ANALYSIS OF CLOUD COMPUTING SERVICES USING AMAZON EC2

Mounika Namburu
Univ of Colorado, Colorado Springs
mnamburu@uccs.edu

ABSTRACT
The main scope of this project is to understand how to best utilize different cloud computing services to get good performance for less cost. Amazon EC2 gained popularity in providing good cloud computing services hence is used here. This paper also deals with performance analysis to uccs.edu from different regions and performance between the regions. Cloud computing is best suited for small organizations which are worried about cost incurred in maintaining the infrastructure. Security is main concern of the cloud based computing and can be the future work of this project.

1. INTRODUCTION

"So what is Cloud Computing? Is it utility computing? Is it an application service provider's offering? Is it virtual machines in the sky?" are some of the general questions that arise when we use the term "cloud computing". All of these are correct depending on who you ask. Generally, Cloud Computing can be defined as "Anything that involves delivering hosted services over the Internet." These services are broadly divided into three categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). The name cloud computing was inspired by the cloud symbol that's often used to represent the Internet in flowcharts and diagrams.

Berkley simply defines Cloud Computing as "Pay as you go SaaS" because Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. The services themselves have long been referred to as Software as a Service (SaaS), so it is used. The datacenter hardware and software is what a Cloud is called. When a Cloud is made available in a pay-as-you-go manner to the public, it is called Public Cloud; the service being sold is Utility Computing. Private Cloud refers to internal datacenters of a business or other organization that are not made available to the public. Thus, Cloud Computing is the sum of SaaS and Utility Computing, but does not normally include Private Clouds.

A cloud service has three distinct characteristics that differentiate it from traditional hosting. It is sold on demand, typically by the minute or the hour; it is elastic -- a user can have as much or as little of a service as they want at any given time; and the service is fully managed by the provider (the consumer needs nothing but a personal computer and Internet access). Significant innovations in virtualization and distributed computing, as well as improved access to high-speed Internet and a weak economy, have accelerated interest in cloud computing.

So Core features of cloud computing are scalable, centrally managed and accessible via the internet. Cloud computing boils down to running software on someone else's robust hardware in a data center somewhere else (someone else's software on someone else's hardware in someone else's data center). Another key component has become a very low barrier to entry in both technical and financial terms. The service should be very easy to begin using and it should be very cheap and/or free (at least at a basic level). The general approach to pricing for most cloud platforms, so far, has been a per usage or subscription basis. Per usage charges you only for what you actually use. If you use a virtual machine, you will pay a certain amount for every CPU hour consumed; for storage you will pay by the gigabyte or terabyte. You will usually have to pay some kind of bandwidth/networking costs.

The final major pieces to the cloud computing puzzle are nearly instant scalability and nearly infinite scalability. If your web site grows from a few thousand hits per day to millions of hits per day, you don't want your customers to get error messages until you can buy and provision new servers. Most platforms will allow you to automatically scale, or scale with just a couple of button presses.

A cloud can be private or public. A public cloud sells services to anyone on the Internet. (Currently, Amazon Web Services is the largest public cloud provider.) A private cloud is a proprietary network or a data center that supplies hosted services to a limited number of people. When a service provider uses public cloud resources to create their private cloud, the result is called a virtual private cloud. Private or public, the goal of cloud computing is to provide easy, scalable access to computing resources and IT services.

2. CLOUD PROVIDERS

SaaS: SaaS is currently the most popular type of cloud computing. Yahoo email, Google apps, zoho, and various other packages like CRM are all instances of SaaS. Application Service Providers (ASP) were the first SaaS providers. ASP was its own buzzword back in the late 1990s and early 2000s.

One of the aspects of SaaS is multi-tenancy or the ability for many customers to share the same service but maintain their own data securely. CRM is the predominant paid SaaS offering but email is, by far, the predominant free SaaS offering.

Any software that is offered over the internet, that runs remotely (where the location is unimportant and unrelated to the user), is a SaaS offering. With SaaS, a user has no need to worry about installation or upgrades. Of course, that also means the user has no control over versioning (or bug fixes).

The majority of SaaS offering are either free (think web mail) or per seat subscriptions (like online collaboration and CRM). SaaS may also be offered as a subscription for access and then charge fees for extras like additional storage.

PaaS: The next step up into the cloud from SaaS is the Platform as a Service. While SaaS offers applications for end users, PaaS offers a development platform for developers. Developers do not need to worry about the operating system, storage or hosting. Developers write the code and the PaaS provider provides a very simple method to upload that code and present it on the internet.

The PaaS provider gives provides the hardware, operating system, software upgrades, security and everything else related to the day to day hosting of an application. Most PaaS providers are limited to specific languages and IDEs. In some cases, the IDE is web based and in others, the IDE provides features for uploading code.

In most cases, developers do not have any access to the underlying operating system. Applications that run on PaaS platforms have to conform to some limitations that protect the provider from abuse (such as malicious software or run away resource usage).

Google App Engine (GAE) was one of the first PaaS offerings. GAE only supports python (Google promises additional languages in the future) and comes with an IDE. A developer writes an application and tests it locally. When ready to deploy the application to the world, the developer presses a button and it is automatically hosted on the Google infrastructure.

Other PaaS providers are Force.com (which builds on top of the SalesForce.com engine – which in itself is SaaS), Engine Yard (Ruby on Rails), Coghead (GUI drag and drop), and Aptana Cloud (PHP, Ruby and JavaScript). The Aptana Could is a bit different in that it is built on top of the Joyent IaaS but provides PaaS features through its Aptana Studio IDE.

Desktop as a Service: Falling some somewhere between software and a platform are Cloud Desktops (also called a Cloud OS). These desktops run inside a browser and are accessible from any desktop with an internet connection.

A cloud desktop offers word processing, spreadsheets, development tools, networking tools and more. While relatively immature at this time, we can expect this market to grow significantly in the coming years, especially as more and more smart phones and ultra portables hit the real world.

Microsoft LiveMesh offers free storage, machine synchronization and a cloud based desktop. iCloud and g.ho.st both offer complete, robust desktops (including browsers, applications and storage) completely in the cloud. The greatest benefit to a cloud desktop is that all of your applications and data is accessible from any computer. The downside is that with no internet access, you have access to none of your data or applications.

EyeOS is a completely open source cloud OS that you can download and install in your own data center. It offers all of the expected functionality and installs as a simple PHP application on your Apache web server.

The nice part about these platforms at the moment is that most are completely free. Sign up, login and you have a virtual desktop off in the clouds. The business model of most of these (such as iCloud or g.ho.st) are subscriptions for enhanced services and extra capabilities (such as bandwidth or disk space).

The other significant aspect of these cloud desktops is the ability to run desktop quality applications, such as word processors and spreadsheets, from a phone. Business at internet speeds and internet availability.

IaaS: Amazon AWS is the largest of all the IaaS providers. Where SaaS offers a complete application as service and PaaS offers the ability to develop an application, IaaS doesn't care about the application at all. IaaS provides the underlying hardware and operating system resources to do anything you want. IaaS has also been referred to as Everything as a Service.

IaaS offers CPU, memory, storage, networking and security as a package. IaaS is the virtual machine in the sky. In general, with IaaS, you choose from a range of operating systems (usually some flavor of open source), a size for your hardware (number of CPUs and CPU power) and an amount of storage.

There are a number of successful IaaS providers: Amazon, Joyent, GoGrid and FlexiScale. While Amazon is the best known of the providers, Joyent is also huge. Joyent hosts many Facebook applications and they host the social network LinkedIn, among others. In addition to proving that Amazon is not the only game in town, this also proves that real businesses (although without much of a business plan) are running in the cloud. It also shows that businesses with a need to scale are doing so.

The benefits of IaaS, in addition to the ability to scale, are the costs to get started and the ability to pay only for what you use. For a startup or small business, one of the most difficult things to do is keep capital expenditures under control. By moving your

infrastructure to the cloud, you have the ability to scale as if you owned your own hardware and data center (which is not realistic with a traditional hosting provider) but you keep the upfront costs to a minimum. So, you are only limited by how much you're willing to spend. But, around $0.10/hr per server, with no long-term contracts or upfront hardware investments, Amazon EC2 provides a cost-efficient solution; especially for those that believe that time is money!

The rest of the paper is divided as follows: Next section talks about basics of the services that Amazon provides and how to get an AWS account. Section 3/4 describes some terms of Amazon ec2 and their importance. Section 4/5 introduces booting of EBS instance and resizing the EBS volume. Next section shows how to create users for that ec2 instance and also some simple steps on how to bring the instance up and down at anytime of the day by the admin. Last but one section

shows performance benchmarks I ran on the machine and their results. I conclude the paper in the last section with some future works and References are provided at the end of the paper.

3. AMAZON WEB SERVICES

Amazon provides a fantastic suite of web services that enables developers to create dynamic and robust applications. Deploying on AWS can save you time, money and manpower compared to building and maintaining more traditional systems.

Amazon Web Services delivers a number of benefits for IT organizations and developers alike, including:

- Cost-effective: Pay only for what you use, as you use it, with no up-front commitments. As the Amazon Web Services cloud grows, our operations, management and hardware costs shrink, and we pass the savings onto you.
- Dependable: Utilize a battle-tested, web-scale infrastructure that handles whatever you throw at it. The Amazon Web Services cloud is distributed, secure and resilient, giving you reliability and massive scale.
- Flexible: Build any application you want using any platform or any programming model. You control the resources you consume and fit them into your application as you see fit.
- Comprehensive: Don't start from scratch. Amazon Web Services gives you a number of services you can incorporate into your applications. From databases to payments, these services help you build great applications cost effectively and with less up-front investment.

Amazon Simple Storage Service (S3): Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers. Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

Amazon S3 is intentionally built with a minimal feature set.

- Write, read, and delete objects containing from 1 byte to 5 gigabytes of data each. The number of objects you can store is unlimited.
- Each object is stored in a bucket and retrieved via a unique, developer-assigned key.
- A bucket can be stored in one of several Regions. You can choose a Region to optimize for latency, minimize costs, or address regulatory requirements. Amazon S3 is currently available in the US Standard, EU (Ireland), US West (Northern California) and Asia Pacific (Singapore) Regions. The US Standard Region automatically routes requests to facilities in Northern Virginia or the Pacific Northwest using network maps.
- Objects stored in a Region never leave the Region unless you transfer them out. For example, objects stored in the EU (Ireland) Region never leave the EU.

Authentication mechanisms are provided to ensure that data is kept secure from unauthorized access. Objects can be made private or public, and rights can be granted to specific users.

Amazon CloudWatch: Amazon CloudWatch is a web service that provides monitoring for AWS cloud resources, starting with Amazon EC2. It provides customers with visibility into resource utilization, operational performance, and overall demand patterns—including metrics such as CPU utilization, disk reads and writes, and network traffic. To use Amazon CloudWatch, simply select the Amazon EC2 instances that you'd like to monitor; within minutes, Amazon CloudWatch will begin aggregating and storing monitoring data that can be accessed using the AWS Management Console, web service APIs or Command Line Tools.

Amazon CloudFront: Amazon CloudFront is a web service for content delivery. It integrates with other Amazon Web Services to give developers and businesses an easy way to distribute content to end users with low latency, high data transfer speeds, and no commitments. Amazon CloudFront delivers your static and streaming content using a global network of edge locations. Requests for your objects are automatically routed to the nearest 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 contracts or monthly commitments for using Amazon CloudFront – you pay only for as much or as little content as you actually deliver through the service.

Creating AWS Account: To access any web service AWS offers, you must first create an AWS account at http://aws.amazon.com. An AWS account is simply an Amazon.com account that is enabled to use AWS products; you can use an existing Amazon.com account login and password when creating the AWS account. Note that creating an AWS account is free of charge. They ask for your payment details only when you try to use any service. AWS provides some access credentials for easy usage of all their services. Go to account and then security credentials located on the top right of the AWS homepage. Save your access key and create and download a X.509 certificate for future use.

4. AMAZON ELASTIC COMPUTE CLOUD (EC2)

EC2 introduces a new paradigm for web hosting. By allowing developers to scale their number of machines up or down within minutes, it offers the capability to create distributed and scalable applications that run in the cloud. EC2 is flexible, reliable, secure, and most importantly cheap! By only paying for the resources that you actually use, you can bring your multi-server application to market much cheaper than ever before, and maintain an extremely high level of quality and availability.

EC2 is the computing part of the Amazon services. EC2 provides the CPU, memory, operating system and transient storage. EC2 is the equivalent of a barebones PC. You get to pick the amount of RAM you need (from a predefined list of configurations), the amount of transient storage you need (also from a list) and the number of CPUs you need (from a series of compute options). For the operating system, you can choose from various flavors of Linux, Solaris or Microsoft Windows Server. The basis if EC2 is the Amazon Machine Image (AMI). An AMI is a virtual machine with your chosen operating system and applications bundled together. You can create your own AMIs from scratch if you want to. To get started though, Amazon offers hundreds of public AMIs with many operating systems and pre-installed applications.

Signing Up: After you create an AWS account, login with that username and password, and choose Elastic Compute Cloud from the list of services available. Then select sign up for EC2 from the EC2 homepage. Amazon providers ask for your credit card details (but is not charged till you use their resources) and then you can continue with the process. After you successfully gained access to EC2 service, go to AWS Management Console, which is a web-based, point-and-click, graphical user interface that makes it even easier to access and manage AWS Infrastructure Web Services.

Information about cost per hour for an instance, bandwidth, data transfer rate and others can be known from https://aws.amazon.com//ec2/ i.e., ec2 homepage.

US – N. Virginia | US – N. California | EU – Ireland | APAC – Singapore

Standard On-Demand Instances	Linux/UNIX Usage	Windows Usage
Small (Default)	$0.085 per hour	$0.12 per hour
Large	$0.34 per hour	$0.48 per hour
Extra Large	$0.68 per hour	$0.96 per hour
High-Memory On-Demand Instances		
Extra Large	$0.50 per hour	$0.62 per hour
Double Extra Large	$1.20 per hour	$1.44 per hour
Quadruple Extra Large	$2.40 per hour	$2.88 per hour
High-CPU On-Demand Instances		
Medium	$0.17 per hour	$0.29 per hour
Extra Large	$0.68 per hour	$1.16 per hour

Pricing is per instance-hour consumed for each instance type, from the time an instance is launched until it is terminated. Each partial instance-hour consumed will be billed as a full hour.

Figure 1. On-Demand Instances Prices Table

AMI: An Amazon Machine Image (AMI) is an encrypted machine image that contains all information necessary to boot instances of your software. For example, an AMI might contain all the software to act as a web server (e.g., Linux, Apache, and your web site) or it might contain all the software to act as a Hadoop node (e.g., Linux, Hadoop, and a custom application).

You launch one or more instances of an AMI. An instance might be one web server within a web server cluster or one Hadoop node.

Regions and Availability Zones: Amazon EC2 provides the ability to place instances in multiple locations. Amazon EC2 locations are composed of Availability Zones and Regions. Regions are dispersed and located in separate geographic areas (e.g., US and EU). Availability Zones are distinct locations within a Region that are engineered to be isolated from failures in other Availability Zones and provide inexpensive, low latency network connectivity to other Availability Zones in the same Region.

By launching instance in separate Regions, you can design your application to be closer to specific customers or to meet legal or other requirements. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location.

The following figure shows Amazon EC2. Each Region is completely independent. Each Availability Zone is isolated, but connected through low-latency links.

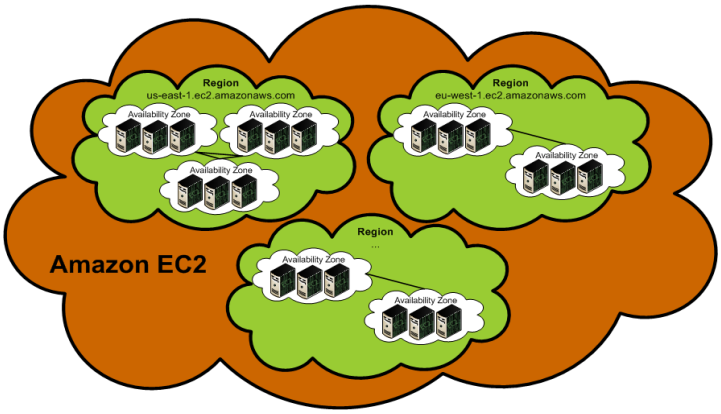

Figure 2. Regions and Availability Zones

A. Regions

Amazon EC2 provides multiple Regions so you can launch Amazon EC2 instances in locations that meet your requirements. For example, you might want to launch instances in Europe to be closer to your European customers or to meet legal requirements.

Each Amazon EC2 Region is designed to be completely isolated from the other Amazon EC2 Regions. This achieves the greatest possible failure independence and stability, and it makes the locality of each EC2 resource unambiguous.

B. Availability Zones

Amazon operates state-of-the-art, highly available data center facilities. However, failures can occur that affect the availability of instances that are in the same location. Although this is rare, if you host all your Amazon EC2 instances in a single location that is affected by such a failure, your instances will be unavailable.

For example, if you have instances distributed across three Availability Zones and one of them fails, you can design your application so the instances in the remaining Availability Zones handle any requests.

Elastic IP Addresses: By default, all Amazon EC2 instances are assigned two IP addresses at launch: a private (RFC 1918) address and a public address that is mapped to the private IP address through Network Address Translation (NAT).

If you use dynamic DNS to map an existing DNS name to a new instance's public IP address, it might take up to 24 hours for the IP address to propagate through the Internet. As a result, new instances might not receive traffic while terminated instances continue to receive requests.

To solve this problem, Amazon EC2 provides elastic IP addresses. Elastic IP addresses are static IP addresses designed for dynamic cloud computing. Elastic IP addresses are associated with your account, not specific instances. Any elastic IP addresses that you associate with your account remain associated with your account until you explicitly release them. Unlike traditional static IP addresses, however, elastic IP addresses allow you to mask instance or Availability Zone failures by rapidly remapping your public IP addresses to any instance in your account.

You can only associate one elastic IP address with one instance at a time. When you associate an elastic IP address with an instance, its current public IP address is released to the Amazon EC2 public IP address pool. If you disassociate an elastic IP address from the instance, the instance is automatically assigned a new public IP address within a few minutes.

5. ELASTIC BLOCK STORE

Amazon Elastic Block Store (Amazon EBS) is a type of storage designed specifically for Amazon EC2 instances. Amazon EBS allows you to create volumes that can be mounted as devices by Amazon EC2 instances. Amazon EBS volumes behave like raw unformatted external block devices. They have user supplied device names and provide a block device interface. You can load a file system on top of Amazon EBS volumes, or use them just as you would use a block device.

You can create up to twenty Amazon EBS volumes of any size (from one GiB up to one TiB). Each Amazon EBS volume can be attached to any Amazon EC2 instance in the same Availability Zone or can be left unattached.

Amazon EBS provides the ability to create snapshots (backups) of your Amazon EBS volumes to Amazon S3. You can use these snapshots as the starting point for new Amazon EBS volumes and can protect your data for long term durability. Additionally, you can share snapshots with specific users or make a snapshot public.

Amazon EBS volumes provide the following:

- Off-instance storage
- Persistence beyond the lifetime of instances
- High availability and reliability
- Ability to attach to and detach from a running instance
- Exposure as a device within an instance

Amazon EBS snapshots provide the following:

- Ability to capture the current state of a volume
- Data backup
- A method for instantiating new volumes that contain the exact contents of a snapshot

The following figure shows the cost per volumes charges by Amazon:

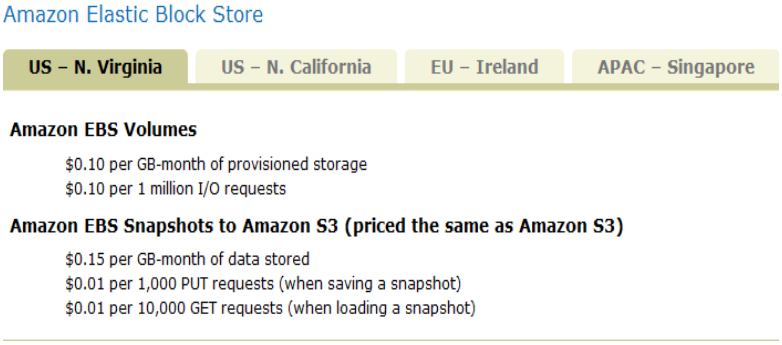

Figure 3. EBS Prices Table

Booting Instance from EBS (Management Console): After you login to the AWS Management Console, go to the EC2 Dashboard and click AMIs in the navigation on the left. A list of AMIs is displayed. In the Viewing menu, select EBS Images to filter the list.

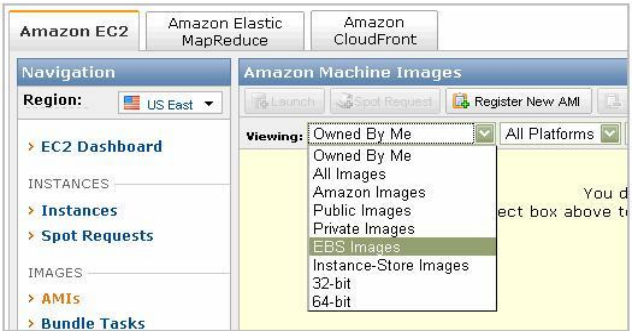

Figure 4. Selecting EBS Images

A list of Amazon EBS-backed AMIs is displayed. Click the label on the Owner column to sort the AMIs by owner. The AMIs created by Amazon (Owner=amazon) appear at the top of the list. Locate and click the AMI you want to launch and look at its details. For example, the following figure shows the details of amazon/getting-started-with-ebs-boot AMI.

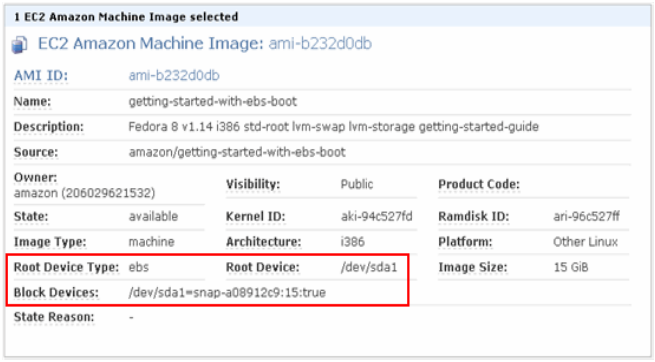

Figure 5. Details of a selected AMI

Notice that the Root Device Type is ebs and the Root Device is /dev/sda1. Notice also that the Block Devices field shows information about the root device's mapping: /dev/sda1=snap-a08912c9:15:true. This means the root device /dev/sda1 is mapped to a 15 GiB volume created from the snap-a08912c9 snapshot, and the volume's DeleteOnTermination flag is true.

Security Groups: A security group is a named collection of access rules. These access rules specify which ingress (i.e., incoming) network traffic should be delivered to your instance. All other ingress traffic will be discarded.

You can modify rules for a group at any time. The new rules are automatically enforced for all running instances and instances launched in the future.

We will also need a key pair to root login via SSH or PuttY. So create a security group and a key pair from the navigation menu. Select the AMI and click Launch Instance. The launch wizard starts. Walk through the wizard and launch an instance. If you'd like, connect to the instance once it's running. Go to the Instances page and look at the instance's information.

Right-click on the running instance and click Connect. This will show you some steps on how to connect to your instance using SSH or PuttY. You need PuttYgen to convert your private file to .ppk format if you are using PuttY to connect to your instance.

Command line tools: The command line tools are available as a ZIP file in the Amazon EC2 Resource Center. These tools are written in Java and include shell scripts for Windows 2000/XP and Linux/UNIX/Mac OSX. The ZIP file is self-contained; no installation is required. You just download it and unzip it. Some additional setup is required in order for the tools to use your AWS account credentials.

The command line tools depend on an environment variable (EC2_HOME) to locate supporting libraries. You'll need to set this environment variable before you can use the tools. This should be set to the path of the directory into which the command line tools were unzipped. This directory is named ec2-api-tools-A.B-nnnn (A, B and n are version/release numbers), and contains sub-directories named bin and lib.

On Linux and UNIX, you can set this environment variable as follows.

$ **export EC2_HOME=<path-to-tools>**

On Windows the syntax is slightly different.

C:\> **set EC2_HOME=<path-to-tools>**

On Linux and UNIX, you can set private key and certificate environment variables as follows.

$ **export EC2_PRIVATE_KEY=~/.ec2/pk-privatekey.pem**
$ **export EC2_CERT=~/.ec2/x509-cert.pem**

On Windows the syntax is slightly different.

C:\> **set EC2_PRIVATE_KEY=c:\ec2\ pk-privatekey.pem**
C:\> **set EC2_CERT=c:\ec2\x509-cert.pem**

To find a suitable AMI, use the **ec2-describe-images** command. To run the instance, use **ec2-run-instances** command**.** To start and stop the instances, use **ec2-start-instances** and **ec2-stop-instances** commands.

Linux Instance: After you connect to the instance with public DNS of the instance, login as root using your private key via SSH or PuttY, you can install httpd using '**yum install httpd'** command on a linux machine. You can test Apache in the web browser after starting httpd with 'etc/init/d/httpd start' command.

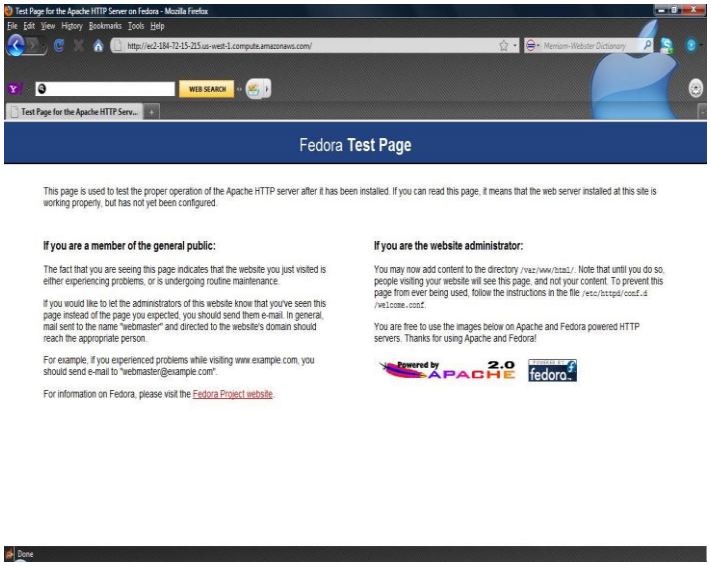

Figure 6. Apache Test Page

Windows Instance: To Remote Desktop to a Windows instance, you need the administrator password, by right clicking on the instance and giving your private key to decrypt it.

Now, you can use Remote Desktop Connection with public DNS of your instance, login as administrator and decrypted password, to connect to the instance.

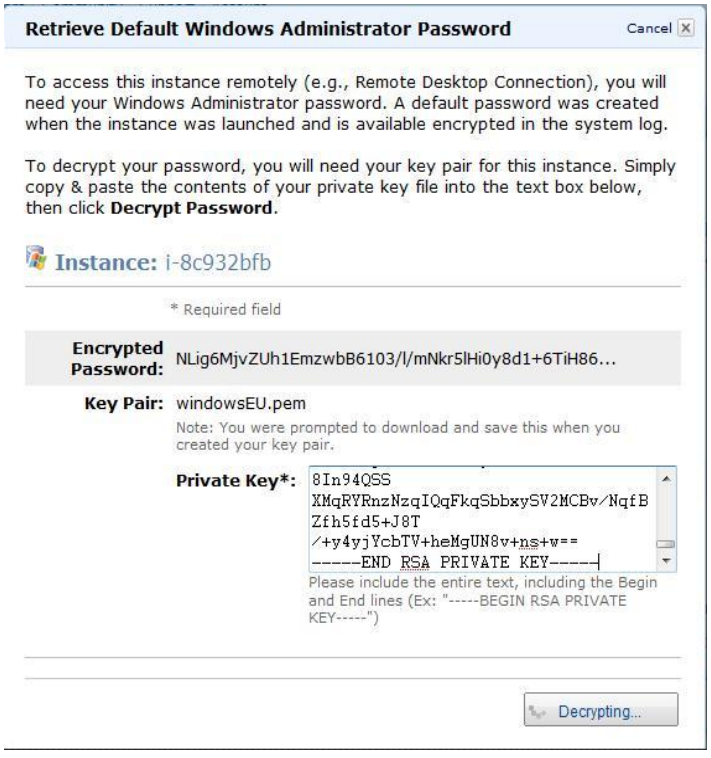

Figure 7. Retrieving Windows Administrator Password

6. AUTO START and STOP

You may need to automatically bring up and bring down the instances at anytime of the day for easy access to your users. Such automatic operations also let you secure your instance's private key without any need to share it with all your users.

Two commands which allow us to do this on Linux machine are:

- **at**- Execute a task at a specific time. For example, shutdown computer, send birthday remainder etc. Useful to schedule one job at a time or a single future event.

- **cron** – If you want to shutdown Linux box automatically everyday 8 pm then you need to use cron instead of at command. Useful to schedule recurring events or daily events such as backup data, or check system security etc.

If you wish to shutdown a Linux machine automatically at 8 pm, type the command 'at 8pm' and press [enter] key, then type halt followed by enter key. To save your job, press CTRL+D.

at 8pm

at> halt

(Press CTRL+D)

Some more examples are: If you wish to run a job at 6am on Monday, use 'at 6am Monday' command.

Run job in 2 minutes time: 'at now + 2 minutes' shown in figure 8.

Run job at 4pm but 3 days later: 'at 4pm + 3 days'

Run job at 10am on 31st July: 'at 10am Jul 31'

Before using 'at' make sure you have 'atd' service running, if not start it using '/etc/init.d/atd start' command.

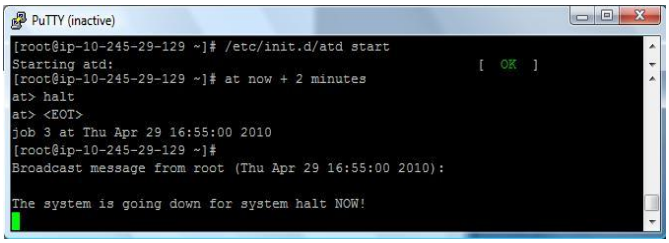

Figure 8. Shutting down the instance

To schedule/add a new cronjob, use '**crontab –e**' command. Append the following entry to it to shutdown box at 20:00 hrs:

0 20 * * * /sbin/shutdown -h now

Save and close the file.

On Windows machine, 'at' command can be used by the command:

c:> at 2:00am c:\admutils\psshutdown.exe -r -f -c -t 10

Where,

- -s: Shutdown windows server
- -r: Reboot windows server
- -f: Forces all running application to exit
- -c: Allow the shutdown to by cancel by user
- -t: Specifies the countdown in seconds until the shutdown

7. CREATING USERS

We may need to create additional users for these instances and secure these accounts with passwords. Very often, these passwords are not very secure and are trivially cracked by intruders. They can also be taken with the employee when he or she leaves your organization.

A far better and certainly more secure mechanism is to use ssh keys for all users, in a similar way to the secure root login method described in the guide. Because the bundled commands only work for the root user, we need to perform some additional steps for adding ssh keys for all additional user accounts.

For generating ssh keys for regular user accounts, we can allow each user in the organization generate their own key. Have them provide their **public key** to your system administrator who may then add the key of any authorized use to the account(s) for which they need access on your ec2 instances. Alternatively, your administrator can generate a key pair (public and private) for each user and send the private keys to them respectively.

Linux/Unix users can give the command "ssh-keygen -b 1024 -f user -t dsa" by replacing user with the login name of each user in their local machine, to generate a key pair. So this will create 2 files:

- user (private key)
- user.pub (public key)

Copy all the public key files that you generated to a temporary place on your instance:
scp -i root *.pub ec2-your-instance-name.compute.amazonaws.com:/tmp

Windows users can use PuttYgen to generate a public/private key pair for each user and with parameters SSH-2 RSA. You can copy all the public files to your instance using the command:

C:\> pscp -i id_rsa-gsg-keypair.ppk pk-privatekey.pem x509-cert.pem ec2-your-instance-name.compute.amazonaws.com:/tmp

User Account Creation

Log in to the instance as root. For each user you are creating, add the user to your instance with the
useradd -c "firstname lastname" user
For simplicity's sake, use the same "user" name as you did for key generation. Now we need to place the key into their ssh authorized keys file (again, replacing "user" with the username you chose earlier)
cd ~user
mkdir .ssh
chmod 700 .ssh
chown user:user .ssh
cat /tmp/user.pub >> .ssh/authorized_keys
chmod 600 .ssh/authorized_keys
chown user:user .ssh/authorized_keys
Ensure that your users all have their appropriate private keys and that they are in the users' ".ssh" directory of the local machine. Each user can now log in to any instance that has them added as a user and a copy of their public key in the .ssh/authorized_keys file in their home directory on the EC2 instance.

8. PERFORMANCE

i. Web Performance

For measuring web performance among the instances of different regions (US East, US West, and Europe) and from different instances to www.uccs.edu, I used Apache HTTP server benchmarking tool **ab**. It is designed to give you an impression of how your current Apache installation performs. This especially shows you how many requests per second your Apache installation is capable of serving.
The following command is run on all 3 machines in 3 different regions to uccs.edu and to eachother:
ab –c 10 –n 2000 http://www.uccs.edu/ > result.txt
Result.txt file is as follows:

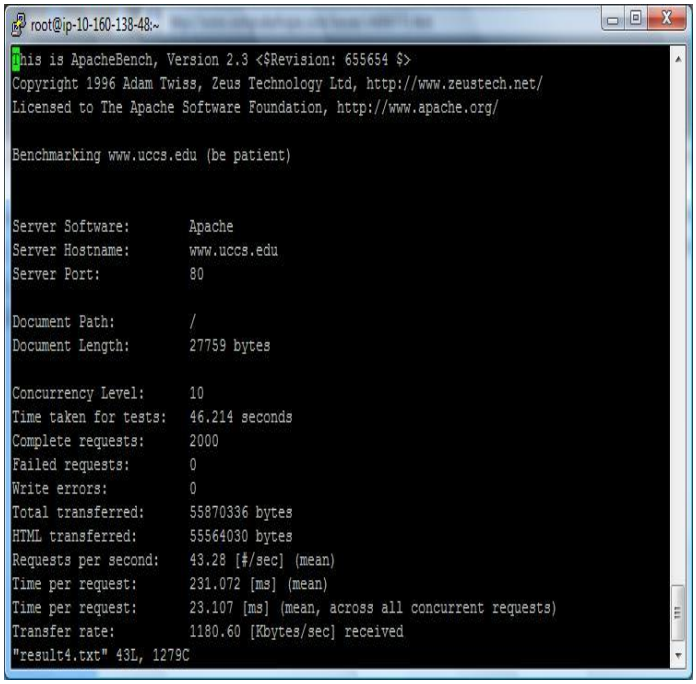

Figure 9. ab command results

I ran the same command 10 times on each machine and took the best results.

The following figure shows the resulted Requests per second from all 3 regions to UCCS.

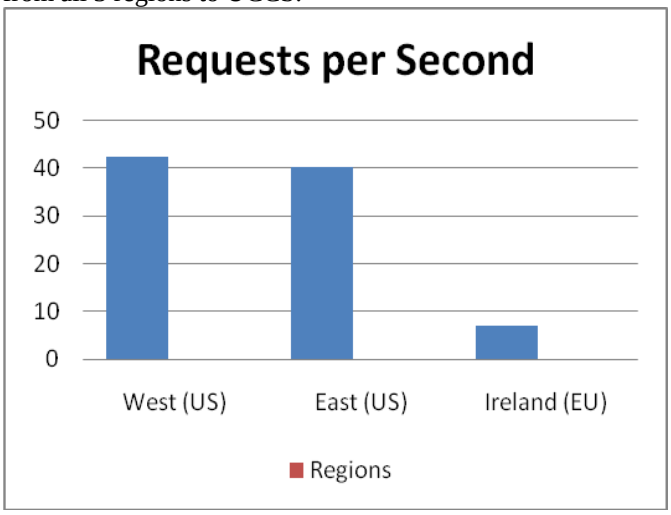

Figure 10. Web Performance (Requests per second)

Requests per second from the result between US East and US West is around 47.20, while it's around 5.99 from Ireland (EU) to West (US) and 10.70 from Ireland (EU) to East (US).

ii. Bandwidth Performance

When I tried to run popular bandwidth tools like Dhrystone and bprobe on the Virtual machines, it was unsuccessful as we may need some special permissions for such commands to execute on Virtual machines. Hence I continued using the round trip time from the ping command run from instances or virtual machines to uccs.edu. Packet sizes I considered were 32bytes and 64bytes.

By default, in linux machines, packet size is 64bytes. So, -s option can be used to change it:

ping –s 32 uccs.edu

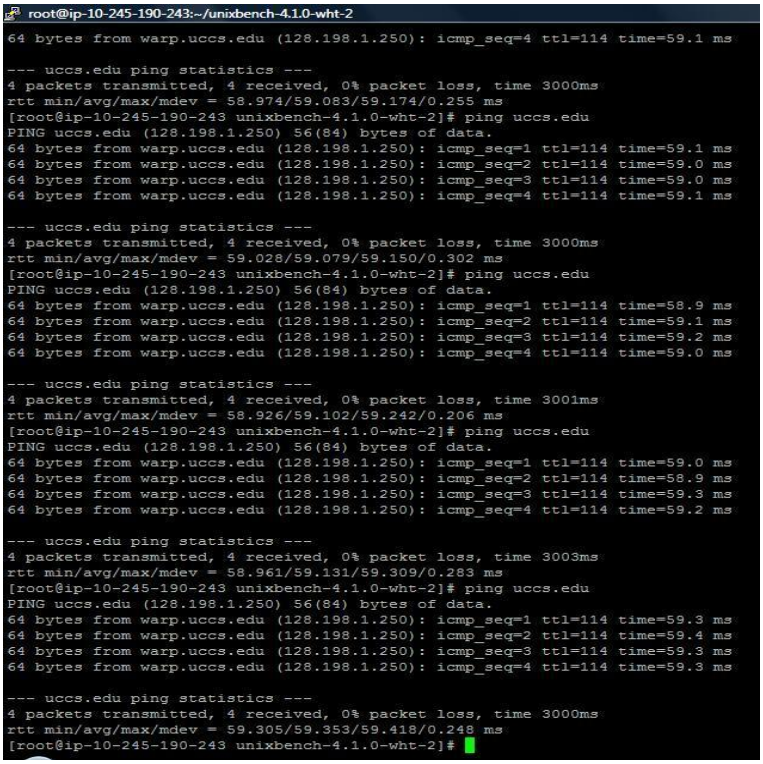

Figure 11. ping command in linux

By default, in windows server 2003, packet size is 32bytes. So, -l option can be used to change it:

ping –l 64 uccs.edu

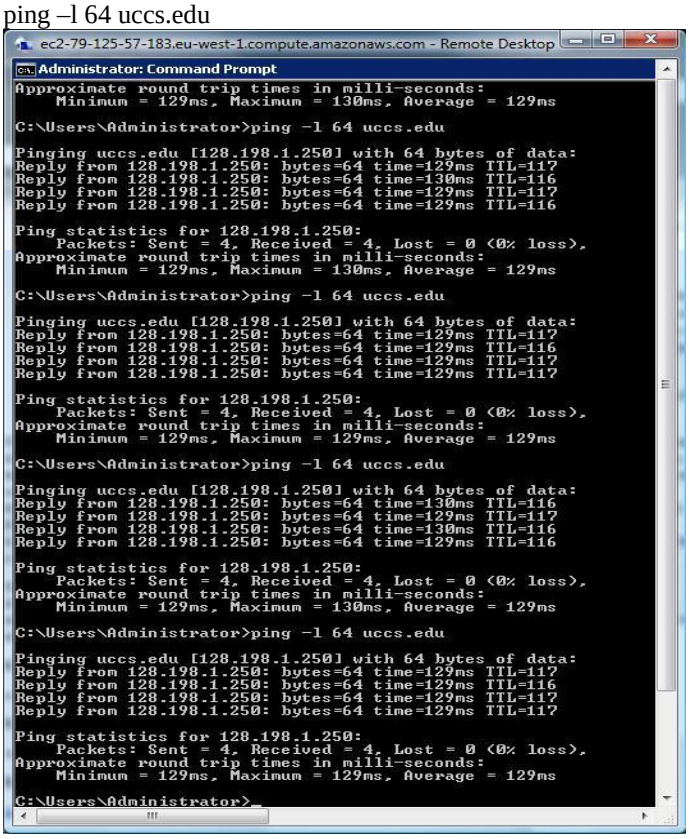

Figure 12. ping command in windows

Best results among 10 runs resulted in a bandwidth range of:

West to UCCS: 640MBits/s to 17648MBits/s

East to UCCS: 256MBits/s to 13120MBits/s

EU to UCCS: 40MBits/s to 85MBits/s

The above resulted bandwidth numbers are just satisfactory. These numbers can be improved efficiently using different services provided by Amazon, but prices can rack up along with the bandwidth.

9. CONCLUSION AND FUTURE WORK

This paper evaluates different services provided by the leading cloud provider, AMAZON. It shows how to create/own/access elastic compute cloud instance in all the possible ways. It deals with issues like bringing the machine up and down at anytime and creating users in a secure way with authenticated keys, which are very useful for an organization like universities, non-IT based companies that are willing to join Amazon Cloud. Finally, it provides some performance benchmark results to better understand the use of having instances on a cloud. One of the future works of this paper is to provide more information on better usage of the available resources on the cloud. Other works can be to incorporate all the services in different combinations (like EC2 with CloudFront, S3, etc.) and then benchmark the best performance, Create machine images from scratch, Security of data on the cloud, etc.

ACKNOWLEDGEMENT

I want to sincerely thank Prof. Edward Chow and Prof. Richard Wiener, for funding me to access all Amazon Resources and complete this project successfully.

REFERENCES

[1] Amazon Web Services (AWS). http://aws.amazon.com/
[2] Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/
[3] http://clouddb.info/
[4] http://docs.amazonwebservices.com/AWSEC2/2009-10-31/GettingStartedGuide/
[5] http://docs.amazonwebservices.com/AWSEC2/2009-10-31/UserGuide/
[6] http://developer.amazonwebservices.com/connect/entry.jspa?externalID=351&categoryID=88
[7] http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1233&categoryID=174
[8] Amazon Elastic Block Store. http://aws.amazon.com/ebs/
[9] http://docs.amazonwebservices.com/AWSEC2/2007-08-29/GettingStartedGuide/putty.html