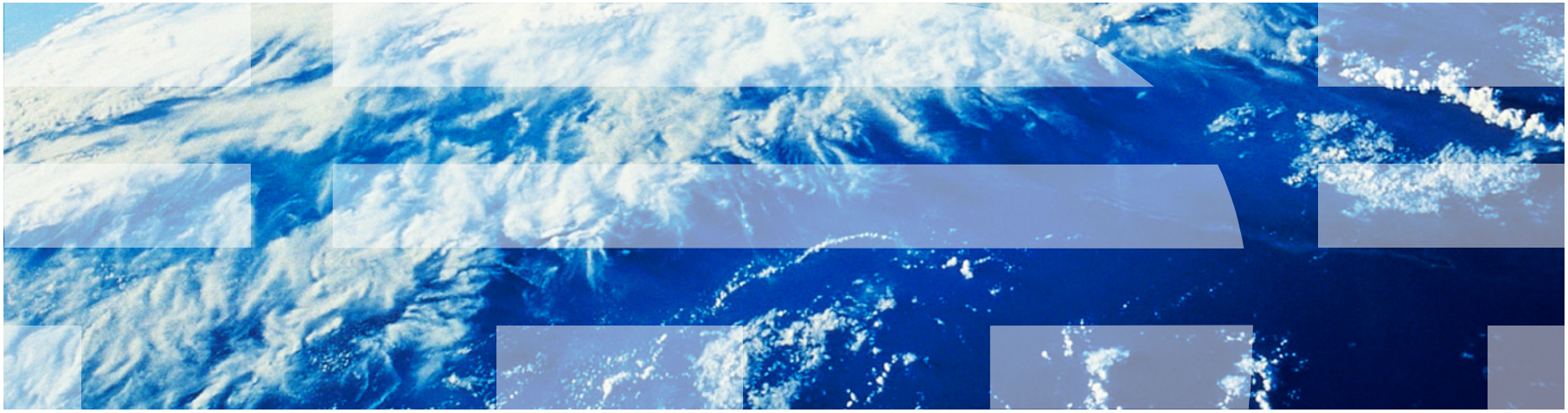
Lecture 4: Understanding On-demand Infrastructure

Sambit Sahu, IBM Research



Lecture 2: IaaS Cloud and Amazon EC2

- We learned how to request a resource using AWS programming APIs
  - Amazon EC2 SDK for java on Eclipse
    - http://aws.amazon.com/eclipse/
    - A simple tutorial http://media.amazonwebservices.com/videos/eclipse-java-sdk-video.html
- Deconstructing provisioning (create a machine) in a IaaS Cloud

This Week: Understanding and leveraging On-demand Infrastructure

- How to preserve state using Amazon EBS
  - Persistence storage for Data (EBS for now)
  - Persisting software/config changes by creating own AMI
- Elastic Infrastructure

Project Ideas

- Face recognition service on Mobile Cloud: work with Facebook image recognition and deep learning methods to develop a mobile cloud service for face recognition.
- ViewMoments: develop a mobile cloud service to find out whether mobile user is paying attention to the phone and the context
- StreetParking: develop a cloud service for finding street parking
- BikeRental: ask for details
- RideShare: automating commute sharing
- SocialRating: Creating video/TV rating from social data
- Facebook Radio: ask for details
- Digital Assistant: create a digital persona based on your app usage on mobile phone to be your digital assistant (ask for details)

Working with On-demand Dynamic Resources

- Example Scenario
  - You are the IT administrator for a large Enterprise. You have transformed your company's IT into a virtual infrastructure that you provision, manage and even provide the backup support. You are also making sure you minimize the cost of this virtual infrastructure that you are providing for your business.
  - You release resources when someone is not using it, but making sure that resources are available (mostly) when needed based on some policy, schedule or active monitoring.
  - Note that next time a user logs in, he/she is expecting to see the machine with data and software what was there before she logged off last time.
- How do you address this scenario?
  - You need to design an on-demand provision and release mechanism as the base service.
  - You shall need ability to snapshot a VM and restore it later on from this snapshot.
  - You need to persist data on a persistent storage.
  - You shall need the ability to detect when machines are idle.
  - You may need to provide some form of static IP to make sure login remains the same.

Storage Cloud from IaaS

- Storage provided as a service
- Storage Cloud examples
  - Amazon EBS (Elastic Block Storage)
  - Object based storage
    - Amazon S3
    - Google Storage Cloud
- Usage scenarios
  - Snapshot VMs and stop VM and restart later on
  - Customize AMIs
  - High availability

Attaching an EBS volume to EC2 instance

- Attaching a EBS volume to a VM

1. Create a EBS volume
2. Attach to EC2 instance
3. Create a file system

   if /dev/sdh is the name of the EBS volume , it will be present without a valid partition table , so format the EBS volume: mkfs -t ext3 /dev/sdh

4. Create a directory /ebs and mount the EBS volume

mount /dev/sdh /ebs

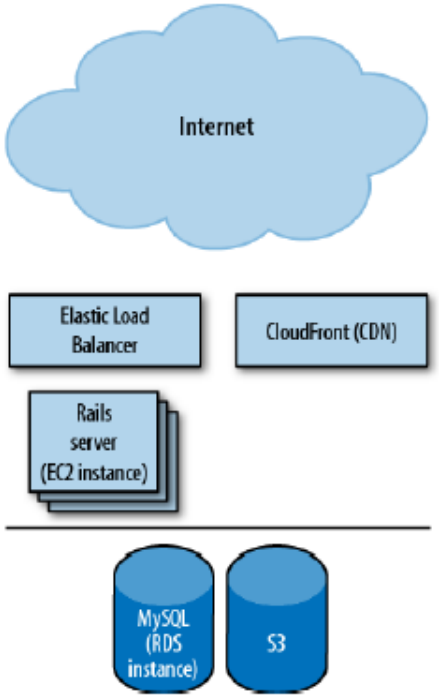
5. Check fstab
   - vi /etc/fstab

Supporting Elasticity

- Elasticity basics
- How Elasticity is supported in Amazon
- Project ideas

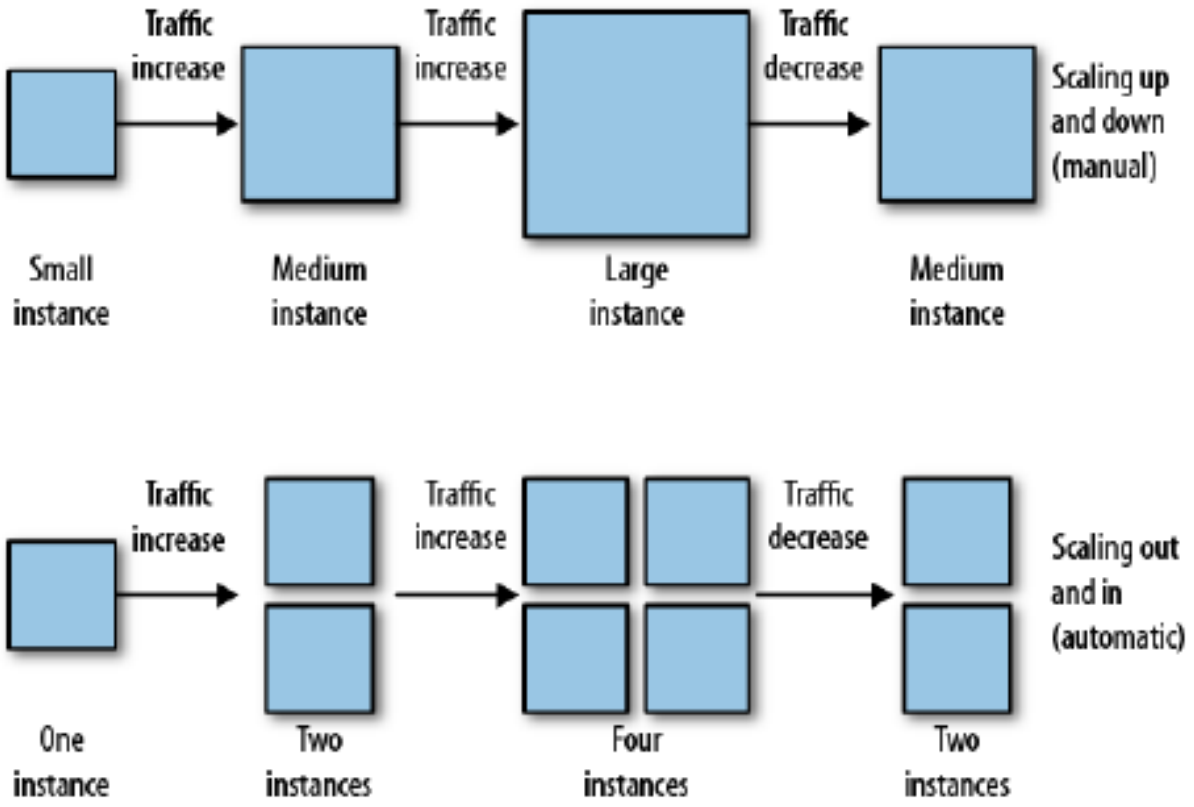
Elasticity Basics

- We have illustrated on-demand resource provisioning in a IaaS Cloud
- Elasticity is the other key attribute – allows users to dynamically request additional resources when needed
- What do we need for elasticity?
  - Detect when current resources are not able to meet the demand
  - Request right amount of resources in time
  - Add these resources to existing deployment
  - Application should be able to use the additional capacity
  - State vs stateless issues



A typical three tier web application

Usage Scenario: Elasticity



Creating our own Elastic Infrastructure (horizontal scaling)

- Monitoring: to detect change for resource capacity
  - Define ALERTs so that you know when you are running out of capacity
- Request additional resources
- Ability to add these resources dynamically
  - Need some form of load balancing
- Modify configurations so that new resources are now consumed
- Design a simple controller code to enable most of these steps

Basic Elasticity Code

- **Step 0: Initial base resource with application deployment**
  - Get on-demand resource for the base requirement
  - Deploy application on the cloud resources
  - Set up resource usage monitoring
- **Step 1: Detect if resource changes needed**
  - Fetch resource usage statistics from monitoring service

```
While (true) {
    if (resource usage too high)
        {
          need additional resources;
          goto Step 2
         }
    else if (resource usage too low)
         {
           need to release resources;
           goto Step 3
          }
}
```

- **Step 2: Get additional resources**
  - Get on-demand additional capacity
  - Make configuration changes so that application uses new resources
  - Set up resource usage monitoring
  - GOTO Step 1
- **Step 3: Remove resources**
  - Release excess capacity
  - Make configuration changes so that application does not use released resources
  - GOTO Step 1

Generic Solutions for Building Elastic Infrastructure

- Several solutions for providing
  - Example: RightScale, Amazon AWS Autoscale and Elastic Load balancer
- Amazon provides Elastic Load Balancer and Autoscale for building generic elastic infrastructure
  - Elastic Load Balancer (ELB)
    - 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. Customers can enable Elastic Load Balancing within a single Availability Zone or across multiple zones for even more consistent application performance.
    - http://aws.amazon.com/elasticloadbalancing/
  - AutoScale
    - Auto Scaling allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define. With Auto Scaling, you can ensure that the number of Amazon EC2 instances you're using increases seamlessly during demand spikes to maintain performance, and decreases 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.
    - http://aws.amazon.com/autoscaling/
- Let us use Amazon AWS autoscale and ELB to build Elastic Infrastructure

AWS Elastic Load Balancer (ELB)

Features of Elastic Load Balancing

- Using Elastic Load Balancing, you can distribute incoming traffic across your Amazon EC2 instances in a single Availability Zone or multiple Availability Zones. Elastic Load Balancing automatically scales its request handling capacity in response to incoming application traffic.
- When used in a Virtual Private Cloud (VPC), you can create and manage security groups associated with your Elastic Load Balancing to provide additional networking and security options.
- Elastic Load Balancing can detect the health of Amazon EC2 instances. When it detects unhealthy load-balanced Amazon EC2 instances, it no longer routes traffic to those Amazon EC2 instances and spreads the load across the remaining healthy Amazon EC2 instances.
- Elastic Load Balancing supports the ability to stick user sessions to specific EC2 instances.
- Elastic Load Balancing supports SSL termination at the Load Balancer, including offloading SSL decryption from application instances, centralized management of SSL certificates, and encryption to backend instances with optional public key authentication.
- Flexible cipher support allows you to control the ciphers and protocols that are accepted by Elastic Load Balancing in the SSL negotiation for client connections.
- Elastic Load Balancing supports use of both the Internet Protocol version 4 and 6 (IPv4 and IPv6).
- Elastic Load Balancing metrics such as request count and request latency are reported by Amazon CloudWatch.

- http://docs.amazonwebservices.com/ElasticLoadBalancing/latest/DeveloperGuide/Welcome.html

ELB set up using Command line tool

```
$ elb-create-lb production \
        --availability-zones us-east-1b \
        --listener "protocol=HTTP, lb-port=80, instance-port=80" \
        --listener "protocol=TCP, lb-port=443, instance-port=443"

$ elb-configure-healthcheck production \
        --target "HTTP:80/" \
        --interval 30 \
        --timeout 2 \
        --healthy-threshold 6 \
        --unhealthy-threshold 2

$ elb-register-instances-with-lb production \
        --instances i-29184c43
```

Reference: Programming Amazon EC2 – Jurg van Vliet

AutoScale

- Command line tool for autoscale
  - http://aws.amazon.com/developertools/2535

- Four key configuration concepts
  - Autoscale groups – holds instances
  - Launch configurations – that determines which instance is launched
  - Alarms – that determines when instance is launched
  - Policies – specify that instances will be launched or terminated

Autoscale

- Autoscale config

```
$ as-create-launch-config app-server-launch-config-1 \
    --image-id ami-6e1deb07 \
    --instance-type c1.medium \
    --group web
```

- Autoscaling group

```
$ as-create-auto-scaling-group app-server-as-group-1 \
    --launch-configuration app-server-launch-config-1 \
  --availability-zones us-east-1c,us-east-1d \
  --min-size 2 \
  --max-size 6 \
  --default-cooldown 120 \
  --load-balancers production
```

Reference: Programming Amazon EC2 – Jurg van Vliet

Autoscale

- Policies

```
$ as-put-scaling-policy app-server-scale-UP-on-CPU \
    --auto-scaling-group app-server-as-group-1 \
    --type ChangeInCapacity \
    --adjustment 2 \
    --cooldown 300

$ mon-put-metric-alarm alarm-app-server-scale-UP \
    --alarm-actions arn:aws:autoscaling:us-east-1:205414005158:scalingPolicy:...
    --metric-name CPUUtilization \
    --unit Percent \
    --namespace AWS/EC2 \
    --statistic Average \

   --dimensions="AutoScalingGroupName=app-server-as-group-1" \
   --period 60 \
   --evaluation-periods 2 \
   --threshold 60 \
   --comparison-operator GreaterThanThreshold

$ as-put-scaling-policy app-server-scale-DOWN-on-CPU \
    --auto-scaling-group app-server-as-group-1 \
    --type ChangeInCapacity \
    --adjustment=-2

$ mon-put-metric-alarm alarm-app-server-scale-UP \
    --alarm-actions arn:aws:autoscaling:us-east-1:205414005158:scalingPolicy:...
    --metric-name CPUUtilization \
    --unit Percent \
    --namespace AWS/EC2 \
    --statistic Average \
    --dimensions="AutoScalingGroupName=app-server-as-group-1" \
    --period 60 \
    --evaluation-periods 5 \
    --threshold 20 \
    --comparison-operator LessThanThreshold
```


Reference: Programming Amazon EC2 – Jurg van Vliet

Cloud Front, S3, HPC on AWS

- CloudFront/S3: http://www.labnol.org/internet/setup-content-delivery-network-with-amazon-s3-cloudfront/5446/
- http://www.longtailvideo.com/support/jw-player/jw-player-for-flash-v5/49/using-cloudfront
- S3fs-c http://aws.amazon.com/customerapps/3814460384379685
- iRadeo: http://aws.amazon.com/customerapps/2786429342960939
- HPC on AWS http://www.slideshare.net/AmazonWebServices/hpc-on-aws-6399706

Virtualization allows on-demand creation of (virtual) compute machines

- So how a IaaS cloud provider is able to provide machine or computing resources on demand?
- A key enabler is Server resource virtualization
  - Virtualization layer virtualizes physical resources
    - CPU
    - Memory
    - Network
    - I/O devices
  - Allows multiple operating systems to run on a single hardware platform
  - Dynamically partitioning and sharing of CPU, storage, memory, I/O devices across these OS instances
- Virtualization layer in Xen
  - Encapsulates physical resources
  - Hypervisor (dom0 in Xen) manages physical resources
- Hypervisor can instantiate a new virtual machine (guestOS) through the supported interfaces
  - createVM
  - destroyVM
  - rebootVM
  - suspendVM
- Virtualization technologies
  - VMWare ESX, ESXi
  - Xen
  - KVM
- http://www.vmware.com/virtualization/what-is-virtualization.html

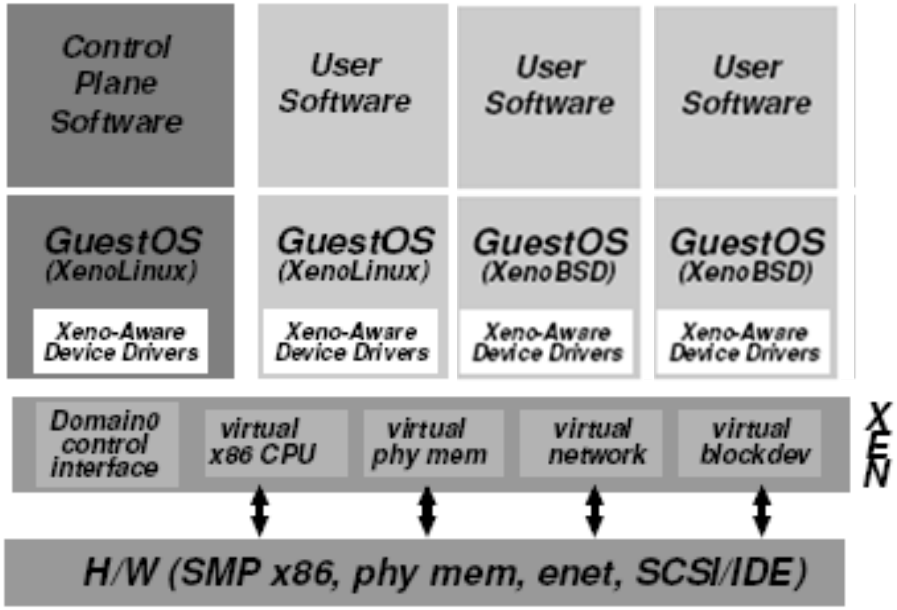


Illustration of Xen based virtualization

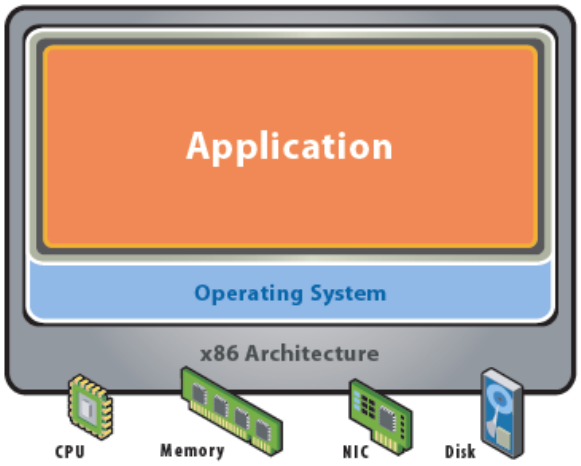
Source: Xen and Art of Virtualization

Virtualization

- Actually an old concept since 1960's invented by IBM, available on mainframes
- Virtualization is a technology
- Cloud computing is a service that is enabled by virtualization technology

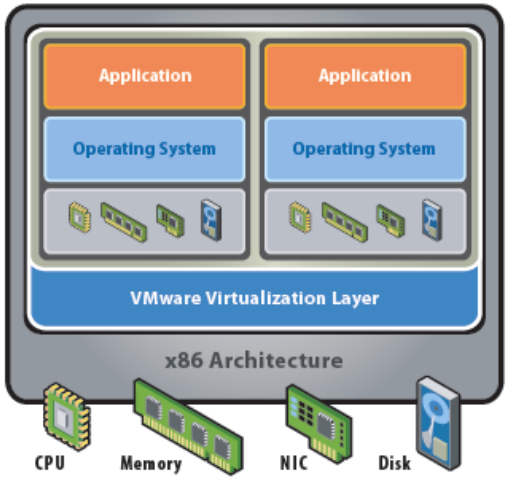
What is virtualization?

- Creation of virtual version of something such as a server, network, storage, etc.
- Separation of a resource or request for a service from the underlying physical delivery of that service.



Before Virtualization:

- Single OS image per machine
- Software and hardware tightly coupled
- Running multiple applications on same machine often creates conflict
- Underutilized resources
- Inflexible and costly infrastructure



After Virtualization:

- Hardware-independence of operating system and applications
- Virtual machines can be provisioned to any system
- Can manage OS and application as a single unit by encapsulating them into virtual machines

Who provides (server) virtualization software products?

- Commodity x86 CPUs
  - VMWare ESXi Hypervisor / VSphere Platform
  - Linux KVM (Kernel Virtual Machine) Hypervisor
  - Citrix Xen Hypervisor
  - Microsoft Hyper-V Hypervisor
  - Sun / Oracle VirtualBox
  - etc.
- Other CPUs
  - IBM PowerVM Hypervisor
  - IBM System z / VM Hypervisor