Distributed Services: AWS Overview

CS498cc

Mirko Montanari
Jan 18, 2013
Different Types of Cloud Services

- **IaaS**: OS layer, provides basic computational infrastructure
- **PaaS**: middleware layer. Provides a set of services to developers to build their applications
- **SaaS**: application layer. Provides applications to final users
Amazon Web Services

- AWS provides a collection of services for building cloud applications
- Pioneers in cloud computing services (2006)

- Services for:
  - **Storage**: S3, EBS
  - **Computation**: Elastic Cloud Computing (EC2), scaling/load balancer, Elastic Map/Reduce, Elastic Beanstalk
  - **Databases**: RDS, DynamoDB, ElastiCache
  - **Coordination**: Simple Notification Service, Simple Workflow Framework
  - ...
  - All services are paid depending on use

Other service providers

- **Windows Azure**
  - Similar services now, pushing the Platform-as-a-service model (PaaS)
- **Rackspace**
  - Infrastructure-as-a-service, powered by OpenStack (opensource clone of EC2/S3)
- **Google App Engine, Google Compute Cloud, Google Apps**
  - App-engine is platform-as-a-service; Compute is IaaS; Apps is a software-as-a-service
- ...
Overview of Services

Network/Distribution
- CloudFront
- Route 53

Computation
- Elastic Compute Cloud (EC2)
- Elastic MapReduce
- Elastic Beanstalk

Databases
- Relational Database Service (RDS)
- SimpleDB
- DynamoDB
- ElastiCache

Data Storage
- Simple Storage Service (S3)
- Elastic Block Store (EBS)

Auto Scaling
- Elastic Load Balance

Coordination services
- Simple Notification Service (SNS)
- Simple Workflow Service (SWF)
Using AWS Services

• AWS Management Console
  • Easy to use, great for manual configurations
  • Use username / password provided

• Command line tools
  • For writing scripts
    • e.g., create a set of machines to analyze data every day
  • Use access key ID and secret access key, or certificates for EC2

• AWS API
  • Integrating cloud services into your applications
    • e.g., storing data on the cloud, running computation in the background
  • Use access key ID and secret access key, or certificates for EC2

• SSH into EC2 instances is performed using a different keypair
Overview of Services

**Network/Distribution**
- CloudFront
- Route 53

**Computation**
- Elastic Compute Cloud (EC2)
- Elastic MapReduce
- Elastic Beanstalk

**Data Storage**
- Simple Storage Service (S3)
- Elastic Block Store (EBS)

**Databases**
- Relational Database Service (RDS)
  - SimpleDB
  - DynamoDB
  - ElastiCache

**Auto Scaling**
- Elastic Load Balance

**Simple Notification Service (SNS)**
- Simple Workflow Service (SWF)

**Coordination services**
Simple Storage Service (S3)

• First publicly available web service from Amazon (2006)
• Unstructured storage of large amount of data with high reliability
  • Automatically replicated across multiple datacenters
  • write, read, delete data objects
• Storage pay-as-you-store model
  • $0.095 a gigabyte + per-request charges + network bandwidth
• Stores data objects into buckets, each data object is up to 5T
• Data can be read and written through a programming API
  • You can use S3 in your applications as a data storage layer
• Relaxed Eventual Consistency Model
  • If you PUT to an existing key, a subsequent read might return the old data or the updated data, but it will never write corrupted or partial data.

S3 Program Example

• Python + boto

```python
from boto.s3.connection import S3Connection
conn = S3Connection(AWS_KEY, AWS_SECRET)
bucket = conn.get_bucket(BUCKET)
destination = bucket.new_key()
destination.name = filename
destination.set_contents_from_file(myfile)
destination.make_public()
```

https://s3.amazonaws.com/cs498cc-mmontan2/greetings.html
Overview of Services

Network/Distribution
- CloudFront
- Route 53

Computation
- Elastic Compute Cloud (EC2)
- Elastic MapReduce
- Elastic Beanstalk

Data Storage
- Simple Storage Service (S3)
- Elastic Block Store (EBS)

Databases
- Relational Database Service (RDS)
- SimpleDB
- DynamoDB
- ElastiCache

Auto Scaling
- Elastic Load Balance

Simple Notification Service (SNS)
Simple Workflow Service (SWF)

Coordination services
Elastic Block Store (EBS)

- “Cloud-based virtual hard drives”
- Block level storage volumes for use with Amazon EC2 instances
- Off-instance storage, persists independently from the life of an instance
- Can be attached to a running Amazon EC2 instance and exposed as a device within the instance
  - 1 GB to 1 TB
- Amazon CloudWatch exposes performance metrics for EBS volumes, giving insight into bandwidth, throughput, latency, ...
- EBS can be (incrementally) backed up on S3
- Higher throughput than Amazon EC2 instance stores for applications performing a lot of random accesses
- Can attach multiple volumes to an instance and stripe across the volumes (RAID0) to achieve further increases in throughput.
Overview of Services

Network/Distribution
- CloudFront
- Route 53

Computation
- Elastic Compute Cloud (EC2)
- Elastic MapReduce
- Elastic Beanstalk

Data Storage
- Simple Storage Service (S3)
- Elastic Block Store (EBS)

Databases
- Relational Database Service (RDS)
- SimpleDB
- DynamoDB
- ElastiCache

Auto Scaling
- Elastic Load Balance

Coordination services
- Simple Notification Service (SNS)
- Simple Workflow Service (SWF)
Elastic Cloud Computing (EC2)

- Launched in 2006 for providing resizable compute capacity in the cloud
- Rent virtual computers on demand
  - Pay for what you use
    - Hourly rates, e.g., $0.065 an hour for a “small” instance
  - Create new instances within tens of seconds
  - Increase the computing capacity of an instance in minutes

- Most startups and several large organizations use EC2 for running their servers
Amazon Machine Image (AMI)

• Virtual instances boot on a Amazon Machine Image (AMI)
• An image of an operating system ready to boot
  • Amazon Linux; Redhat; Ubuntu Server; Windows;...
  • They might be preconfigured with Apache, Mysql ...
  • Anybody can create AMIs and send them to Amazon
    • It’s created from a snapshot of the files in a computer

• For the class: use “free tier” enabled images.
  • We suggest the Ubuntu Server 12.04.1 LTS
Where are my instances?

- **Regions**
  - Amazon has data centers in different areas of the world (e.g., North America, Europe, Asia, etc.)
  - design an application to be closer to specific customers or to meet legal or other requirements

- Each Region contains multiple distinct locations called **Availability Zones**
- Availability Zones are isolated from failures in others
- Inexpensive, low-latency network connectivity to other zones in the same Region
- Launching instances in separate Availability Zones → protect applications from failure in a single location
Create a new instance

- `ec2-run-instances <ami> -k <keypair> --instance-type <type> -z <region-availability zone>

- Instance Type: Micro (search for free tier)

- **EBS-back**
  - In EBS-backed, your root disk is on a network storage.
  - Stop and restart maintain the data
    - Depending on settings, termination might delete the EBS volume

- **Instance-backed**
  - Everything is stored on the local disk of the machine
  - Data is lost when the machine is stopped / terminated.
  - Limited in what you can change after boot
  - Excellent for temporary jobs that require a local disk space
Using the Instance

• Once the instance starts, it is your computer
  • Users, configurations, servers, it’s all up to the cloud user (you).
  • AMI provides initial configurations, but you can change anything you want

• Accessing the instance:
  • AMI come preconfigured with users
    • “ec2-user” for the Amazon Linux AMIs, “ubutu” for the Ubuntu image
  • At the first boot, Amazon loads a ssh public key that you provide in the user directory so you can log in.
  • After that, you can change anything you want
  • Use ssh to access the instance:

    • ssh -i <privatekey> user@instanceip
Creating the key-pair

• Create your own set of keys for the group
  • You can use the AWS Management Console
    • There are also command line tools and API
  • Put the name of your group in the key pair, so that you know which one you should use

• The private key can be downloaded only once, so put it in a safe place
  • If you lose the private key, you can create a new keypair.
  • However, you might not be able to access instances created with your previous key
Configuring Firewalls

- Instances are put into “security groups”
- Each security group defines a set of firewall rules on who can connect to the instance
  - Make sure that port 22 (ssh) is open so you can log in the instance.
  - If you are running additional services you might need to add more rules
    - e.g., port 80 for HTTP traffic
    - format for IP ipaddress/length of the netmask
- Create your own security group with the group name in it
EC2 API

- Python + boto

```python
from boto.ec2.connection import EC2Connection
conn = EC2Connection('<AWS_ACCESS_KEY_ID>', '<AWS_SECRET_ACCESS_KEY>)

conn.run_instances(
    '<ami-image-id>',
    key_name='myKey',
    instance_type='c1.xlarge',
    security_groups=['your-security-group-here'])
```

- Java interface is a more syntactically complex, but semantically it’s the same
Shared Account for the Class

- Tag instanced with your group name when you create them
  - Everybody in the class access the same set of instances and no access control
  - so please please be considerate and careful!

- When listing the instances, put the group name in the search box so you only see your own

- Stop instances when you are not using them
Overview of Services

Network/Distribution
- CloudFront
- Route 53

Computation
- Elastic Compute Cloud (EC2)
- Elastic MapReduce
- Elastic Beanstalk

Data Storage
- Simple Storage Service (S3)
- Elastic Block Store (EBS)

Databases
- Relational Database Service (RDS)
- SimpleDB
- DynamoDB
- ElastiCache

Auto Scaling
- Elastic Load Balance

Coordination services
- Simple Notification Service (SNS)
- Simple Workflow Service (SWF)
Auto Scaling & Elastic Load Balance

• Auto Scaling
  • Monitor the load on EC2 instances using CloudWatch
  • Define Conditions
  • Spawn new instances when there is too much load or remove instances when not enough load

• Elastic Load Balance
  • Automatically distributes incoming application traffic across multiple EC2 instances
  • Detects EC2 instance health and divert traffic from bad ones
  • Support different protocols
    • HTTP, HTTPS, TCP, SSL, or Custom
  • They can work together
Overview of Services

Network/Distribution
- CloudFront
- Route 53

Computation
- Elastic Compute Cloud (EC2)
  - Elastic MapReduce
  - Elastic Beanstalk
- Auto Scaling
- Elastic Load Balance
- Simple Notification Service (SNS)
- Simple Workflow Service (SWF)

Data Storage
- Simple Storage Service (S3)
- Elastic Block Store (EBS)

Databases
- Relational Database Service (RDS)
  - SimpleDB
  - DynamoDB
  - ElastiCache

Coordination services
Elastic MapReduce (EMR)

- EMR utilizes a hosted Hadoop framework running on the web-scale infrastructure of Amazon EC2 and Amazon S3

1. Write your Hadoop program in Java
2. Submit the jar for to EMR
3. Store the input in S3
4. Tell EMR to run it (web interface or CLI)
5. EMR runs it and stores the results back in S3

It takes up to 10 minutes to start your job, EMR looks for unused resources to minimize the costs
Overview of Services

**Network/Distribution**
- CloudFront
- Route 53

**Computation**
- Elastic Compute Cloud (EC2)
- Elastic MapReduce
- Elastic Beanstalk

**Data Storage**
- Simple Storage Service (S3)
- Elastic Block Store (EBS)

**Databases**
- Relational Database Service (RDS)
  - SimpleDB
  - DynamoDB
  - ElastiCache

**Auto Scaling**
- Elastic Load Balance

**Coordination services**
- Simple Notification Service (SNS)
- Simple Workflow Service (SWF)
Elastic Beanstalk

- Solution for Enterprise server-side java application deployment
  - Write a Tomcat app, let Amazon deploy it, scale it when traffic increases, and detect failures.

- Create your normal Tomcat Java Web Application (e.g. Eclipse).
- Upload your application code in as WAR file.
- Deploy the application
  - Elastic Beanstalk handles the provisioning of a load balancer and the deployment of the WAR file to one or more EC2 instances running the Apache Tomcat application server

- Access the application at a customized URL (e.g. http://myapp.elasticbeanstalk.com/).
Overview of Services

- **Network/Distribution**
  - CloudFront
  - Route 53

- **Computation**
  - Elastic Compute Cloud (EC2)
  - Elastic MapReduce
  - Elastic Beanstalk

- **Data Storage**
  - Simple Storage Service (S3)
  - Elastic Block Store (EBS)

- **Databases**
  - Relational Database Service (RDS)
  - SimpleDB
  - DynamoDB
  - ElastiCache

- **Auto Scaling**
  - Elastic Load Balance

- **Simple Notification Service (SNS)**

- **Simple Workflow Service (SWF)**

- **Coordination services**
Relational Database Service (RDS)

- Preconfigured EC2 instances with MySQL or Oracle installed
  1. Create an RDS instance
  2. Dump your database into it
     - `mysqldump acme | mysql --host=hostname --user=username --password acme`
  3. Update SQL connection strings in your application (which might be running anywhere, including EC2 VMs)

- Features
  - Pre-configured
  - Monitoring and Metrics (CloudWatch)
  - Automatic Software Patching
  - Automated Backups
  - DB Snapshots
  - Changing the instance type ( = increase computer power)
    - Through EBS snapshots
  - Multi-AZ Deployments
  - Read Replicas
    - Scaling for read-heavy database workloads
  - Isolation and Security
SimpleDB

- A NoSQL database, non-relational
- Eventual consistency or strong consistency, depending on the request
- Data model is comprised of domains, items, attributes and values
  - Large collections of items organized into domains
  - Items are little hash tables containing attributes of key, value pairs
- Use Put, Batch Put, & Delete to create and manage the data set
- Use GetAttributes to retrieve a specific item
- Attributes can be searched with various lexicographical queries
- The service manages infrastructure provisioning, hardware and software maintenance, replication, indexing of data items, and performance tuning
- Tables limited to 10 GB, typically under 25 writes/second
- User manages partitioning and re-partitioning of data over additional SimpleDB tables

<table>
<thead>
<tr>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexes all the attributes</td>
</tr>
<tr>
<td>Uses less dense drives</td>
</tr>
<tr>
<td>Better optimized for random access</td>
</tr>
</tbody>
</table>
DynamoDB

- Amazon Dynamo paper (2007) → Open-source Apache Cassandra project → DynamoDB (1/2012)
  - Dynamo is a highly available, key-value structured storage system
- Fully managed NoSQL non-relational Database
- Data model is comprised of domains, items, attributes and values (similar to SimpleDB)
  - Domains are collections of items that are described by attribute-value pairs
- **Pay by reserved throughput + indexed storage**
- Integrates with Hadoop MapReduce using Elastic MapReduce
- Run on solid state disks (SSDs)
- There are no limits on the request capacity or storage size for a given table.
  - DynamoDB automatically partitions data and workload over a sufficient number of servers to meet the scale requirements
Overview of Services

Network/Distribution
- CloudFront
- Route 53

Computation
- Elastic Compute Cloud (EC2)
- Elastic MapReduce
- Elastic Beanstalk

Data Storage
- Simple Storage Service (S3)
- Elastic Block Store (EBS)

Databases
- Relational Database Service (RDS)
  - SimpleDB
  - DynamoDB
  - ElastiCache

Auto Scaling
- Elastic Load Balance

Coordination services
- Simple Notification Service (SNS)
- Simple Workflow Service (SWF)

CloudWatch
Amazon CloudWatch

- Monitor AWS resources automatically
  - Monitoring for Amazon EC2 instances: seven pre-selected metrics at five-minute frequency
  - Amazon EBS volumes: eight pre-selected metrics at five-minute frequency
  - Elastic Load Balancers: four pre-selected metrics at one-minute frequency
  - Amazon RDS DB instances: thirteen pre-selected metrics at one-minute frequency
  - Amazon SQS queues: seven pre-selected metrics at five-minute frequency
  - Amazon SNS topics: four pre-selected metrics at five-minute frequency
- Custom Metrics generation and monitoring
- Set alarms on any of the metrics to receive notifications or take other automated actions
- Use Auto Scaling to add or remove EC2 instances dynamically based on CloudWatch metrics
Summary

• AWS provides a diverse set of services that permits the creation of scalable applications
• Many of cloud providers provide similar services
  • Storage; Computation; Databases; and other frameworks for building applications

• The MPs will provide opportunity to experiments some of the services.