonly known as cloud computing. One of the key benefits of cloud computing is the opportunity to replace up-front capital infrastructure expenses with low variable costs that scale with your business. With the Cloud, businesses no longer need to plan for and procure servers and other IT infrastructure weeks or months in advance. Instead, they can instantly spin up hundreds or thousands of servers in minutes and deliver results faster.

Today, Amazon Web Services provides a highly reliable, scalable, low-cost infrastructure platform in the cloud that powers hundreds of thousands of businesses in 190 countries around the world. With data center locations in the U.S., Europe, Brazil, Singapore, Japan, and Australia, customers across all industries are taking advantage of the following benefits:

Low Cost

AWS offers low, pay-as-you-go pricing with no up-front expenses or long-term commitments. We are able to build and manage a global infrastructure at scale, and pass the cost saving benefits onto you in the form of lower prices. With the efficiencies of our scale and expertise, we have been able to lower our prices on 15 different occasions over the past four years.

Agility and Instant Elasticity

AWS provides a massive global cloud infrastructure that allows you to quickly innovate, experiment and iterate. Instead of waiting weeks or months for hardware, you can instantly deploy new applications, instantly scale up as your workload grows, and instantly scale down based on demand. Whether you need one virtual server or thousands, whether you need them for a few hours or 24/7, you still only pay for what you use.

Open and Flexible

AWS is a language and operating system agnostic platform. You choose the development platform or programming model that makes the most sense for your business. You can choose which services you use, one or several, and choose how you use them. This flexibility allows you to focus on innovation, not infrastructure.

Secure

AWS is a secure, durable technology platform with industry-recognized certifications and audits: PCI DSS Level 1, ISO 27001, FISMA Moderate, HIPAA, and SAS 70 Type II. Our services and data centers have multiple layers of operational and physical security to ensure the integrity and safety of your data.

2.2 Amazon Products and Services

https://aws.amazon.com/

Amazon platform offers a number of products and services. It has a long list, and in this report I will not cover every amazon product. But still we have plenty to mention.

2.2.1 Compute

Amazon Elastic Compute Cloud(EC2)
Amazon Elastic Compute Cloud delivers scalable, pay-as-you-go compute capacity in the cloud.

Amazon Elastic Map Reduce
Amazon Elastic Map Reduce is a web service that enables businesses, researchers, data analysts, and developers to easily and cost-effectively process vast amounts of data.

Auto Scaling
Auto Scaling allows you to automatically scale your Amazon EC2capacity up or down according to conditions you define.

Elastic Load Balancing
Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances.

2.2.2 Storage

Amazon Simple Storage Service

Amazon Simple Storage Service provides a fully redundant data storage infrastructure for storing and retrieving any amount of data, at any time, from anywhere on the Web.

Amazon Glacier
Amazon Glacier is an extremely low-cost storage service that provides secure and durable storage for data archiving and backup.

Amazon Elastic Block Store
Amazon Elastic Block Store provides block level storage volumes for use with Amazon EC2 instances. Amazon EBS volumes are off-instance storage that persists independently from the life of an instance.

AWS Import/Export
AWS Import/Export accelerates moving large amounts of data into and out of AWS using portable storage devices for transport.

AWS Storage Gateway
AWS Storage Gateway is a service connecting an on-premises software appliance with cloud-based storage to provide seamless and secure integration between an organization's on-premises IT environment and AWS's storage infrastructure.

2.3 Amazon Solutions

https://aws.amazon.com/

Hundreds of thousands of customers have joined the Amazon Web Services (AWS) community and use AWS solutions to build their businesses. The AWS cloud computing platform provides the flexibility to build your application, your way, regardless of your use case or industry. You can save time, money, and let AWS manage your infrastructure, without compromising scalability, security, or dependability.

Common customer use cases or solutions using AWS include:

2.3.1 Application Hosting

Application vendors are looking for ways to offer hosted alternatives to packaged software. Hosted applications or Software-as-a-Service (SaaS) offerings allow businesses to take advantage of subscription-based revenue models, easier upgrade paths, reduced time to market, more predictable costs, and other benefits. Amazon Web Services offers a number of products, including Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Elastic Block Store (Amazon EBS) that give application vendors an easy way to use the massive compute power of the Amazon cloud computing platform to host their existing software on the Internet.

2.3.2 Backup and Storage

Even as storage becomes more plentiful and affordable, businesses are still faced with the task of managing their growing storage infrastructure. Amazon Web Services provides a cost-effective solution for storing information in the cloud that eliminates the burden of provisioning and managing hardware. Amazon Simple Storage Service (Amazon S3) provides a highly scalable, reliable, and inexpensive data storage infrastructure that enables you to build dependable backup solutions. Thousands of customers already use Amazon S3 as their backup location, and other customers have created compelling end-user backup, storage, and disaster recovery solutions using AWS.

2.3.3 Content Delivery

Amazon CloudFront makes it easier to distribute content to end-users quickly, with low latency and high data transfer speeds. The service lets you deliver your content through a worldwide network of edge locations and automatically routes end-user requests to closest edge location, so content is delivered with the best possible performance. Amazon CloudFront works seamlessly with Amazon Simple Storage Service (Amazon S3), which durably stores the original, definitive versions of your files. Like other Amazon Web Services, there are no long term contracts or monthly commitments and you only pay for what you use.

2.3.4 Databases

Amazon Web Services provides a number of database solutions for developers and businesses. Amazon RDS enables you to run a fully featured relational database while offloading database administration; Amazon SimpleDB provides simple index and query capabilities with seamless scalability; and using one of our many relational database AMIs on Amazon EC2 and Amazon EBS allows you to operate your own relational database in the cloud. There are important differences between these alternatives that may make one more appropriate for your use case.

2.3.5 E-commerce Applications

E-commerce applications require an infrastructure that ensures security, dependability, and the ability to handle sudden and/or seasonal spikes in demand. In addition, e-commerce applications require proven credit card processing capabilities. Amazon Web Services and other Amazon.com business units offer a number of solutions to help you run an e-commerce website. For example, the Amazon Flexible Payments Service (Amazon FPS) offers a powerful solution for handling payment. Amazon FPS automatically processes payments and refunds so that you can focus on delivering great solutions for your customers instead of managing financial transactions.

2.3.6 Enterprise IT

Many enterprises maintain and operate their own IT infrastructure — a task that often distracts from differentiating their business from competition and providing value for their customers. Amazon Web Services offers a reliable, secure environment on which businesses and their IT partners can host both internal- and external-facing applications. By hosting on the Amazon cloud computing platform, businesses can access on-demand, cost-effective infrastructure solutions and subsequently save time, money, and management resources.

2.3.7 High Performance Computing

Many businesses have problems that could be solved quickly and easily if only they had access to an infinite supply of compute power. Businesses with high performance computing (HPC) needs are turning to Amazon Web Services for access to a massive compute cloud. Amazon Elastic Compute Cloud (Amazon EC2) provides access to a tremendous amount of compute power, enabling you to process very large data sets across multiple virtual compute instances using any algorithm, programming language, or operating system you wish. Alternatively, businesses can use Amazon Elastic MapReduce which offers a hosted Hadoop framework so you no longer need to worry about the set-up, management or tuning of Hadoop clusters.

2.3.8 Media Hosting

Media distribution can be an unpredictable and expensive business. Companies looking to serve media files can incur extremely high storage and bandwidth costs, far exceeding those of many other businesses. Cost-effectively storing and distributing large media files become primary concerns for making the business economics work. Amazon Web Services enables cost-effective, scalable, and dependable distribution of content with services such as Amazon S3 and Amazon CloudFront.

2.3.9 On-Demand Workforce

Businesses regularly need tasks completed that computers cannot accurately accomplish. To find and hire people with specific skills to accomplish these tasks adds unwanted complexity and staffing costs. Amazon Mechanical Turk helps businesses connect with skilled workers for time- or project-specific tasks. It gives businesses access to a vast network of human intelligence with the efficiency of computers, and enables them to complete projects faster and more cost-effectively.

2.3.10 Search Engine Applications

Search engines and web crawlers require a large amount of processing power and storage in order to index the web and provide their customers with a satisfactory experience. Using the massive compute power and storage of Amazon Web Services, businesses with search engine or web crawler applications are able to scale their infrastructure resources to meet their ever changing needs.

2.3.11 Web Hosting

Web hosting development has unique requirements for scalability, reliability, security, and more. Amazon Web Services provides massive compute power that can scale as your business grows, very large amounts of storage that can handle the most demanding requirements your application may throw at it, a powerful database service that is both easy to use and flexible enough for the unique requirements of web applications, and several other services that satisfy your dynamic web hosting needs.

2.3.12 Life Sciences

Decreasing costs and rapid technological innovation have resulted in a tremendous increase in the volume and throughput of biological data being generated at large research institutes, individual labs and biopharma companies. At the same time, Life Sciences research is becoming increasingly collaborative and complex, leveraging multiple technologies to get a systems level understanding of diseases and organisms. Bioinformaticians, software developers and IT departments are now leveraging AWS to create scalable and highly available IT infrastructures.

2.4 Introduction to Boto Interface

Boto is a python interface to Amazon web services. It’s an integrated interface to current and future infrastructural services offered by Amazon Web Services.

The goal of Boto is to support the full breadth and depth of Amazon Web Services. In addition, Boto provides support for other public services such as Google Storage in addition to private cloud systems like Eucalyptus, Open Stack and Open Nebula.

Boto is developed mainly using Python 2.6.6 and Python 2.7.1 on Mac OSX and Ubuntu Maverick. It is known to work on other Linux distributions and on Windows. Boto requires no additional libraries or packages other than those that are distributed with Python. Efforts are made to keep boto compatible with Python 2.5.x but no guarantees are made.

2.5 Currently supported Amazon Services by Boto API

Now Boto API currently supports, numbers of Amazon services. But for our purpose we will only discuss two services of amazon in detail, as these are two services that we are concerned for the design of a Controller.

2.5.1 Amazon Elastic Cloud Compute (EC2)

http://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud

Is a central part of Amazon's cloud computing platform, Amazon Web Services (AWS).EC2 allows users to rent virtual computers on which to run their own computer applications. EC2 allows scalable deployment of applications by providing a Web service through which a user can boot an Amazon Machine Image to create a virtual machine, which Amazon calls an "instance", containing any software desired. A user can create, launch, and terminate server instances as needed, paying by the hour for active servers, hence the term "elastic". EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy.

Elastic Block Storage

Elastic Block Storage (EBS) provides raw block devices. That can be attached to Amazon EC2 instances. These block devices can then be used like any raw block device. In a typical use case, this would include formatting the device with a file system and mounting said file system. In addition EBS supports a number of advanced storage features, including snapshotting and cloning. Currently EBS volumes can be up to 1TB in size. EBS volumes are built on replicated back end storage, so that the failure of a single component will not cause data loss. The EBS product was introduced to the general public by Amazon in August 2008.

Instances Types

EC2 uses Xen virtualization. Each virtual machine, called an "instance", functions as a virtual private server. Amazon sizes instances based on "Elastic Compute Units". The performance of otherwise identical virtual machines may vary.

Reserved Instances

Reserved instances enable EC2 service users to reserve an instance for one or three years. There is a fee associated with reserving an instance. The corresponding per hour rate charged by Amazon to operate the instance is much less than the rate charged for on-demand instances.

Features

When it launched in August 2006, the EC2 service offered Linux and later Sun Microsystems' OpenSolaris and Solaris Express Community Edition. In October 2008, EC2 added the Windows Server 2003 and Windows Server 2008 operating systems to the list of available operating systems. In November 2012, Amazon officially supports running FreeBSD in EC2. In March 2011, NetBSD AMIs became available.

2.5.2 Auto Scaling

Amazon's Auto Scaling feature of EC2 allows it to automatically adapt computing capacity to site traffic. Auto Scaling allows you to automatically scale your Amazon EC2 capacity up or down according to conditions you define. With Auto Scaling, you can ensure that the number of Amazon EC2 instances you’re using scales up seamlessly during demand spikes to maintain performance, and scales down automatically during demand lulls to minimize costs. Auto Scaling is particularly well suited for applications that experience hourly, daily, or weekly variability in usage. Auto Scaling is enabled by Amazon CloudWatch and available at no additional charge beyond Amazon CloudWatch fees.

2.5.2.1 Auto Scaling with Elastic Load Balancing

Elastic Load Balancing

Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances. It enables you to achieve even greater fault tolerance in your applications, seamlessly providing the amount of load balancing capacity needed in response to incoming application traffic. Elastic Load Balancing detects unhealthy instances within a pool and automatically reroutes traffic to healthy instances until the unhealthy instances have been restored. You can enable Elastic Load Balancing within a single Availability Zone or across multiple zones for even more consistent application performance. Amazon CloudWatch can be used to capture a specific Elastic Load Balancer’s operational metrics, such as request count and request latency, at no additional cost beyond Elastic Load Balancing fees.

Role Played in Auto Scaling

Let's say that you want to make sure that the number of healthy Amazon EC2 instances behind an Elastic Load Balancer is never fewer than two. You can use Auto Scaling to set this condition, and when Auto Scaling detects that this condition has been met, it automatically adds the requisite amount of Amazon EC2 instances to your Auto Scaling Group. Or, if you want to make sure that you add Amazon EC2 instances when latency of any one of your Amazon EC2 instances exceeds 4 seconds over any 15 minute period, you can set that condition, and Auto Scaling will take the appropriate action on your Amazon EC2 instances — even when running behind an Elastic Load Balancer. Auto Scaling works equally well for scaling Amazon EC2 instances whether you're using Elastic Load Balancing or not.

2.5.3 Simple Storage Service (S3)

http://en.wikipedia.org/wiki/Amazon_S3

Amazon S3 is an online storage web service offered by Amazon Web Services. Amazon S3 provides storage through web services interfaces (REST, SOAP, and BitTorrent. Amazon launched S3, its first publicly available web service, in the United States in March 2006 and in Europe in November 2007.

Design

Details of S3's design are not made public by Amazon. According to Amazon, S3's design aims to provide scalability, high availability, and low latency at commodity costs.

S3 is designed to provide 99.999999999% durability and 99.99% availability of objects over a given year. Though there is no service level agreement for durability.

Buckets

S3 stores arbitrary objects (computer files) up to 5 terabytes in size, each accompanied by up to 2 kilobytes of metadata. Objects are organized into *buckets* (each owned by an Amazon Web Services or AWS account), and identified within each bucket by a unique, user-assigned key. Amazon Machine Images (AMIs) which are modified in the Elastic Compute Cloud (EC2) can be exported to S3 as bundles.

Buckets and objects can be created, listed, and retrieved using either a REST-style HTTP interface or a SOAP interface. Additionally, objects can be downloaded using the HTTP GET interface and the BitTorrent protocol.

Hosting an entire Website

As of February 18, 2011, Amazon S3 provides options to host static websites with Index document support and error document support. This support was added as a result of user requests dating at least to 2006. For example, suppose that Amazon S3 was configured with CNAME records to host http://subdomain.example.com/. In the past, a visitor to this URL would find only an XML-formatted list of objects instead of a general landing page (e.g., index.html) to accommodate casual visitors. Now, however, websites hosted on S3 may designate a default page to display, and another page to display in the event of a partially invalid URL. However, the current domain registration infrastructure only allows a subdomain to be hosted this way, not a second level domain. That is, subdomain.example.com can be hosted, but not example.com. One may use an A record pointing to the S3 server, but this method is not documented by Amazon.

Chapter 3

Nimbus and FutureGrid Project

3.1 Introduction to Nimbus

Nimbus is an open-source toolkit that, once installed on a cluster, provides an infrastructure as a service cloud to its client via Web Services Resource Framework (WSRF-based) or Amazon EC2 Web Services Description Language (WSDL) web service APIs. Nimbus supports the Xen hypervisor or KVM and virtual machine scheduler's portable batch system (PBS) and Oracle/sun Grid engine (SGE). It allows deployment of self-configured virtual clusters via contextualization. It is configurable with respect to scheduling, networking leases, and usage accounting.

Now this above discussion has open a whole new debate and many new words that need an explanation, but to make things simple we would not discuss all of these terms as for my work, we don't need to go in detail of all these technologies.

We need to know only that Nimbus is an open source project focused on cloud computing, it is built around three goals targeting three different communities:

- Enable resource owners to provide their resources as an infrastructure cloud
- Enable cloud users to access infrastructure cloud resources more easily
- Enable scientists and developers to extend and experiment with both sets of capabilities.

The Nimbus project has been created by an international collaboration of open source contributors and institutions. The first goal is realized by the Nimbus Infrastructure (the Workspace Service and Cumulus components providing a compute and storage cloud, respectively), the second by the Nimbus Platform (e.g., the Context Broker and cloudinit.d tools), and the third by strongly supporting open source development practices via modular, extensible code and engagement with open source developers.

3.2 Nimbus Platform

Nimbus Platform is an integrated set of open source tools that allow users to easily leverage "Infrastructure-as-a-Service" (IaaS) cloud computing systems. This includes application instantiation, configuration, monitoring, and repair.

3.3 Nimbus Infrastructure

Nimbus Infrastructure is a set of open source tools that together provide an "Infrastructure-as-a-Service" (IaaS) cloud computing solution. Our mission is to evolve the infrastructure with emphasis on the needs of science, but many non-scientific use cases are supported as well.

Nimbus Infrastructure allows a client to lease remote resources by deploying virtual machines (VMs) on those resources and configuring them to represent an environment desired by the user.

3.4 About Nimbus

http://www.nimbusproject.org/about/

Nimbus is an open-source toolkit focused on providing Infrastructure-as-a-Service (IaaS) capabilities to the scientific community. To achieve this we focus on three goals:

- ***Enable providers of resources to build private or community IaaS clouds***. *The* Nimbus Workspace Service provides an implementation of a compute cloud allowing users to lease computational resources by deploying virtual machines (VMs) on those resources. A complementary tool, Cumulus, provides an implementation of a quota-based storage cloud. Cumulus is designed for scalability and allows providers to configure multiple storage cloud implementations.

- ***Enable users to use IaaS clouds***. An example of a tool in this area is the Nimbus Context Broker, which creates a common configuration and security context across resources provisioned from potentially multiple clouds. Nimbus also offers scaling tools allowing users to automatically scale across multiple distributed providers. We call these tools "sky computing tools" as they often operate in a multi-cloud environment combining private and public cloud capabilities.

- ***Enable developers to extend experiment and customize IaaS***. To achieve this goal we provide a high-quality, highly configurable and extensible open source implementation. For example, the Workspace Service can be configured to support different virtualization implementations (Xen or KVM), resource management options (including schedulers such as PBS), interfaces (including compatibility with Amazon EC2), and many other options. Nimbus components are regularly acceptance tested which allows developers to easily extend them. Over time, the project has attracted a community of open source contributors and committers.

Combining those tools and capabilities in different ways allows users to rapidly develop custom community-specific solutions. For example, one community might want to use Nimbus IaaS to configure a private or community cloud while another may want to focus on augmenting resources of an already existing cloud with resources from various community and public clouds.

3.5 Introduction to FutureGrid

https://portal.futuregrid.org/

FutureGrid Project provides a capability that makes it possible for researchers to tackle complex research challenges in computer science related to the use and security of grids and clouds. These include topics ranging from authentication, authorization, scheduling, virtualization, middleware design, interface design and cyber security, to the optimization of grid-enabled and cloud-enabled computational schemes for researchers in astronomy, chemistry, biology, engineering, atmospheric science and epidemiology. The project team will provide a significant new experimental computing grid and cloud test-bed, named FutureGrid, to the research community, together with user support for third-party researchers conducting experiments on FutureGrid.

The test-bed will make it possible for researchers to conduct experiments by submitting an experiment plan that is then executed via a sophisticated workflow engine, preserving the provenance and state information necessary to allow reproducibility.

The test-bed includes a geographically distributed set of heterogeneous computing systems, a data management system that will hold both metadata and a growing library of software images, and a dedicated network allowing isolatable, secure experiments. The test-bed will support virtual machine-based environments, as well as native operating systems for experiments aimed at minimizing overhead and maximizing performance. The project partners will integrate existing open-source software packages to create an easy-to-use software environment that supports the instantiation, execution and recording of grid and cloud computing experiments.

One of the goals of the project is to understand the behavior and utility of cloud computing approaches. Researchers will be able to measure the overhead of cloud technology by requesting linked experiments on both virtual and bare-metal systems. FutureGrid will enable US scientists to develop and test new approaches to parallel, grid and cloud computing, and compare and collaborate with international efforts in this area. The FutureGrid project will provide an experimental platform that accommodates batch, grid and cloud computing, allowing researchers to attack a range of research questions associated with optimizing, integrating and scheduling the different service models. The FutureGrid also provides a test-bed for middleware development and, because of its private network, allows middleware researchers to do controlled experiments under different network conditions and to test approaches to middleware that include direct interaction with the network control layer. Another component of the project is the development of benchmarks appropriate for grid computing, including workflow-based benchmarks derived from applications in astronomy, bioinformatics, seismology and physics

The FutureGrid will form part of NSF's high-performance cyber infrastructure. It will increase the capability of the XSEDE to support innovative computer science research requiring access to lower levels of the grid software stack, the networking software stack, and to virtualization and workflow orchestration tools.

Education and broader outreach activities include the dissemination of curricular materials on the use of FutureGrid, pre-packaged FutureGrid virtual machines configured for particular course modules, and educational modules based on virtual appliance networks and social networking technologies that will focus on education in networking, parallel computing, virtualization and distributed computing. The project will advance education and training in distributed computing at academic institutions with less diverse computational resources. It will do this through the development of instructional resources that include preconfigured environments that provide students with sandboxed virtual clusters. These can be used for teaching courses in parallel, cloud, and grid computing. Such resources will also provide academic institutions with a simple opportunity to experiment with cloud technology to see if such technology can enhance their campus resources. The FutureGrid project leverages the fruits of several software development projects funded by the National Science Foundation and the Department of Energy.

3.6 Using IaaS Clouds on FutureGrid Project

Infrastructure-as-a-Service (IaaS) cloud computing encompasses techniques that have driven major recent advances in information technology supporting elastic, on-demand, "pay as you go" computing as a service. Key technologies behind IaaS cloud computing are resource virtualization, as well as cloud middleware that enables the management of clusters of virtualized resources through service interfaces.

The FutureGrid test bed provides capabilities that allow users to experiment with open-source cloud middleware and virtualization platforms, and there are different ways you may want to use these platforms in the test bed.

Following are three open-sourced services packages, offered by FutureGrid to its users.

1. Nimbus Clouds:

We have already discussed it in detail in previous sections of the chapter.

2. OpenStack Clouds

OpenStack is a recently open-sourced, IaaS cloud computing platform founded by Rackspace Hosting and NASA, and used widely in industry. It includes three components: Compute (Nova), Object Storage (Swift), and Image Service (Glance). OpenStack Nova supports an Amazon Web Services (AWS) complaint EC2-based web service interface for interacting with the Cloud service, and can be used with the same client-side "eucatools" that is used with Eucalyptus.

FutureGrid currently features the OpenStack Nova compute cloud. OpenStack Nova in FutureGrid is useful if you are interested in experiments within a cloud, and in comparison of cloud middleware stacks.

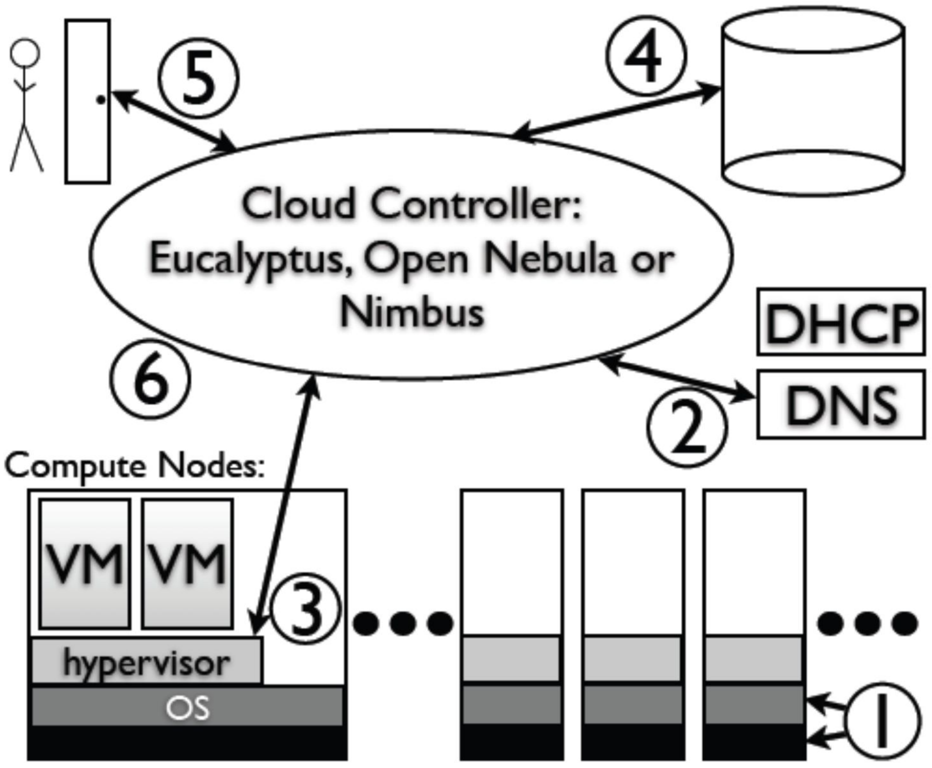


Fig : Abstract Cloud Architecture
The layout of the different parts of a generic cloud computing system. Note that the role of the cloud control software is to coordinate all of these pieces and to provide sufficient abstraction so that a user can simply request VMs with minimal concern for how these components must interact. The numbers in this diagram correspond to the components as we list them in this section.

3. Eucalyptus Clouds

Eucalyptus is an open-source software platform that implements IaaS-style cloud computing. Eucalyptus provides an Amazon Web Services (AWS) complaint EC2-based web service interface for interacting with the Cloud service. Additionally, Eucalyptus provides services such as the AWS Complaint Walrus and a user interface for managing users and images.

Eucalyptus is also available on distributed FutureGrid resources. Eucalyptus in FutureGrid is useful if you are interested in experiments within a cloud, across clouds, and in comparison of cloud middleware stacks.

3.7 Nimbus on FutureGrid

Nimbus is an open source service package that allows users to run virtual machines on FutureGrid hardware. You can easily upload your own VM image or customize an image provided by us. When you boot a VM, it is assigned a public IP address (and/or an optional private address); you are authorized to log in as root via SSH. You can then run services, perform computations, and configure the system as desired. After using and configuring the VM, you can save the modified VM image back to the Nimbus image repository.

Nimbus is installed on four FutureGrid clusters:

1. **Hotel** (University of Chicago)
   41 nodes, 328 cores
2. **Foxtrot** (University of Florida)
   26 nodes, 208 cores
3. **Sierra** (San Diego Supercomputer Center)
   18 nodes, 144 cores
4. **Alamo** (Texas Advanced Computing Center)
   15 nodes, 120 cores

By default, users are limited to running 16 VMs simultaneously and claiming two cores per VM. If you have a good reason for this limitation to be lifted for your account, contact https://portal.futuregrid.org/help

All FutureGrid users are allowed access to Nimbus on all sites.

Chapter 4

Virtual Machine Controller

4.1 Preface

Now this is the Most Important part of my thesis. This chapter is the reason I had given so much theoretical background in the previous three chapters. You will realize in this chapter that the theoretical knowledge that was covered before is very useful and fruitful.

In this chapter I would explain in detail the step by step approach for my work. Some of the conceptual background that will be used in this chapter has already been covered in the previous chapters.

4.2 Problem Formulation

4.2.1 Background

Now on futuregrid, using the remote resources for example the service S3 or EC2, one really needs to have a sound knowledge of all these technologies and services in order to use them. Now it's a long procedure before you get to use all these services on futuregrid. And it's completely not user friendly.

For example In order to start an EC2 instance on futuregrid using nimbus platform, one had to follow a series of long steps that are mentioned below:

i) First step is to download the nimbus client

ii) Unpack the archive into your system, and try to login into the hotel machine, and this take a long journey and series of terminal commands to develop a connection with the hotel machine.

iii) Now once you are login to the hotel machine, you can get a List of Available Images by the terminal command as

```
$ ./bin/cloud-client.sh –list
```

iv) Now from the list you can select from one of the images which satisfy your needs and you can run the image. As the output you would see something as

```
Launching workspace.

Workspace Factory Service:
    https://svc.uc.futuregrid.org:8443/wsrf/services/WorkspaceFactorySer
vice

Creating workspace "vm-013"... done.


       IP address: 149.165.148.118
         Hostname: vm-148-118.uc.futuregrid.org
       Start time: Tue Nov 20 05:47:49 CET 2012
    Shutdown time: Tue Nov 20 07:47:49 CET 2012
 Termination time: Tue Nov 20 07:57:49 CET 2012

Waiting for updates.


"vm-013" reached target state: Running

Running: 'vm-013'
```

Wow, this is quite much confusing, not really, if you see carefully Its tells that successfully we have a running workspace that is our virtual machine is running.

v) Now the fifth and the important part , we need to ssh/telnet into the root, so we also had to manually ssh into the root of our virtual machine. So we do as

```
$    ssh root@vm-148-118.uc.futuregrid.org
```

vi) Now we could play around in the root, but to reach this step we had to really go a long way in, and follow a series of long steps. Now this was only for just starting a virtual machine. And none of other functions were done like for example starting multiple virtual machines or load balancing or controlling our instances like CPU usage, memory etc.

So in conclusion, we surely got a problem here. One had to find another way around to figure out how to avoid all these steps, and we could do a lot of work, in less time.

4.2.2 Using Boto API

Now people realizing the problem, start thinking about writing an API, so that they do not have to go through all the hectic procedure to do use simple services like EC2 and S3. They gave Boto as an add-on to python. It's a library of python.

Now boto is like a magic, like all the steps needed to develop a connection and evoke an EC2 connection can just be done in one step:

ec2conn = boto.connect_ec2()

And for having the list of all available images we could just do as follows:

images=ec2conn.get_all_images() // if you print images, you will get a List of available images.

Similarly for a lot services, I have already said that currently Boto supports a lot of Services and Products of for Nimbus open-source Cloud Platform on Futuregrid. I am not going to name them all as it is not part of the report.

4.3 Need for Automation

Now as with boto it seems the things, have now become pretty much easy, but as we say, humans are humans and we are dumb, and we hate to do things which are long and take some steps in order to be done.

Like for example in order to switch-on a virtual machine on hotel machine on futuregrid, even using boto I had to follow few steps to do that which are as follows

i) Develop and EC2 connection. (ec2conn=boto.connect_ec2())

ii) Run a virtual machine

```
    # now run the VM
    reservation = ec2conn.run_instances("hello-
cloud",security_groups=['default'])
    instance = reservation.instances[0]
```

```
# poll for status
 while instance.state != 'running':
 print 'instance is %s' % instance.state
 time.sleep(30)
 instance.update()
```

Now as you see, we still had a long journey to start a simple Virtual machine, and this all is Boto (an add-on to python), so what we need a Controller which can automate a lot of these services and can make life easier. Now what this Controller should do, now it should ask user about "what you want" and user can say start a Virtual machine for me, and this controller just do all these steps for him, and the user is happy to save his time. Now more details about the controller will be covered in the remaining chapter.

4.4 Limitations of the Controller

Now it is not GUI (Graphical User Interface). Now I know there are now very big companies into this business of designing new and high-tech fancy Controller, to control the remote resources. And they also offer a numbers of features. But one had to pay a lot to get them.

4.5 In detail My Virtual Machine Controller

Now under this heading, I would explain everything, the services my Controller allows the user to do, the steps taken to design it, and in some cases the pseudo code for the service.

Now the virtual machine controller offers its user in total seven services, I am going to make a heading for each offered service.

i) List
ii) Start
iii) Stop
iv) Live_instances
v) Image_attributes
vi) Live_attributes
vii) Simple storage service (S3)
viii) Quit everything

Now one by one I am going to explain all these services. And tell about the detailed procedure taken in order to fulfill the service requirement.

4.5.1 List

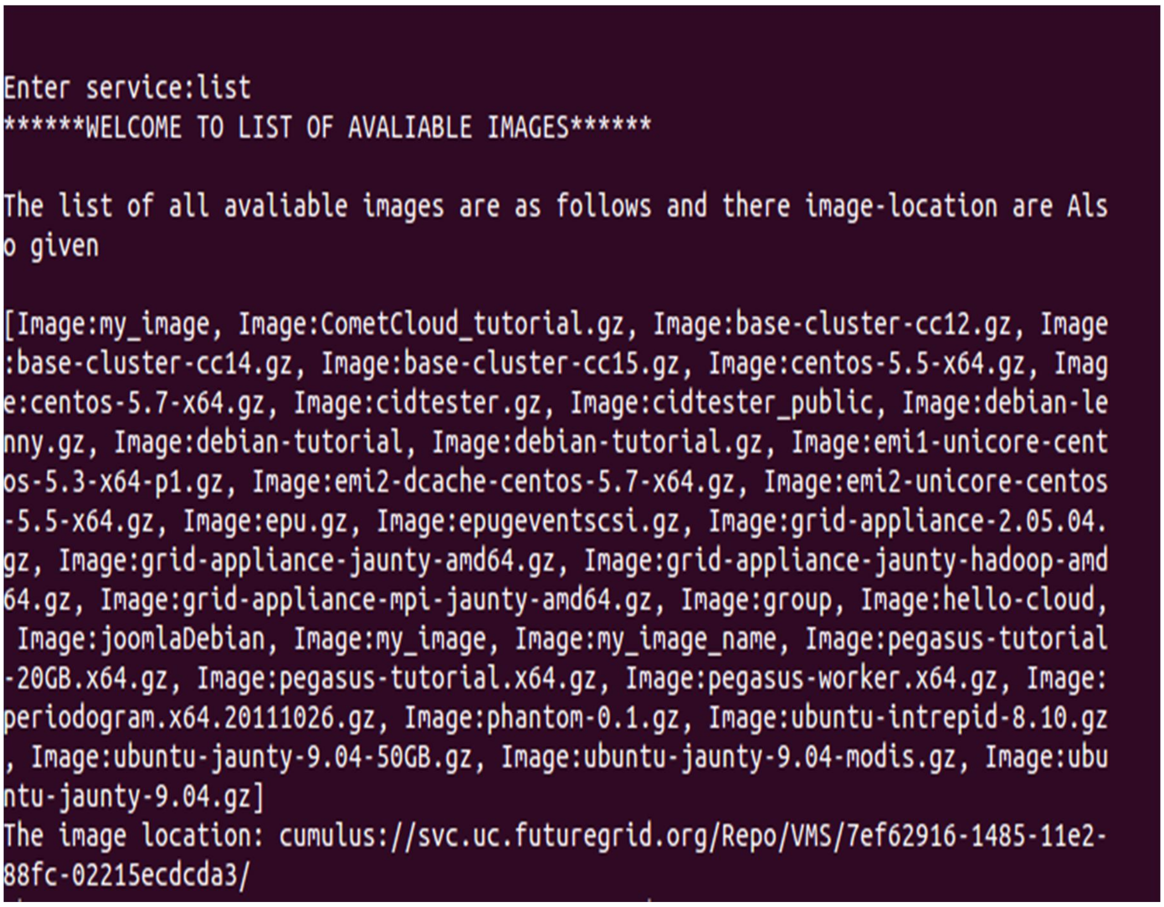
Now this service offers, to list all the available images of the EC2 instances. Among the list includes the pre-configured images made available by futuregrid, or it may also include your own configured image if you have upload your image to the work-space under your credentials.

So in short, this service is important to evoke first as after invoking this service, an EC2 and S3 connections have been developed and necessary credentials requirements have been fulfilled and now you can proceed on with the later services.

The result for this service:

```
[Image:my_image, Image:CometCloud_tutorial.gz, Image:base-cluster-cc12.gz,
Image:base-cluster-cc14.gz, Image:base-cluster-cc15.gz, Image:centos-5.5-x64.gz,
Image:centos-5.7-x64.gz, Image:cidtester.gz, Image:cidtester_public, Image:debian-
lenny.gz, Image:debian-tutorial, Image:debian-tutorial.gz, Image:emi1-unicore-
centos-5.3-x64-p1.gz, Image:emi2-dcache-centos-5.7-x64.gz, Image:emi2-unicore-
centos-5.5-x64.gz, Image:epu.gz, Image:epugeventscsi.gz, Image:grid-appliance-
2.05.04.gz, Image:grid-appliance-jaunty-amd64.gz, Image:grid-appliance-jaunty-
hadoop-amd64.gz, Image:grid-appliance-mpi-jaunty-amd64.gz, Image:group,
Image:hello-cloud, Image:joomlaDebian, Image:my_image, Image:my_image_name,
Image:pegasus-tutorial-20GB.x64.gz, Image:pegasus-tutorial.x64.gz,
Image:pegasus-worker.x64.gz, Image:periodogram.x64.20111026.gz,
Image:phantom-0.1.gz, Image:ubuntu-intrepid-8.10.gz, Image:ubuntu-jaunty-9.04-
50GB.gz, Image:ubuntu-jaunty-9.04-modis.gz, Image:ubuntu-jaunty-9.04.gz]
```

The Terminal snapshot for this option is:



```
Enter service:list
******WELCOME TO LIST OF AVALIABLE IMAGES******

The list of all avaliable images are as follows and there image-location are Als
o given

[Image:my_image, Image:CometCloud_tutorial.gz, Image:base-cluster-cc12.gz, Image
:base-cluster-cc14.gz, Image:base-cluster-cc15.gz, Image:centos-5.5-x64.gz, Imag
e:centos-5.7-x64.gz, Image:cidtester.gz, Image:cidtester_public, Image:debian-le
nny.gz, Image:debian-tutorial, Image:debian-tutorial.gz, Image:emi1-unicore-cent
os-5.3-x64-p1.gz, Image:emi2-dcache-centos-5.7-x64.gz, Image:emi2-unicore-centos
-5.5-x64.gz, Image:epu.gz, Image:epugeventscsi.gz, Image:grid-appliance-2.05.04.
gz, Image:grid-appliance-jaunty-amd64.gz, Image:grid-appliance-jaunty-hadoop-amd
64.gz, Image:grid-appliance-mpi-jaunty-amd64.gz, Image:group, Image:hello-cloud,
 Image:joomlaDebian, Image:my_image, Image:my_image_name, Image:pegasus-tutorial
-20GB.x64.gz, Image:pegasus-tutorial.x64.gz, Image:pegasus-worker.x64.gz, Image:
periodogram.x64.20111026.gz, Image:phantom-0.1.gz, Image:ubuntu-intrepid-8.10.gz
, Image:ubuntu-jaunty-9.04-50GB.gz, Image:ubuntu-jaunty-9.04-modis.gz, Image:ubu
ntu-jaunty-9.04.gz]
The image location: cumulus://svc.uc.futuregrid.org/Repo/VMS/7ef62916-1485-11e2-
88fc-02215ecdcda3/
```

4.5.2 Start

Now this service in a bigger picture as the name tells you, will start a virtual machine for the user.

But it just not only start a virtual machine but also do other things for the user, and in addition will also give a user a number of options to select from. Let me go on step by step.

Now when the user invoke the start service, he will be asked by the controller to select from the two options:

i) Select an image from the preconfigured or listed images
ii) Upload am image of your own.

Preconfigured Images

Now if the user selects option one, as a result of it, the user will be asked to enter the name of the pre-configured available image name he is interested in. Now after the image name is entered, than everything is done, and you would see on the screen the status of your virtual machine. As the virtual machine is running state, the menu will print you the IP address and the host name.

Now after above all again the controller will ask you to select from few options.

i) Remote access to the root
For remote access to the root of your started virtual machine, you had to choose this option, as it will direct you to the root of your started instance.

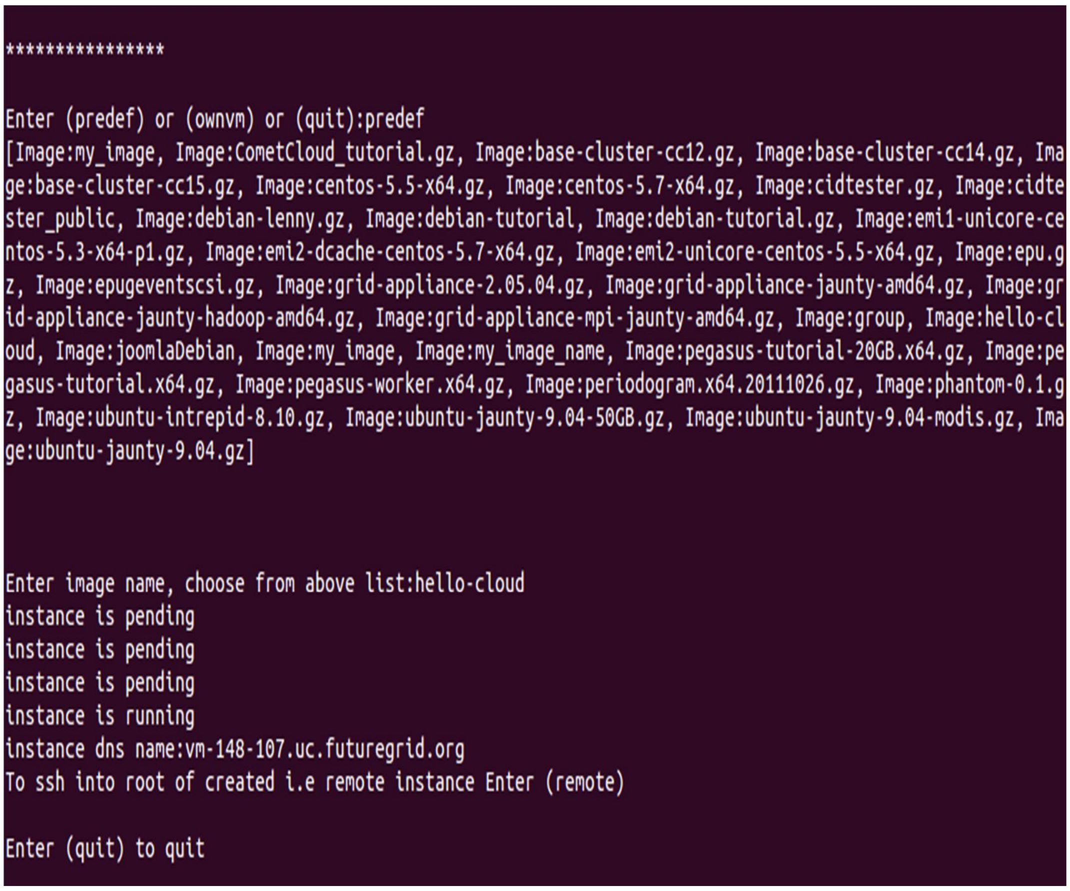
ii) Quit

Now this option does a number of things, first it will quit out of the preconfigured images option, and second it will tell you the way as to how you can start multiple instances, and that is very simple repeat the same process again as many times as many instances you want to start.

The output for this option is as follows:

```
Enter image name, choose from above list:hello-cloud
instance is pending
instance is running
instance dns name:vm-148-109.uc.futuregrid.org
To ssh into root of created i.e remote instance Enter (remote)
```

The snapshot of the terminal is as follows:



```
***************

Enter (predef) or (ownvm) or (quit):predef
[Image:my_image, Image:CometCloud_tutorial.gz, Image:base-cluster-cc12.gz, Image:base-cluster-cc14.gz, Ima
ge:base-cluster-cc15.gz, Image:centos-5.5-x64.gz, Image:centos-5.7-x64.gz, Image:cidtester.gz, Image:cidte
ster_public, Image:debian-lenny.gz, Image:debian-tutorial, Image:debian-tutorial.gz, Image:emi1-unicore-ce
ntos-5.3-x64-p1.gz, Image:emi2-dcache-centos-5.7-x64.gz, Image:emi2-unicore-centos-5.5-x64.gz, Image:epu.g
z, Image:epugeventscsi.gz, Image:grid-appliance-2.05.04.gz, Image:grid-appliance-jaunty-amd64.gz, Image:gr
id-appliance-jaunty-hadoop-amd64.gz, Image:grid-appliance-mpi-jaunty-amd64.gz, Image:group, Image:hello-cl
oud, Image:joomlaDebian, Image:my_image, Image:my_image_name, Image:pegasus-tutorial-20GB.x64.gz, Image:pe
gasus-tutorial.x64.gz, Image:pegasus-worker.x64.gz, Image:periodogram.x64.20111026.gz, Image:phantom-0.1.g
z, Image:ubuntu-intrepid-8.10.gz, Image:ubuntu-jaunty-9.04-50GB.gz, Image:ubuntu-jaunty-9.04-modis.gz, Ima
ge:ubuntu-jaunty-9.04.gz]



Enter image name, choose from above list:hello-cloud
instance is pending
instance is pending
instance is pending
instance is running
instance dns name:vm-148-107.uc.futuregrid.org
To ssh into root of created i.e remote instance Enter (remote)

Enter (quit) to quit
```

Now there are other ways also to have a remote access to the root of you r started virtual machine, one easy approach will be to simply get first the hostname of your instance. And that you can get easily by seeing every time when you start an instance, as the host name is always printed for each instance you start. Once you get the host id for your virtual machine now you can remotely enter the root of that machine by simply giving at terminal the command ($ ssh root@hostname).

Upload an image of your own

Now this option is pretty interesting, as you can upload your own configured image. Now how we do it, for this the user have to be a little good with programming, as he needs to give the path of of his own configured image file in the boto module.

Now as far as I am concerned, I already have my own image configured file is already up on the platform, and when the user gives the command of list all the available images, also my image is shown, as for sure user is using my credentials right now, so it always logged in into my work space in futuregrid.

So again the steps are, you first give the path of your image file, in the required space and then you need to wait for a while for the image file to be uploaded. Now this might take some time, as depending on the size of the image file. As for mine it took me more than 20 minutes to upload an image of approximately 530MB, so you can decide how much time it will take for your image.

Once you have uploaded your own image. Then the story is quite the same as above, you can remotely enter into the root of your virtual machine and so on.

4.5.3 Stop

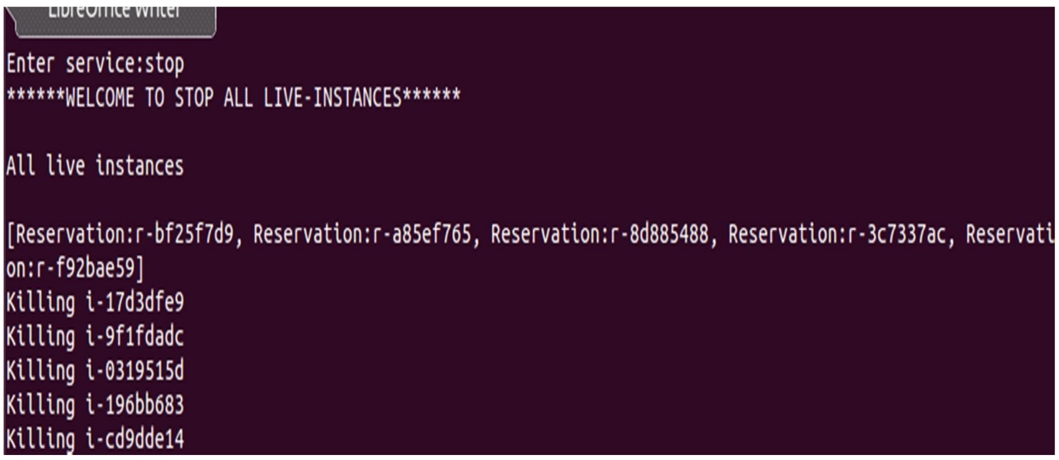
Now this service stops all the live virtual machines that you have started, it will first give you a list of live instances and then show there host id and instances id and later will terminate them.

This step is very important, that needs to be done every time you want to totally quit the controller. As this will kill all the live instances that you no longer want to use, and it will save your money as you are been billed for your instances.

Now the result of this option is as follows:

```
Enter service:stop
******WELCOME TO STOP ALL LIVE-INSTANCES******

All live instances

[Reservation:r-bf25f7d9, Reservation:r-a85ef765, Reservation:r-8d885488,
Reservation:r-3c7337ac, Reservation:r-f92bae59]
Killing i-17d3dfe9
Killing i-9f1fdadc
Killing i-0319515d
Killing i-196bb683
Killing i-cd9dde14
```

Now the screenshot of the terminal for stop all the instances option is as follows:



```
Enter service:stop
******WELCOME TO STOP ALL LIVE-INSTANCES******

All live instances

[Reservation:r-bf25f7d9, Reservation:r-a85ef765, Reservation:r-8d885488, Reservation:r-3c7337ac, Reservati
on:r-f92bae59]
Killing i-17d3dfe9
Killing i-9f1fdadc
Killing i-0319515d
Killing i-196bb683
Killing i-cd9dde14
```

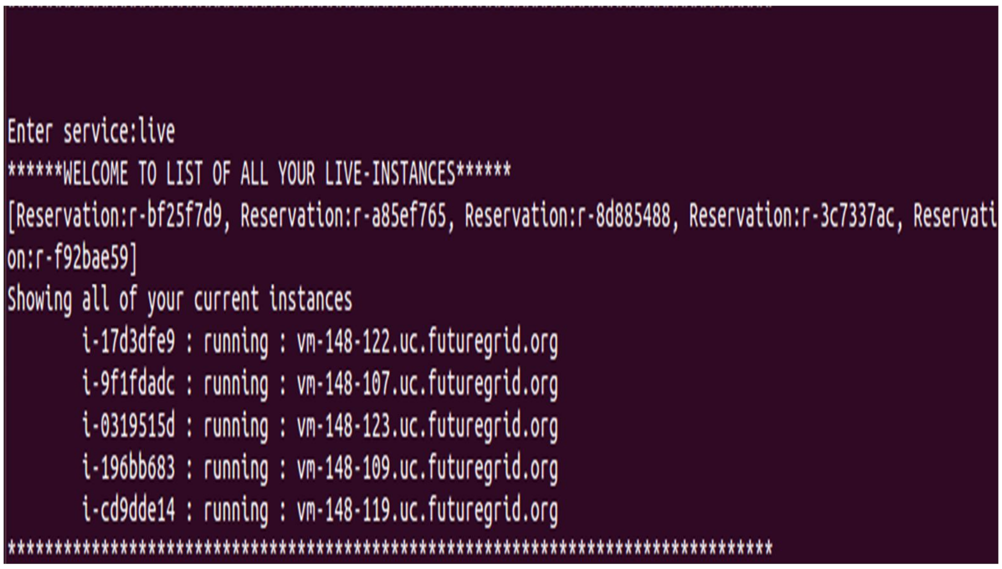
4.5.4 live_instances

This service will enlist all the live instances, that you started. It will also give your instances id and there host id, now from the host id you can remotely enter into the root of any of your started instance by simply from the terminal ($ ssh root@hostname).

Now In order to show the output I already had started 5 instances, of hello-cloud image, so the output for this option is as:

```
Enter service:live
******WELCOME TO LIST OF ALL YOUR LIVE-INSTANCES******
[Reservation:r-bf25f7d9, Reservation:r-a85ef765, Reservation:r-8d885488,
Reservation:r-3c7337ac, Reservation:r-f92bae59]
Showing all of your current instances
   i-17d3dfe9 : running : vm-148-122.uc.futuregrid.org
   i-9f1fdadc : running : vm-148-107.uc.futuregrid.org
   i-0319515d : running : vm-148-123.uc.futuregrid.org
   i-196bb683 : running : vm-148-109.uc.futuregrid.org
   i-cd9dde14 : running : vm-148-119.uc.futuregrid.org
```

The terminal snapshot for this option is as:



```
Enter service:live
******WELCOME TO LIST OF ALL YOUR LIVE-INSTANCES******
[Reservation:r-bf25f7d9, Reservation:r-a85ef765, Reservation:r-8d885488, Reservation:r-3c7337ac, Reservati
on:r-f92bae59]
Showing all of your current instances
        i-17d3dfe9 : running : vm-148-122.uc.futuregrid.org
        i-9f1fdadc : running : vm-148-107.uc.futuregrid.org
        i-0319515d : running : vm-148-123.uc.futuregrid.org
        i-196bb683 : running : vm-148-109.uc.futuregrid.org
        i-cd9dde14 : running : vm-148-119.uc.futuregrid.org
*********************************************************************************
```

4.5.5 Image_Attributes and Live_instance_Attributes

Now here I am jointly enlisted two services.

The first is image_attributes, and it's as by its name very clear will give you attributes of the image available. Now it will ask you the name of the image and attribute you are interested in knowing.

The second service is live_instance_attributes, also this gives you attributes, but of the instances that you started. In the start it will ask you the instance id of the virtual machine you are interested in knowing the attributes.

4.5.6 Simple Storage Service (S3)

Now I have already told you about what is S3, so now this service is quite simple, it first enlist all the available buckets that you created. And if you are first time user, it will create a bucket for you. But in that case it will ask you the name of the bucket you want to select.

Now after above step, you need to give two more things, first the name of the file you want to upload from and the name of the file you want to download to.

Now it will take some time to upload your contents from the file.

4.5.7 Quit Everything

Now this is obviously not a service, but it will quit from the controller. And will show you what are the necessary steps you need to take before you get out of the controller.

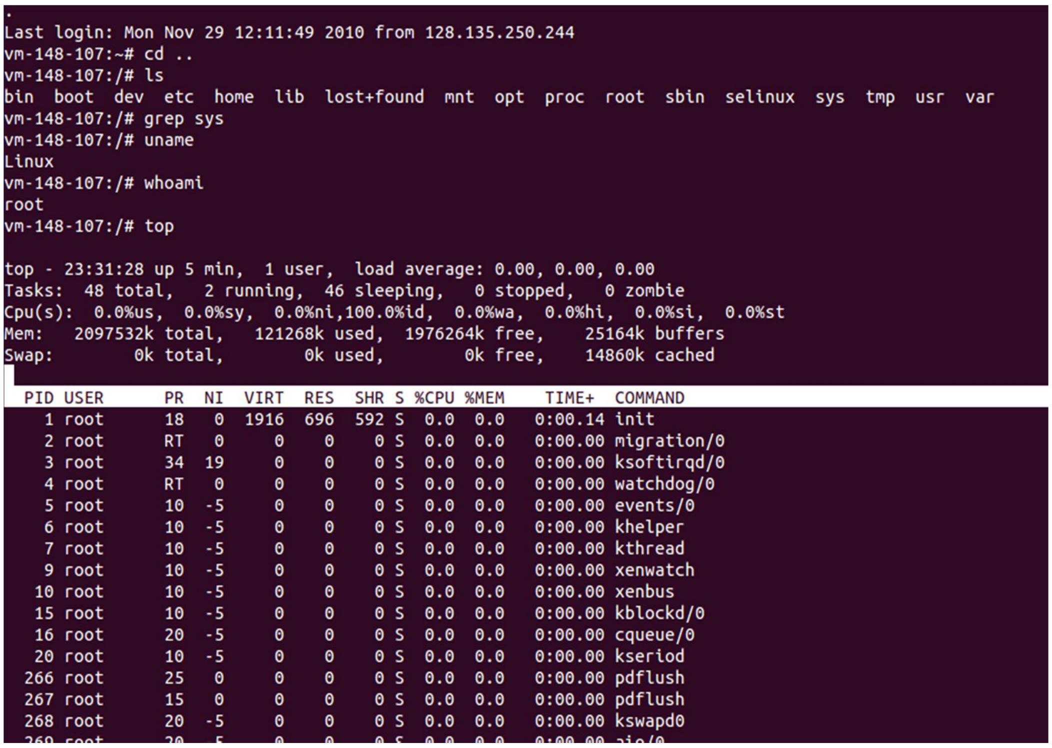
4.6 Telnet/SSH into root@hostname

Now under this heading, we will play around a little with the root of our virtual machine. Now how to get to the root, I have already mentioned a number of ways, and also it is a built in service in our virtual machine controller.

Now the point is once you get into the root of your virtual machine what u would like to do. Now there are many possibilities, but we are interested in the system information of our virtual machine and for that we would like to know the processes, RAM, CPUINFO, free memory, and related stuff. So we begin as:

Now first we would like to know the top process running in our virtual machine. Note here that we already in the root of our virtual machine.

```
vm-148-107:/# uname
Linux
vm-148-107:/# whoami
root
vm-148-107:/# top

top - 23:32:02 up 5 min,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  48 total,   1 running,  47 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   2097532k total,   121392k used,  1976140k free,    25212k buffers
Swap:        0k total,        0k used,        0k free,    14860k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU
%MEM    TIME+  COMMAND
    1 root      18   0  1916  696  592 S  0.0  0.0   0:00.14 init
    2 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    4 root      RT   0     0    0    0 S  0.0  0.0   0:00.00
watchdog/0
    5 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 events/0
    6 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 khelper
    7 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kthread
    9 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 xenwatch
   10 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 xenbus
   15 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kblockd/0
   16 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 cqueue/0
   20 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kseriod
  266 root     25   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  267 root     15   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  268 root     20  -5     0    0    0 S  0.0  0.0   0:00.00
kswapd0
  269 root     20  -5     0    0    0 S  0.0  0.0   0:00.00 aio/0
  275 root     10  -5     0    0    0 S  0.0  0.0   0:00.00 cifsoplockd
```

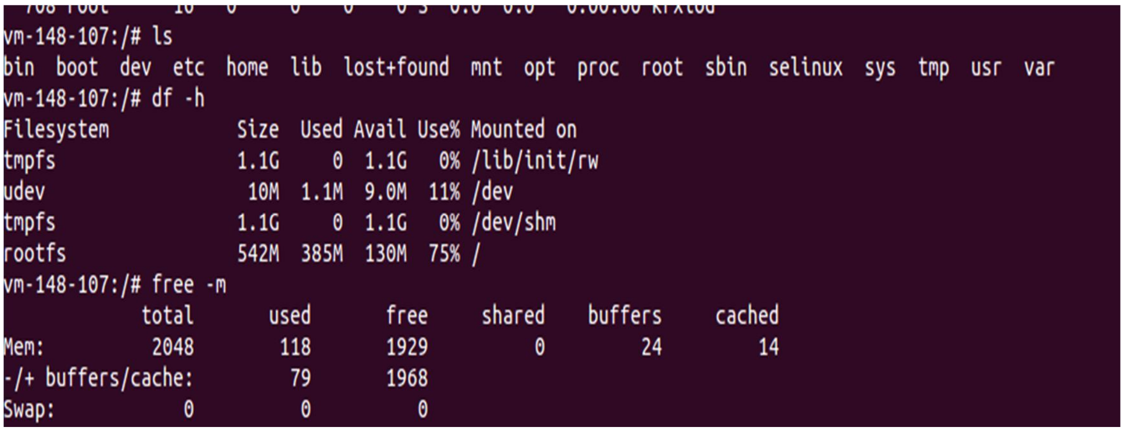


```
.
Last login: Mon Nov 29 12:11:49 2010 from 128.135.250.244
vm-148-107:~# cd ..
vm-148-107:/# ls
bin  boot  dev  etc  home  lib  lost+found  mnt  opt  proc  root  sbin  selinux  sys  tmp  usr  var
vm-148-107:/# grep sys
vm-148-107:/# uname
Linux
vm-148-107:/# whoami
root
vm-148-107:/# top

top - 23:31:28 up 5 min,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  48 total,   2 running,  46 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   2097532k total,   121268k used,  1976264k free,    25164k buffers
Swap:        0k total,        0k used,        0k free,    14860k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      18   0  1916  696  592 S  0.0  0.0   0:00.14 init
    2 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    4 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    5 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 events/0
    6 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 khelper
    7 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kthread
    9 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 xenwatch
   10 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 xenbus
   15 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kblockd/0
   16 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 cqueue/0
   20 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kseriod
  266 root      25   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  267 root      15   0     0    0    0 S  0.0  0.0   0:00.00 pdflush
  268 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 kswapd0
```

Now after the process we would be interested in knowing the root directories and the memory information, so for that we do as follows

```
vm-148-107:/# ls
bin  boot  dev   etc  home  lib   lost+found  mnt  opt  proc  root  sbin   selinux  sys  t
mp  usr   var
vm-148-107:/# df -h
Filesystem            Size  Used Avail Use% Mounted on
tmpfs                 1.1G     0  1.1G   0% /lib/init/rw
udev                   10M  1.1M  9.0M  11% /dev
tmpfs                 1.1G     0  1.1G   0% /dev/shm
rootfs                542M  385M  130M  75% /
vm-148-107:/# free -m
             total       used       free     shared    buffers     cached
Mem:          2048        118       1929          0         24         14
```

-/+ buffers/cache: 79   1968
Swap: 0   0   0

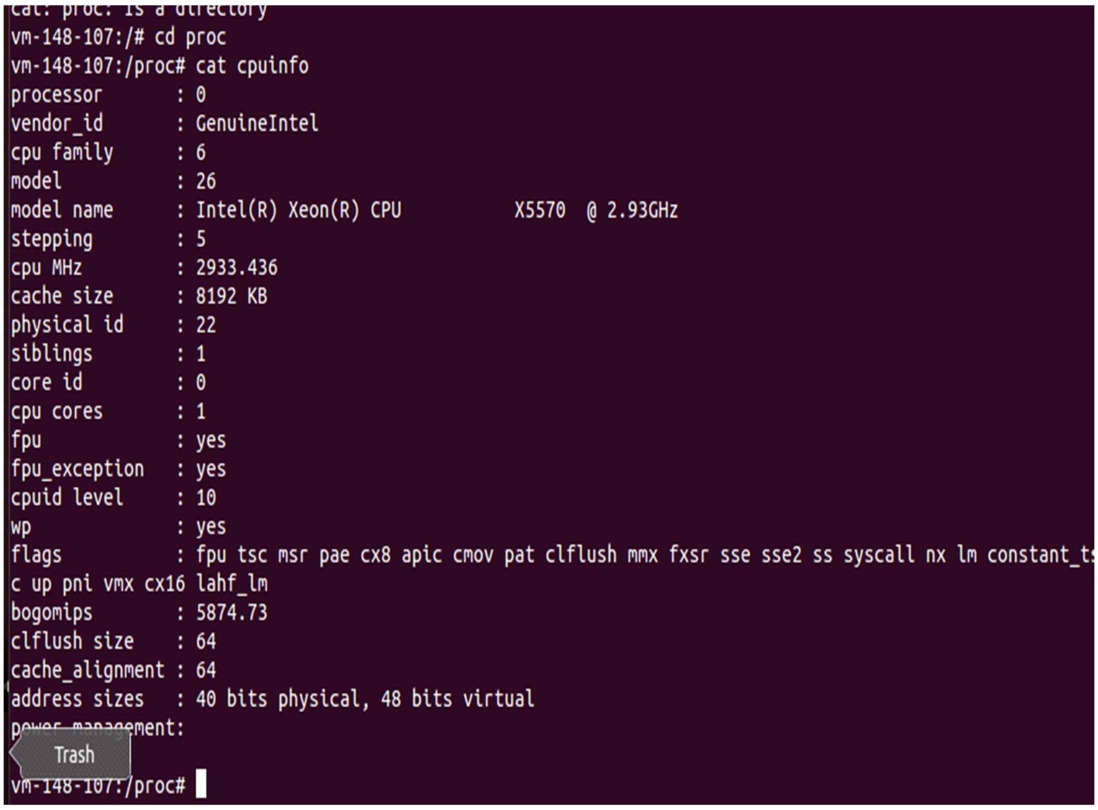


```
vm-148-107:/# ls
bin  boot  dev  etc  home  lib  lost+found  mnt  opt  proc  root  sbin  selinux  sys  tmp  usr  var
vm-148-107:/# df -h
Filesystem            Size  Used Avail Use% Mounted on
tmpfs                 1.1G     0  1.1G   0% /lib/init/rw
udev                   10M  1.1M  9.0M  11% /dev
tmpfs                 1.1G     0  1.1G   0% /dev/shm
rootfs                542M  385M  130M  75% /
vm-148-107:/# free -m
             total       used       free     shared    buffers     cached
Mem:          2048        118       1929          0         24         14
-/+ buffers/cache:         79       1968
Swap:            0          0          0
```

CPUINFO OF VIRTUAL MACHINE

Now we would like to know the best part, i.e CPUINFO, so in our case for our machine we have something as:

```
vm-148-107:/# cd proc
vm-148-107:/proc# cat cpuinfo
processor	: 0
vendor_id	: GenuineIntel
cpu family	: 6
model		: 26
model name	: Intel(R) Xeon(R) CPU           X5570  @ 2.93GHz
stepping	: 5
cpu MHz		: 2933.436
cache size	: 8192 KB
physical id	: 22
siblings	: 1
core id		: 0
cpu cores	: 1
fpu		: yes
fpu_exception	: yes
cpuid level	: 10
wp		: yes
flags		: fpu tsc msr pae cx8 apic cmov pat clflush mmx fxsr sse sse2 ss syscall
nx lm constant_tsc up pni vmx cx16 lahf_lm
bogomips	: 5874.73
clflush size	: 64
cache_alignment	: 64
address sizes	: 40 bits physical, 48 bits virtual
power management:
```



```
cat: proc: is a directory
vm-148-107:/# cd proc
vm-148-107:/proc# cat cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 26
model name      : Intel(R) Xeon(R) CPU           X5570  @ 2.93GHz
stepping        : 5
cpu MHz         : 2933.436
cache size      : 8192 KB
physical id     : 22
siblings        : 1
core id         : 0
cpu cores       : 1
fpu             : yes
fpu_exception   : yes
cpuid level     : 10
wp              : yes
flags           : fpu tsc msr pae cx8 apic cmov pat clflush mmx fxsr sse sse2 ss syscall nx lm constant_t
c up pni vmx cx16 lahf_lm
bogomips        : 5874.73
clflush size    : 64
cache_alignment : 64
address sizes   : 40 bits physical, 48 bits virtual
power management:

vm-148-107:/proc# 
```

4.7 Python + Boto code for my VM Controller

Now after a long discussion finally we get to look at the code. And this is what kept me busy for the last one and half month. And I tell you it looked quite easy once you are done but sure was a lot of pain when things weren't working fine for the controller and there were bugs in the code.

Now here the program is quite self-explanatory and a lot commentated.

```python
import boto
from boto.s3.key import Key
import time
import boto.ec2.connection
from boto.s3.connection import OrdinaryCallingFormat
from boto.s3.connection import S3Connection
from boto.ec2.connection import EC2Connection
from boto.ec2.regioninfo import RegionInfo
from subprocess import call
```

```
access_id="Rzh6HjUNRfhuIeBcxqbc5"
access_secret="hRINe4bEGgMQpiuAXZ9T5QtzwWlODAnA8k5QWSYQOF"
host="svc.uc.futuregrid.org"
port=8444
cumulusport=8888
canonical_id="common"

region = RegionInfo(name="nimbus", endpoint=host)
ec2conn = boto.connect_ec2(access_id, access_secret,
region=region, port=port)
cf = OrdinaryCallingFormat()
s3conn = S3Connection(access_id, access_secret, host=host,
port=cumulusport, is_secure=False, calling_format=cf)



while 1:
    print
"***************************************************************
**********************"
    print "* This is a VM_CONTROLLER.
**\n"
    print "* Enter (list), to get List of avaliable images on
futuregrid nimbus platform   **\n"
    print "* Enter (start) to start an EC2 instance
**\n"
    print "* Enter (stop) for terminate instances
**\n"
    print "* Enter (live) to know about all live instances and
there states                **\n"
    print "* Enter (img_atr) to know the attributes of image
avaliable                     **\n"
    print "* Enter (live_atr)to know attribute of
live_instance                            **\n"
    print "* Enter (s3) to use simple storage service, to
store files.                     **\n"
    print "* Enter (quit) key to exit
**\n"
    print
"***************************************************************
*********************\n\n"




    service=raw_input("Enter service:")


************************************************************


    if service=="list":
        print "******WELCOME TO LIST OF AVALIABLE
IMAGES******\n"
```

```
        print "The list of all avaliable images are as follows 
and there image-location are Also given\n"
        images=ec2conn.get_all_images()
        print images
        image = images[0]
        for image in images:
            print 'The image location: %s'% image.location


*************************************************************


    elif service=="start":
        while 1:
            print "******WELCOME TO START AN EC2 
INSTANCE******\n"
            print "you are starting an EC2 instance\n"
            print "\nEnter (predef) to Launch from already-
existing Images \n"
            print "Enter (ownvm) for own Configured 
my_image\n"
            print "Enter (quit) to quit\n"
            print "******NOTE******\n"
            print "BY ENTERING EVERYTIME IN (predef) AGIAN. WE 
START A NEW INSTANCE\n"
            print "SO TO START MULTIPLE EC2 INSTANCES, ENVOKE 
THIS OPTION MULTIPLE TIMES\n"
            print "****************\n"

            img_op=raw_input("Enter (predef) or (ownvm) or 
(quit):")

            if img_op=="predef":

                images=ec2conn.get_all_images()
                print images
                img_id=raw_input("\n\n\nEnter image name, 
choose from above list:")
                reservation = 
ec2conn.run_instances(image_id=img_id,security_groups=['defaul
t'])
                instance = reservation.instances[0]

                # status updates
                while instance.state != 'running':
                    print 'instance is %s' % instance.state
                    time.sleep(30)
                    instance.update()

                print 'instance is %s' % instance.state
                print 'instance dns name:%s' % 
instance.dns_name
```

```
                """Now here you get All the running instances
dns_name.
                   Now you can control the remote root by just
simple
                   terminal command ( $ ssh root@dns_name ).
                   Or you can ping at ($ PING root@dns_name),
                   one can test if server is active or not"""


                while 1:
                    print "To ssh into root of created i.e
remote instance Enter (remote)\n"
                    print "Enter (quit) to quit\n"

                    sname=raw_input("Enter (remote) or
(quit):")

                    if sname=="remote":

                    # ssh into root, of the remote instance
                    #print root

                        root="root@"+instance.dns_name
                        call(["ssh", root])

                    elif sname=="quit":
                        print "You decided to quit ssh remote
session development"
                        break #out of while
                    else:
                        print "error in keyword\n"



            # Now upload image from own configured VM, And
upload it, using S3.
            elif img_op=="ownvm":

                # upload the file according to the Nimbus
naming convention
                bucket = s3conn.get_bucket("Repo")
                k = boto.s3.key.Key(bucket)
                k.key = "VMS/%s/%s" % (canonical_id,
"my_image")# I have my_image configured by me.

k.set_contents_from_filename('/home/jawad/nimbus-cloud-client-
021/./my_image')


                # now run the VM
                reservation =
ec2conn.run_instances("my_image",security_groups=['default'])
                instance = reservation.instances[0]
                # status updates
                while instance.state != 'running':
                    print 'instance is %s' % instance.state
                    time.sleep(30)
                    instance.update()
                print 'instance is %s' % instance.state
                print 'instance dns name:%s' %
instance.dns_name


                """Now here you get All the running instances
dns_name.
                   Now you can control the remote root by just
simple
                   terminal command ( $ ssh root@dns_name ).
                   Or you can ping at ($ PING root@dns_name),
                   one can test if server is active or not"""



                #print "To ssh into root of created/remote
instance enter remote\n"
                while 1:

                    print "To ssh into root of created i.e
remote instance Enter (remote)\n"
                    print "Enter (quit) to quit\n"

                    sname=raw_input("Enter (remote) or
(quit):")

                    if sname=="remote":
                        # ssh into root, of the remote
instance

                        #print root
                            root="root@"+instance.dns_name
                            call(["ssh", root])


                    elif sname=="quit":
                        print "You decided to quit ssh remote
session development"
                        break #out of while
                    else:
                        print "error in keyword\n"


            elif img_op=="quit":
                print "you have quit the imp_op,REMEMBER TO
TERMINATE ALL YOUR INSTANCES\n"
                print "******GOTO STOP TO TERMINATE ALL YOUR-
INSTANCES******\n"
                break
            else:
                print "error in keyword in imp_op\n"

*************************************************************


    # Now Below is to stop all the running instances

    elif service=="stop":
        print "******WELCOME TO STOP ALL LIVE-
INSTANCES******\n"

        print "All live instances\n"

        all_inst = ec2conn.get_all_instances()
        print all_inst
        # iterate through all the matching instances and stop
them
        for res in all_inst:

            for instance in res.instances:
                print "Killing %s" % (instance.id)
                instance.terminate()


***********************************************************


    elif service=="live":
        print "******WELCOME TO LIST OF ALL YOUR LIVE-
INSTANCES******"

        # get a list of all your running instances
        all_inst = ec2conn.get_all_instances()
        print all_inst

        # iterate through all of the reservations
        print "Showing all of your current instances"
        for res in all_inst:
            # each reservation have a instance:
            for instance in res.instances:
                print "\t%s : %s : %s" % (instance.id,
instance.state, instance.public_dns_name)



                """Now here you get All the running instances
dns_name.
                   Now you can control the remote root by just
simple
                   terminal command ( $ ssh root@dns_name ).
                   Or you can ping at ($ PING root@dns_name),
                   one can test if server is active or not"""
```

```
**************************************************************
    elif service=="img_atr":
        print "The list of all avaliable images are as follows
and there img_id are Also given\n"
        images=ec2conn.get_all_images()
        print images
        #for i in images:
         #   print '\nimages_name: %s and its img_id: %s'% (i,
images.id)

        print "Enter the image you want attributes of\n"
        img_at=raw_input("Enter img_id:")
        att = ec2conn.get_image_attribute(img_at)
        print att

*********************************************************


    elif service=="live_atr":
        all_inst = ec2conn.get_all_instances()
        print all_inst
        print "Enter the image_id you want to see attributes
and attribute you want\n"
        live_at=raw_input("Enter live instance_id:")
        attribute=raw_input("Enter attribute:")
        latt =
ec2conn.get_instance_attribute(live_at,attribute)
        print latt

***********************************************************


    elif service=="s3":
        print "Already existing buckets are\n"
        rs= s3conn.get_all_buckets()

        c=len(rs)
        print 'The no of already existing buckets %s' % c
        for b in rs:
            print "The buckets names are\n"
            print  b.name
            #bucket.delete()
        print "Enter name for bucket\n"
        bucket_name=raw_input("Enter bucket_name:")

        bucket=s3conn.create_bucket(bucket_name)
        k=Key(bucket)
        k.key = 'myfile'
        print "Enter filenames \n"
        cn=raw_input("Enter source_file_name:")
        df=raw_input("Enter des_file_name:")
        k.set_content_from_filename(cn)
        k.get_content_from_filename(df)

**************************************************************

    elif service=="quit":
        print "You have quited the VM_CONTROLLER"

        break
    else:
        print "Enter the correct service you need\n"

**************************************************************
**

**************************************************************
**

**************************************************************
**

**************************************************************
**
```

So this is the whole code, I hoped you would understand it all, but if you ask me in scripting languages like python even the programmer don't know what happens below, but the best part is things happen and you get result results. Until you go into every module and look in detail what your called modules are doing.

4.8 Test platform

The code for the virtual machine controller was tested on the futuregrid. We could also test it on amazon platform, as when things don't work fine on futuregrid. But one had to pay on amazon and on futuregrid its free. The simple way to test the code is just run the vm_controller.py file at the terminal and you would see a list of services. Choose anyone as you desired, and you would be directed to its menu and from there you would be able to see the output.

4.9 The Audience

The Audience of this work are obviously student community. Like for example someone among the students interested in the topic will take on this controller and add more features to it. And make it much better and much more useful.

Now as far the user community is concerned, now this controller serves the purpose of useful enough for simple uses, and basic cloud services. Now if someone don't want the pain of understanding the underline technology and concepts related to boto, Nimbus and related stuff and just want to use the EC2 instances and storage services can directly interact with the software and it's very easy in use and quite self-explanatory. Anyways there are a number of people to whom it may benefit. Especially to me, as by showing it to my professor, I will be able to graduate from Politecnico.

4.10 Future Improvements in the Controller

Now there are number of ways you can improve the controller. Now I can make a list of it and I hope one can think of as many as he wants.

i) Load balancer and AutoScaling (I have already explained them in detail in above chapters)
ii) You can add a GUI (graphical user interface) to everything
iii) You can make the S3 service much better, as right now it only do the basic storage operations
iv) You can add features like add or remove volume from the instances, control the cap of the instances, ram and other stuff.
v) Now to be more efficient and if one wants to get to rich, you can make a controller like amazon guys are giving for their products and then you can sell it and get real rich.

So in short there are a number of possibilities in which you can make this controller much better than now. And it would be fun to do it, as in the end one understands a lot of different underlying technologies and difficult concepts related to cloud computing and virtualization environment.

Chapter 5

Conclusion

So this is the last chapter in the thesis, and it focus on concluding everything. Now In the previous chapter about the virtual machine controller, I had concluded a lot, but just if something is left it will be covered in this chapter. But I will avoid repetition.

Now we started with the all the basic things about Boto, now is the point to conclude everything about my VM Controller.

This virtual machine controller does only few services, but exploring more and more will give up the opportunity to use more services available. We could auto-scale, we can have a load balancer, so there are so much other things we could, by following the pattern used in the controller.

I remember when I choose this topic, related to Boto API , I was a bit afraid as I had never been expose to computer Science a lot, but in the end it was very useful and informative.

Now with my Controller in future any student who wants to proceed with the work can add more features to it.

On the other hand it give user an easy way to starting multiple no off instance in a very simple way, according to it needs. And after finish his needs, can stop them.

Also the user have the option to automatically to get into the root of the one's own remote virtual machine, very simply.

So In the end I Conclude by saying, It was hard doing all above with a time-limit, but it's fun in the end.

Bibliography References

[1] Amazon AWS – AWS Solutions (2012). http://aws.amazon.com/solutions/aws-solutions/.

Amazon Elastic Compute Cloud (Amazon EC2) (2012). http://aws.amazon.com/ec2.
[2] Salesforce Homepage (2012). http://www.salesforce.com/

[3] FutureGrid Project www.futuregrid.org/

[4] Nimbus Platform www.nimbus.org

[5] National institute of standards and Technology (US Department of Commerce), accessed documents from their website.

[6] https://portal.futuregrid.org/sites/default/files/Boto%20Hands%20On.pdf

[7] http://boto.s3.amazonaws.com/genindex.html

[8] http://virt-manager.org/ (Virtual machine manager).

[9] http://code.google.com/p/boto/

[10] http://www.nimbusproject.org/docs/2.9/elclients.html

[11] http://wiki.xen.org/wiki/Main_Page

[12] http://www.vmware.com/