15-319 / 15-619
Cloud Computing

Recitation 6
February 17th & 19th, 2015
Overview

• **Administrative issues**
  Office Hours, Piazza guidelines

• **Last week’s reflection**
  Project 2.2, OLI unit 3 module 6 and 7

• **This week’s schedule**
  – Project 2.3 - Feb 22nd
  – Unit 3 - Module 8 - Resource Virtualization - CPU

• **Demo**
Announcements

• Monitor AWS expenses regularly
  - Check your bill frequently (use Cost Explorer and filter by tags).

• Terminate your resources when not in use
  - Stop still costs EBS money ($0.1/GB/Month)
  - Amazon EC2 and Amazon Cloudwatch fees for monitoring, ELB
  - Autoscaling group - no additional fees

• Use spot instances
Last Week’s Reflection

• Content
  - Unit 3 - Modules 6 and 7 Virtualizing Resources of the cloud
  - Quiz 2 completed
• EC2 APIs
  - Amazon CLI, Java, Python
• Load Balancing and Autoscaling
  - Experience ASG (horizontal scaling) on AWS
  - Manage cloud resources and deal with failures using programs.
Types of Failures

Elevated Error Rates

**ISSUE:** We are continuing to monitor performance issues affecting some customers. We have made changes which should mitigate the problem for most customers and are continuing to monitor the situation.
FEB 08, 2011 – 21:39 UTC – 19 MINUTES AGO

**ISSUE:** We are continuing to monitor sporadic reports of poor performance.
FEB 08, 2011 – 21:11 UTC – 42 MINUTES AGO

**ISSUE:** We are continuing to investigate poor performance in the platform.
FEB 08, 2011 – 20:40 UTC – 74 MINUTES AGO

**ISSUE:** We are continuing to investigate poor performance in the platform.
FEB 08, 2011 – 20:40 UTC – 74 MINUTES AGO

**ISSUE:** We are investigating elevated error rates on the platform.
FEB 08, 2011 – 20:09 UTC – 105 MINUTES AGO

Transient Failure

Permanent Failure
### Project 2.2

- **Manual Grading: 20 Points are for the code**
  - Always make sure that your code is readable
  - Follow style presented in Recitation 2

<table>
<thead>
<tr>
<th>Violation</th>
<th>Penalty of the project grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spending more than $10 for this project phase</td>
<td>-10%</td>
</tr>
<tr>
<td>Spending more than $20 for this project phase</td>
<td>-100%</td>
</tr>
<tr>
<td>Failing to tag all your resources(EC2 instances, ELB, ASG) for this project</td>
<td>-10%</td>
</tr>
<tr>
<td>Submitting your AWS credentials in your code for grading</td>
<td>-10%</td>
</tr>
<tr>
<td>Using instances other than m1/m3.small/medium/large</td>
<td>-10%</td>
</tr>
</tbody>
</table>
This Week: Content

• UNIT 3: Virtualizing Resources for the Cloud
  – Module 6: Introduction and Motivation
  – Module 7: Virtualization
  – **Module 8: Resource Virtualization - CPU**
  – Module 9: Resource Virtualization - Memory
  – Module 10: Resource Virtualization – I/O
  – Module 11: Case Study
Resource Virtualization - CPU

- Virtualizing an ISA
- Full Virtualization and Paravirtualization
- Emulation
- Virtual CPU
This Week: Project

- **P2.1: Introduction and APIs**
  - MSB Recruitment Exam

- **P2.2: Autoscaling and Elastic Load Balancing**
  - Junior System Architect at the MSB

- **P2.3: Advanced Scaling Concepts: Caching**
  - Speed up a web-service using caching
Caching - I

Old idea, used across many layers within a single machine and across machines. An example of caching between CPU and MM:

CPU

Word 32 / 64 bits

CACHE

Block, ~128 bytes

MAIN MEMORY

Page, ~4096 bytes

DISK
CPU request for a record

Block containing record in cache?

Access main memory for block with record

NO

Allocate cache line for this main memory block

YES

Fetch from Cache (fast)

DONE

Deliver record to CPU
Caching - III

- Exploiting locality
  - Temporal locality
  - Spatial locality

- Things to consider in caching
  - Which data to fetch?
  - Which block to invalidate during writing?
  - Eviction policy, when you run out of space?
P2.3 - This Week Speed Up Webservice

YOU

Find Target ID: 1234

Get 1234

LOCAL MEMORY <EMPTY>

Front-end

Back-end Server 1 (DCI1)

Back-end Server 2 (DCI2)
Web Service: Before Caching

A. BEFORE CACHING

MSB Agent

2 ms
1. Find TargetID
1234

2 ms
3. Respond & cache

Local Cache Memory
<EMPTY>

Your Front-end

1 ms
2. Get
1234

10 ms

MSB Backends
Web Service: After Caching

B. AFTER CACHING

MSB Agent

Your Front-end

2 ms
1. Find TargetID
1234

2 ms
2. Respond from cache

Local Cache Memory
1234:<1234_DATA>

MSB Backends
P2.3 - what you have to do

• Speed up the web-service using caching
  ○ Size of cache on backend and frontend is predetermined
  ○ Learn the characteristics of the traces
  ○ Skeleton code provided gives low RPS
    - Trace 1: Skeleton code RPS ~ 65
      Target RPS ~ 1300

Note: You may only cache up to 1000 records at a time on the front end.
Querying a single record from the DCI

URL: `<DCI-DNS>/target?targetID=1`

Response:

```
Target 1 Daniel De Rossi's encrypted conversations are:
97E9271445034E690549230452A0C234BF22B47020556211A003E1507A7C82
081CA481611AAF53D47000241260549230652B7C822ABEF210B0A0B4911A0C6
4321495425131DB6CA29E3F13C080F0943301A0038151BAFC43BAAEF2947044
EE23171018492A0E003E0400B5C422A6F26E06170900260C49230552A2C92EB
01E3CC2FA7A127094501412A100290E1FA2C42FB0AF6E340C015028300C6D0
6408536D0052B0C833B5E82D02450356211B002C411CA6D936ACF325B451B4
2602174C542C084E6D0E05ADC825EFA12CE450D4E640C4E294107B0C833E3E
A6DB2EAFF72B02450F4F29195359081CA48D35A6F3230E0B034C2BOE596D0E0
06466D021DAED3487E8200451E45376535F0217B0836180ED212014C32
47160952320C523E4113AD6961B0EE2813120D5221494E281505ACDF2A0B0A13
C324E3E02D04001F53641D4F6D021DAED3487E43C4716095232043281252A
536D0C17B0DE2A04E4624715004521A456D041FA2C42DE3F5262450F4F311
2A0243D4954250452A5C433B0F56E010A1A45641A5438051A1DD932E3F32B0
60E2A06E33D0D55364064E60D1BADC861A0EE3B15160900230056281252B0D
201107BC42FA4AD0E0E111F002107412F0D1BADC6187E420D0F0B034C2B0E4
1E49210743284106ABDF2EB6E26647151E4F2E0C43391252B6D928A8E340E0
B6E56E040A015031D49230652B0C833B5E82D02164C41360C002F041BADCA6
151BAC332E3E0200345054E640441231852A7C22CA2E820144B
```
Querying multiple records from the DCI

URL: <DCI-DNS>/range?
start_range=1&end_range=3

Response:

Target 1 Daniel De Rossi's encrypted conversations are:
97E92145236E90954253042A0C240342F122D473052611A03E1507A7C82FB7F260E08
58D02AEE3B034524F4F2911853391CA48161A4F39D4700241260692306527C822ABEF
0E118A6DE6E3E582003458412A0D53600E1E3C8393B2E43C0E0024321495452131DB6CA2
B4C24923062A2C2E1B3545B14920064C34113A0DF2EBF2E60415A4136004539
194509412A100298E1FA242FB0AF6340C815028100C6D021EACD825E32121A1200142D6
F13B130E024764036D0052B0C8335E8B200244035621100B02C411CA6D936ACF32548510
F22A6F26E617090036C4E390416F8BD3A2F52602174542084E6D0E05ADC825EF12C
0EBD8D02AFEE3B03450F4F2919553981CA48D28D0A12F4717094321075B4628B69
96D0B0E3C024B7E0330F80A1DEB020685328852540CC351B0185396040E294114A0C
2786553F217B0836180ED212014C432B045638151B0ADCA61A3EF3808091A4537494281
609523620C5234E113ACD618B0E2813120D5212494E281505ACD2F2A0A1481802054
2F00084C412A000822011E1A3C324E3E2084001F53641D4F6D201DAEDD3E743C4716951
EB57A1F150804C4126546651D2E3C924A8F3371114C42500360D1780E20A4E64247:
B7CC27A5A12880174C41645C5650D1DADD5B3E3E04711044564085355E815DAADC24AD!
80517ADD932EF3F9280400565214452058081E352EFA2DF43D49452F4F2A0E522C150AFC:
064E6001B0AD6C610A0E3B1516090230056281252B0D8347E4201364C12A4949F380406
9446424A4F20116787C424F44ADDE0E11FF002017412FD01BDAC61B7E42D0FB03C2B8E49:
4784B24646014120501EECC2F3E436170E4921697432841068DF2EB602647151042F:
0ED212104C08065441370E1CE3F3A241A11D20171A49270C53644F5280C12E656E40A0
CA161A2E5211710944614E929041EB8D0204F32114164C416414F42F38081B7D61ACE76B:
8527A2CE42E821448B;Target 2 Abner Felipe Souza De's encrypted conversations
150A0716524F28480001E000030000812456420493791E451A101425070111010270E
352632A6A190D50264F3E1F00E084234902818E450C8A023E807801503200F611E9
0A8F02034900320115A09412A81536C0303450C1C11241907081C432345180128055340:
20949304F1616E552045000010C1F060A4116070C4521435143050A15160974D3F3616E8:
4F171F084E23454125031D110C004136080A001E366040F1B1F1653730E550C00522D0E854

Page 18
Skeleton Code Provided

Skeleton code in java (**MSB.java**) and python (**MSB.py**)
- You can get full score by optimizing this function.
- Feel free to optimize other parts too!

```java
/*
 * retrieveDetails - you have to modify this function to achieve a higher RPS value
 * Input: the targetID
 * Returns: The result from querying the database instance
 */
private String retrieveDetails(String targetID) {
    try{
        return sendRequest(generateURL(0, targetID));
    } catch (Exception ex){
        System.out.println(ex);
        return null;
    }
}
```
Resources


Latency comparison with Caching

Query the frontend web-service:
Get record for targetID = 2000

Without the record cached: 0.035s
With the record cached at backend: 0.020s
With the record cached at front-end: 0.004s
Project 2.3 Hints - 1

- Caches will be effective if the memory accesses exhibit good locality of references

- Temporal locality - resources accessed in close points in time.

- Spatial Locality - likelihood that a resource will be accessed that is near a resource that was just accessed
Project 2.3 Hints - 2

• Read project description more than once
• Think about workflow before starting
• Read the Hints section! It *might* be useful
• When you are working on the caching code, stop the load generator and data center instances to save costs.
• Start early. 🐔🌞
## Project 2.3 Penalties

<table>
<thead>
<tr>
<th>Violation</th>
<th>Penalty of the project grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spending more than $10 for this project phase</td>
<td>-10%</td>
</tr>
<tr>
<td>Spending more than $20 for this project phase</td>
<td>-100%</td>
</tr>
<tr>
<td>Failing to tag all your resources for this project</td>
<td>-10%</td>
</tr>
<tr>
<td>Submitting your AWS credentials in your code for grading</td>
<td>-10%</td>
</tr>
<tr>
<td>Using instances other than the ones specified</td>
<td>-100%</td>
</tr>
<tr>
<td>Caching more than 1000 records in the front end</td>
<td>-100%</td>
</tr>
</tbody>
</table>
Upcoming Deadlines

- Project 2.3: Advanced Scaling Concepts - Caching
  Due: 02/22/15 11:59PM Pittsburgh

- Unit 3: Virtualization Resource for the Cloud Module
  8: Resource Virtualization - CPU

- 15619Project Team Formation Deadline
  Due: Friday 02/20/15 11:59PM Pittsburgh

- Quiz 3: Unit 3 - Virtualizing Resources
  Due: Next Week (Start reading, long unit)
Upcoming Deadline

Form your 15619Project teams and submit your team info to:

http://bit.ly/1EFAsdO

- Once submitted, teams are final and binding
There will also be a report due at the end of each phase, where you are expected to discuss optimizations.
Find Target ID: 1234

Front-end
(You need to optimize the cache)

Back-end Server 1 (DCI1)

Get 1234

Back-end Server 2 (DCI2)

LOCAL MEMORY
<EMPTY>
Project 2.3: Load Generator UI

Welcome to MSB Load Generator, mshishod!

Test logs
Trace files
Project 2.3: Data Center UI

Welcome to MSB Data Center, mshishod!

- Track Person
- Track Device
- Track Target
- View log