CS15-319 / 15-619
Cloud Computing

Recitation 4
February 4th & 6th, 2014
Announcements

• Encounter a general bug:
  – Post on Piazza publicly

• Encounter a grading bug:
  – Post Privately on Piazza

• Do not post your code on Piazza

• Post feedback on OLI

• Do not forget to Tag your instances
  – Key: Project       Value: 2.1
Last Week

• You have completed
  – Sequential Analysis
  – Elastic MapReduce

• You should have learned
  – Why MapReduce for big data
  – How MapReduce works
  – How to write your own Mapper/Reducer
Project 1.2 Checkpoint

• We will manually grade Question 1
  – Be sure to make your code readable
  – We will not deduct points for style
    • ONLY this time!
Piazza Questions

- Elastic MapReduce Billing Question
  - Normalized Hours (Elastic MapReduce)

<table>
<thead>
<tr>
<th>Date</th>
<th>Elapsed Time</th>
<th>Normalized Instance Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:59 EDT</td>
<td>1 hour 46 minutes</td>
<td>40</td>
</tr>
</tbody>
</table>

- 1 hour of m1.small = 1 hour normalized compute time
- 1 hour of m1.medium = 2 hours normalized compute time
- 1 hour of m1.large = 4 hours normalized compute time
- 1 hour of m1.xlarge = 8 hours normalized compute time
- 1 hour of c1.medium = 2 hours normalized compute time
Piazza Questions

- Elastic MapReduce Billing Question
  - Elastic MapReduce Pricing (On demand)
  - for US East (N. Virginia)

<table>
<thead>
<tr>
<th>Standard On-Demand Instance</th>
<th>Amazon EC2 Price</th>
<th>Amazon Elastic MapReduce Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>$0.06 per Hour</td>
<td>$0.015 per Hour</td>
</tr>
<tr>
<td>Medium</td>
<td>$0.12 per Hour</td>
<td>$0.03 per Hour</td>
</tr>
<tr>
<td>Large</td>
<td>$0.24 per Hour</td>
<td>$0.06 per Hour</td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.48 per Hour</td>
<td>$0.12 per Hour</td>
</tr>
</tbody>
</table>

Billing price = ($0.12 * 4 instances + $0.03) * 2 hours
Piazza Questions

• Late policy
  – We do not have a late policy!
    • All deadlines are hard.
  – Please start early!
Piazza Posts

![Graph showing the trend of Piazza Posts from 01/11 to 02/01. The graph indicates an increase in posts, peaking around 01/25 and decreasing sharply thereafter.]
# Piazza Traffic

Top Student Answerers

<table>
<thead>
<tr>
<th>Name, Email</th>
<th>number of answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Haibo Wang <a href="mailto:haibow@andrew.cmu.edu">haibow@andrew.cmu.edu</a></td>
<td>7</td>
</tr>
<tr>
<td>Benjamin Gummoe <a href="mailto:bgummoe@andrew.cmu.edu">bgummoe@andrew.cmu.edu</a>, <a href="mailto:b.gummoe@gmail.com">b.gummoe@gmail.com</a></td>
<td>7</td>
</tr>
<tr>
<td>Pavan Kumar Sunder <a href="mailto:psunder@andrew.cmu.edu">psunder@andrew.cmu.edu</a></td>
<td>7</td>
</tr>
<tr>
<td>Patrick Woody <a href="mailto:pwoody@andrew.cmu.edu">pwoody@andrew.cmu.edu</a></td>
<td>6</td>
</tr>
<tr>
<td>Karan Sikka <a href="mailto:ksikka@cmu.edu">ksikka@cmu.edu</a></td>
<td>6</td>
</tr>
<tr>
<td>Sairam Krishnan <a href="mailto:sbkrishn@andrew.cmu.edu">sbkrishn@andrew.cmu.edu</a></td>
<td>5</td>
</tr>
<tr>
<td>Jack Wang <a href="mailto:hojehw@andrew.cmu.edu">hojehw@andrew.cmu.edu</a>, <a href="mailto:dilta.ppc@gmail.com">dilta.ppc@gmail.com</a></td>
<td>5</td>
</tr>
<tr>
<td>Dylan Fitzpatrick <a href="mailto:djfitzpa@andrew.cmu.edu">djfitzpa@andrew.cmu.edu</a></td>
<td>4</td>
</tr>
<tr>
<td>David Curd <a href="mailto:dcurd@andrew.cmu.edu">dcurd@andrew.cmu.edu</a></td>
<td>4</td>
</tr>
<tr>
<td>Taylor Makela <a href="mailto:tmakela@andrew.cmu.edu">tmakela@andrew.cmu.edu</a></td>
<td>4</td>
</tr>
</tbody>
</table>
This Week

• Unit 2: Data Centers
  – Module 3: Data Center Trends
  – Module 4: Data Center Components

• Read and complete:
  – Module 5: Design Considerations
  – Unit 2: Checkpoint Quiz

  • 150 minutes, due 6\textsuperscript{th} Feb., 2014 (Pittsburgh Time)
This Week

• Project 2.1: Introduction to APIs
  – Single Instance Benchmarking

• Start early!
  – Project 2.1, due 9th February, (Pittsburgh time)
  – You have three attempts, but no late submission
• Started with 12 engineers, now 140+ employees
• Total number of users: 70 million
• Growth in web traffic from 9/12-9/13: 66.52%
• Utilizes AWS
  – Auto Scaling
    • Scale up and down
  – Elastic Load Balancer
    • Distribute traffic
  – S3
    • File storage
  – Elastic MapReduce
    • Data analysis
Project 2 - Scenario

• Online Photo Verification, *PixVerify*

Sales

Reliability
Current Infrastructure

User

Application Server

Database Server
Overview

Auto Scaling Group

Users
Cloud Infrastructure

Instance Types
- Small
- Medium
- Large

Middleware (Virtualization)

Bare Metal Resources
Maximize Requests per Dollar

**GOAL:** MAXIMIZE (REQUESTS/DOLLAR)
Load Testing Request & Response Flow

Launchpad

Benchmark Script

CloudWatch

Small

Medium

Large
Amazon APIs

• Provisioning & Monitoring:
  – EC2 API
  – CloudWatch API

• Supported APIs
  – Command Line Interface API Tools
  – AWS SDK for Java
  – AWS SDK for Python
Load Testing Request & Response Flow

Steps:
- Launch the Launchpad
- Upload script/program
  - Provision test instance
  - Run benchmark script
  - CloudWatch to retrieve CPUUtilization
  - Store benchmark output into a file
  - Terminate test instance
  - Repeat for other test instances
- Requests/second & CPUUtilization
Tips

• Make sure the Launchpad and other instances are in the same subnet (availability zone)
Tips

• Terminate instances vs. Stop instances
  – Stop will charge for storage (eg. EBS volumes)
  – Record all the data and results your need before you terminate an instance.

• S3 URL submission
  – Verification link
    • http://ec2-54-225-106-182.compute-1.amazonaws.com/
  – Do not include a space in the S3 link

• Start early!
Upcoming Deadlines

• Unit 2: **Timed Quiz (150 minutes)**

**UNIT 2: Data Centers**
- Module 3: Data Center Trends
- Module 4: Data Center Components
- Module 5: Design Considerations

**Quiz 2: Data Centers**
- **Checkpoint**
- **Available 2/03/14 12:01 AM**
- **Due 2/06/14 11:59 PM**

• **Project 2: Tag - Key: Project**  **Value: 2.1**

**Project 2**
- **Introduction and APIs**
- **Single Instance Benchmarks**
- **Checkpoint**
- **Available Now**
- **Due 2/09/14 11:59 PM**
Demo Outline

• 1. Launch an instance using command line
  – Setup command line tool environment
  – Launch an instance
  – Demonstrate CloudWatch command line

• 2. Launch an instance using Java API
  – Install AWS SDK in Eclipse
  – Configure AWS account in Eclipse
  – Demonstrate Amazon EC2 API
Setup ec2 command line tool

- Download and Copy ec2 command line tool into the ec2 instance
- Configuration:
  - Make sure that JAVA_HOME environment variable is set
  - Set EC2_HOME & system PATH variable
  - Configure your access key and secret key
  - Set your private key & certificate
Launch an instance

- `ec2-run-instances ami_id [-n instance count] [-k keypair] [--instance-type instance_type] [--availability-zone zone]`

CloudWatch Command Line

• Download CloudWatch

• Fill your access key and secret key in
  `$AWS_CLOUDWATCH_HOME/credential-file-path.template`

• You will have to figure out how to use the command line
  – Hint: mon-get-stats
Install AWS Toolkit for Eclipse

- Plug-in for the Eclipse Java IDE
- Help -> Install New Software
- Enter ‘http://aws.amazon.com/eclipse’ in ‘Work with’
- Select ‘AWS Toolkit for Eclipse’
- http://aws.amazon.com/eclipse/
Amazon EC2 API

• Load AWS Credentials
  – BasicAWSCredentials
• Create Amazon EC2 Client
  – AmazonEC2Client
• Create and Configure Instance Request
  – RunInstancesRequest
• Launch Instance
  – RunInstanceResult
Amazon CloudWatch API

- Load AWS Credentials
  - BasicAWSCredentials
- Create Amazon CloudWatch Client
  - AmazonCloudWatchClient
- Create and Configure Metric Request
  - GetMetricStatisticsRequest
- Get Metric Result
  - GetMetricStatisticsResult
Important Notes

1. Run your program of project 2.1 on the same directory of benchmark;

2. Figure out how to
   a. Set security group for your instance;
   b. Set availability zone for your instance;
   c. Tag your instance;
   d. Use CloudWatch;

3. Start Early;

4. Q & A.