

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



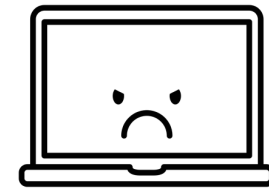
- Monitor AWS expenses regularly
 - Check your bill frequently (use Cost Explorer and filter by tags).
- Terminate your resources when not in use
 - Stop still costs EBS money (\$0.1/GB/Month)
 - Amazon EC2 and Amazon Cloudwatch fees for monitoring, ELB
 - Autoscaling group - no additional fees
- Use 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



heroku :: status

APP OPERATIONS TOOLS

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 – 21:38 UTC – 16 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

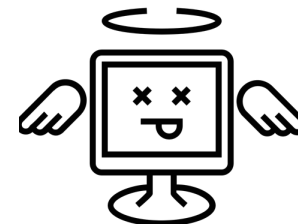
ISSUE: We are investigating elevated error rates on the platform.
FEB 08, 2011 – 20:09 UTC – 105 MINUTES AGO



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

Violation

Penalty of the project grade

Spending more than \$10 for this project phase

-10%

Spending more than \$20 for this project phase

-100%

Failing to tag all your resources(EC2 instances, ELB, ASG) for this project

-10%

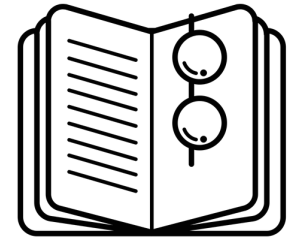
Submitting your AWS credentials in your code for grading

-10%

Using instances other than m1/m3.small/medium/large

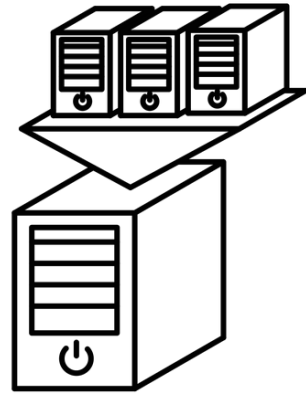
-10%

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

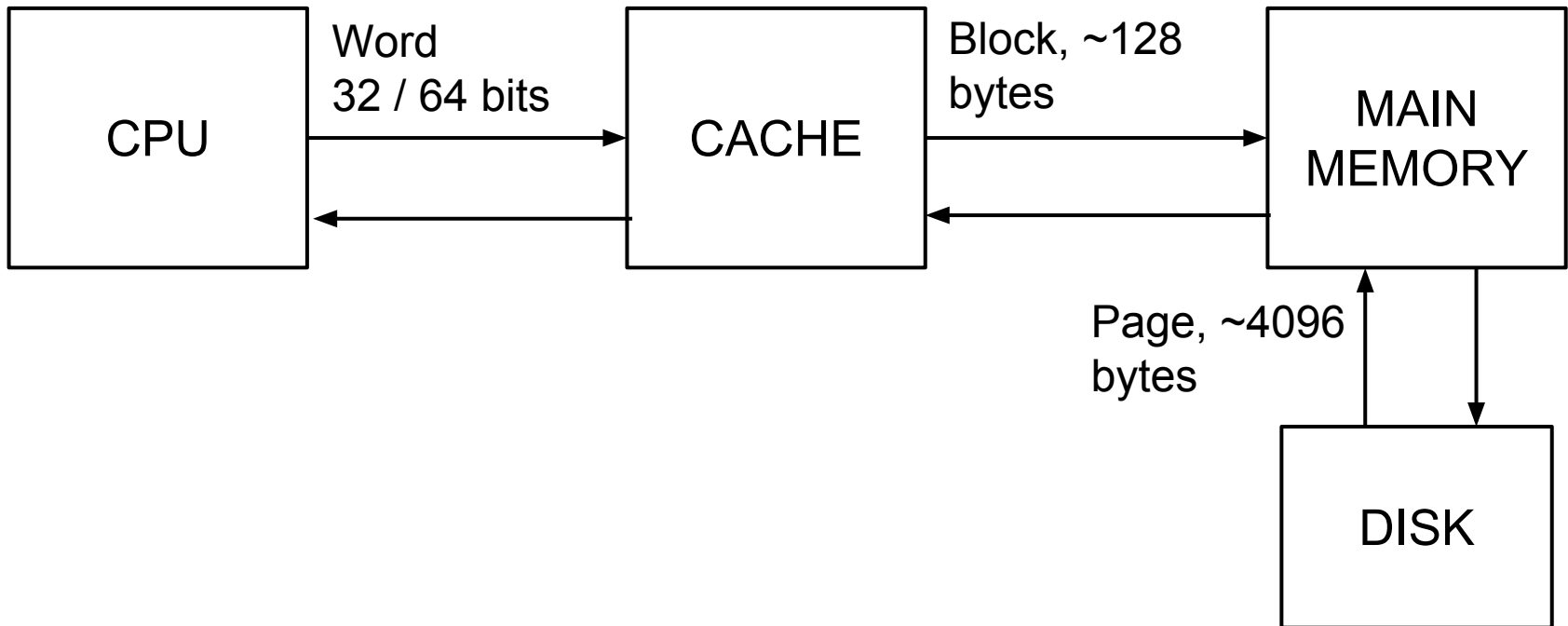
This Week: Project



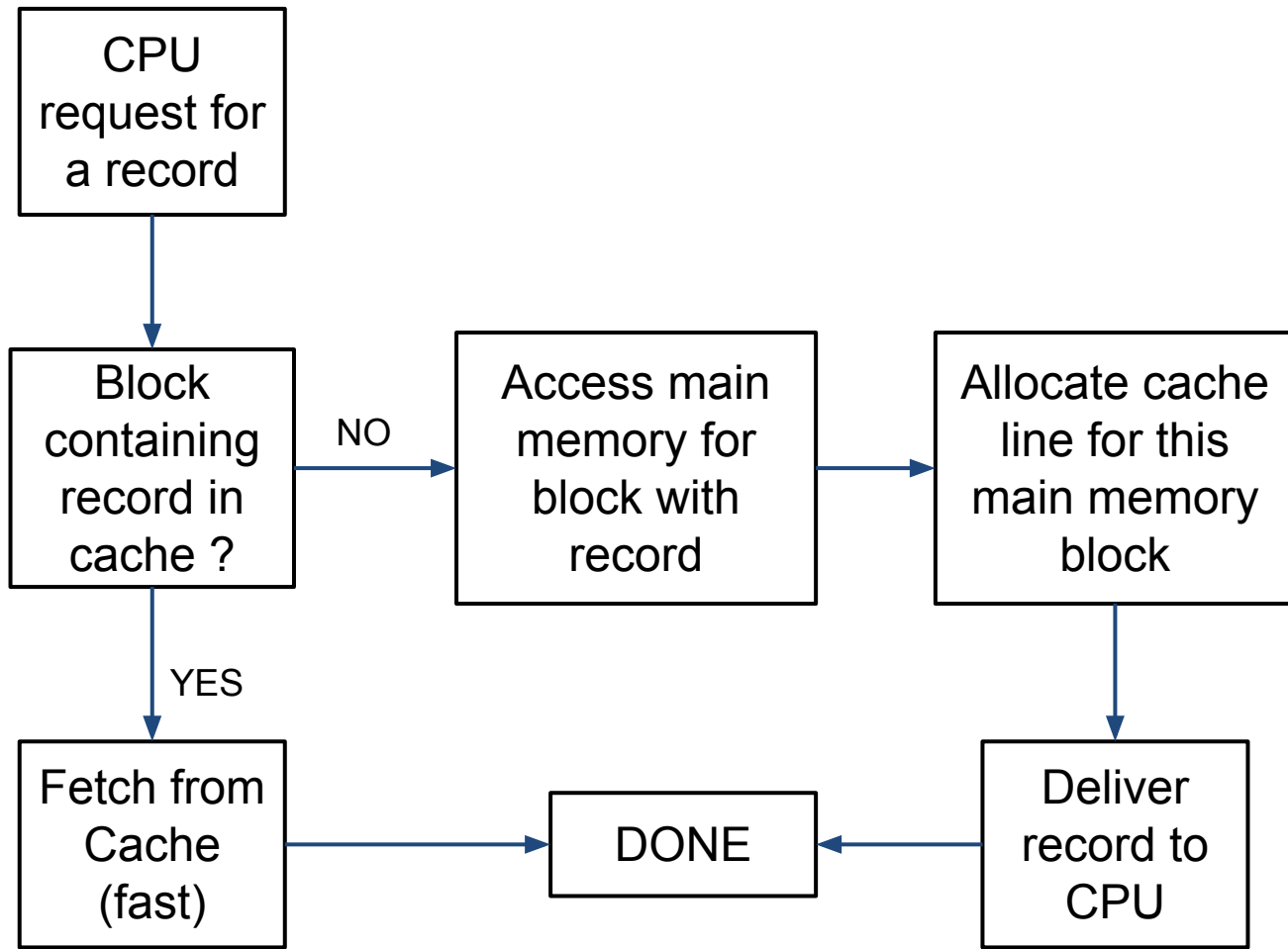
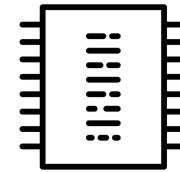
- P2.1: Introduction and APIs
 - MSB Recruitment Exam
- P2.2: Autoscaling and Elastic Load Balancing
 - Junior System Architect at the MSB
- P2.3: Advanced Scaling Concepts: Caching
 - Speed up a web-service using caching

Caching - I

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



Caching - II

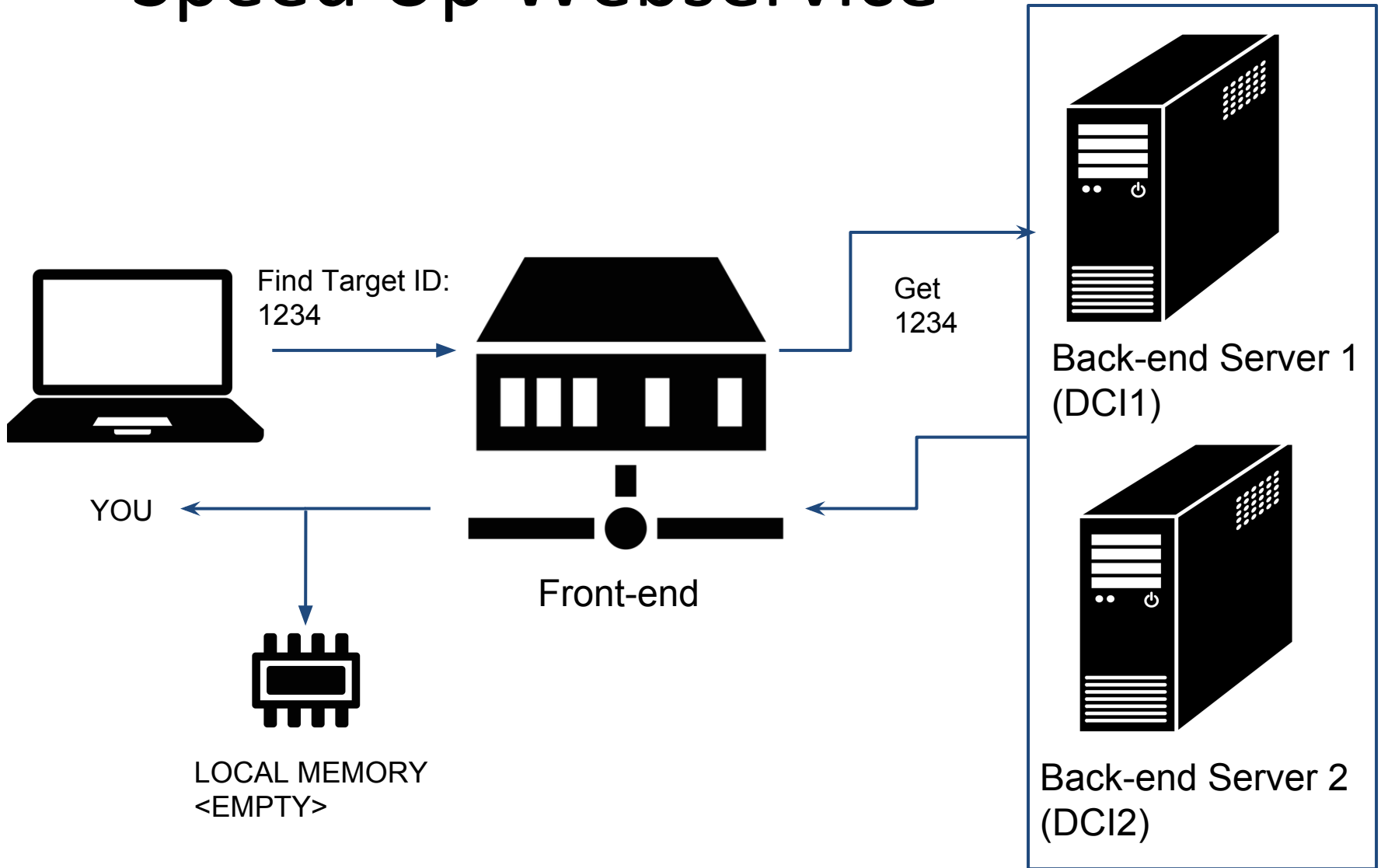


Caching - III

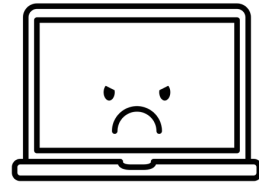
- Exploiting locality
 - Temporal locality
 - Spatial locality
- Things to consider in caching
 - Which data to fetch?
 - Which block to invalidate during writing?
 - Eviction policy, when you run out of space?

P2.3 - This Week

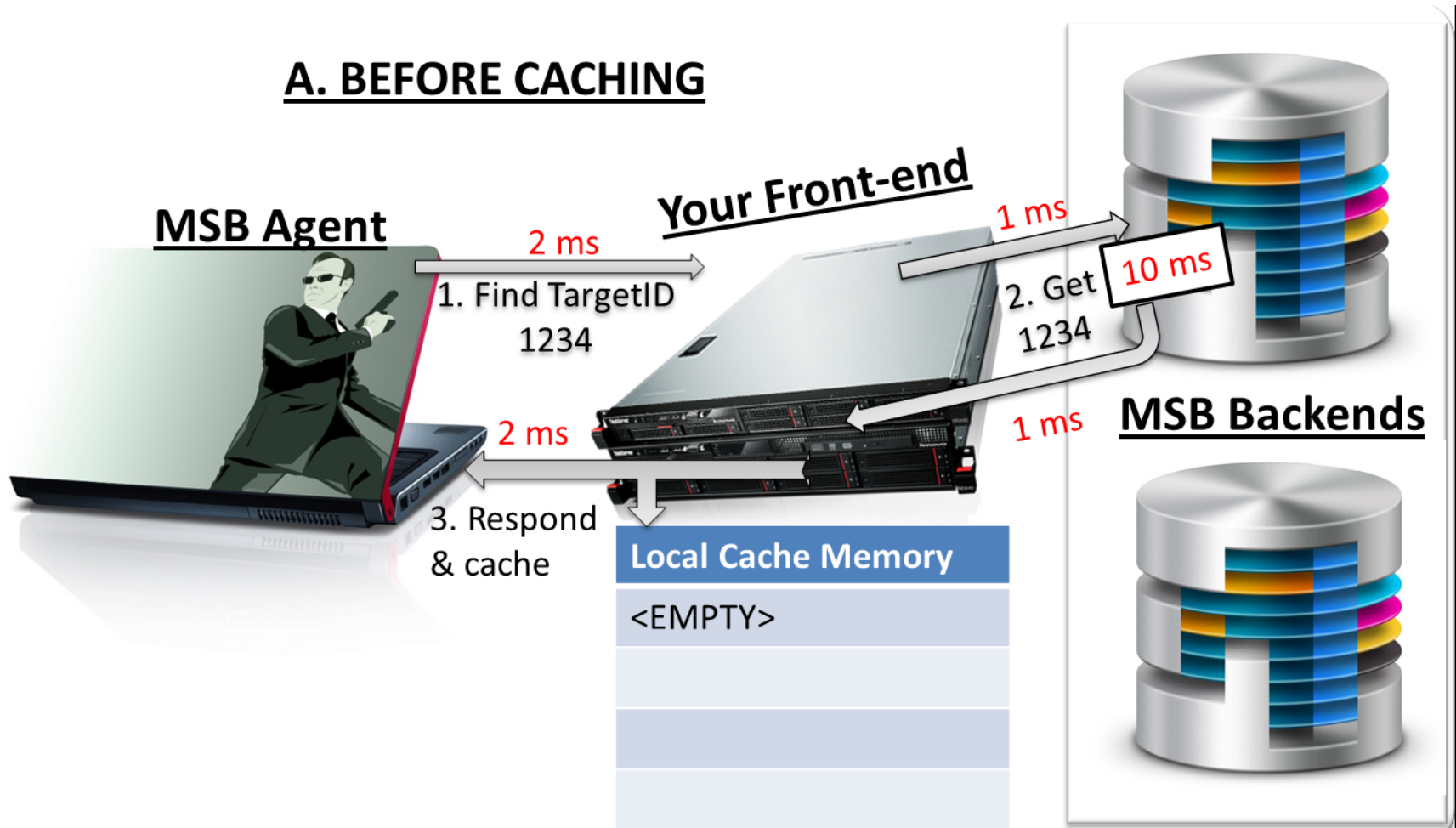
Speed Up Webservice



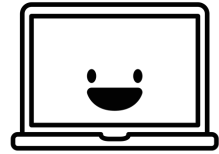
Web Service: Before Caching



A. BEFORE CACHING



Web Service: After Caching



B. AFTER CACHING

MSB Agent



2 ms

1. Find TargetID
1234

Your Front-end



Local Cache Memory

1234:<1234_DATA>

2 ms
2. Respond
from cache



MSB Backends



P2.3 - what you have to do



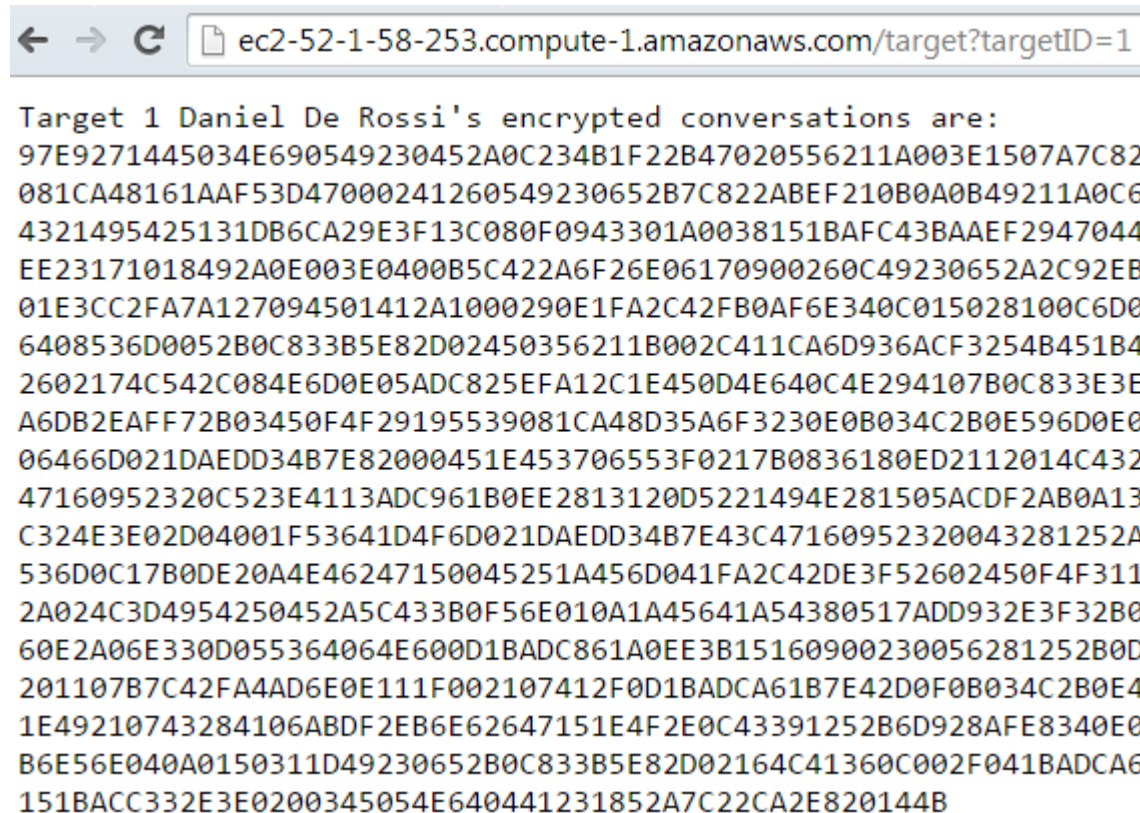
- Speed up the web-service using caching
 - Size of cache on backend and frontend is predetermined
 - Learn the characteristics of the traces
 - Skeleton code provided gives low RPS
 - Trace 1: Skeleton code RPS ~ 65
 - Target RPS ~ 1300

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:



The screenshot shows a web browser window with the address bar containing the URL `ec2-52-1-58-253.compute-1.amazonaws.com/target?targetID=1`. The main content area displays a response consisting of a title and a long block of hexadecimal data.

```
Target 1 Daniel De Rossi's encrypted conversations are:  
97E9271445034E690549230452A0C234B1F22B47020556211A003E1507A7C82  
081CA48161AAF53D47000241260549230652B7C822ABEF210B0A0B49211A0C6  
4321495425131DB6CA29E3F13C080F0943301A0038151BAFC43BAAEF2947044  
EE23171018492A0E003E0400B5C422A6F26E06170900260C49230652A2C92EB  
01E3CC2FA7A127094501412A1000290E1FA2C42FB0AF6E340C015028100C6D0  
6408536D0052B0C833B5E82D02450356211B002C411CA6D936ACF3254B451B4  
2602174C542C084E6D0E05ADC825EFA12C1E450D4E640C4E294107B0C833E3E  
A6DB2EAF72B03450F4F29195539081CA48D35A6F3230E0B034C2B0E596D0E0  
06466D021DAEDD34B7E82000451E453706553F0217B0836180ED2112014C432  
47160952320C523E4113ADC961B0EE2813120D5221494E281505ACDF2AB0A13  
C324E3E02D04001F53641D4F6D021DAEDD34B7E43C47160952320043281252A  
536D0C17B0DE20A4E46247150045251A456D041FA2C42DE3F52602450F4F311  
2A024C3D4954250452A5C433B0F56E010A1A45641A54380517ADD932E3F32B0  
60E2A06E330D055364064E600D1BADDC861A0EE3B15160900230056281252B0D  
201107B7C42FA4AD6E0E111F002107412F0D1BADCA61B7E42D0F0B034C2B0E4  
1E49210743284106ABDF2EB6E62647151E4F2E0C43391252B6D928AFE8340E0  
B6E56E040A0150311D49230652B0C833B5E82D02164C41360C002F041BADCA6  
151BACC332E3E0200345054E640441231852A7C22CA2E820144B
```

Querying multiple records from the DCI

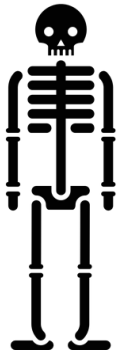
URL: <DCI-DNS>/range?
start_range=1&end_range=3

Response:

```
← → ↻ ec2-52-1-58-253.compute-1.amazonaws.com/range?start_range=1&end_range=3

Target 1 Daniel De Rossi's encrypted conversations are:
97E9271445034E690549230452A0C234B1F22B47020556211A003E1507A7C82FB7F26E060E
58D02AFEE3B03452F4F29195539081CA48161AAF53D47000241260549230652B7C822ABEF:
0E11A8DE6DE3E020034504412A0D53600E1CE3C839B3E43C0E00024321495425131DB6CA2:
B4C240252A0C12EB6E56E4F2401413E064E6D3617A18D12A6F3380E0609536D47000E0D1D:
00260C49230652A2C92EB3F52B03451B49200C4C344113A0DF2EB0F26E06451A413600453:
94501412A1000290E1FA2C42FB0AF6E340C015028100C6D021EACD825E3E2210A1519542D:
F13B130C02476408536D0052B0C833B5E82D02450356211B002C411CA6D936ACF3254B451:
F22A6F26E06170900360C4E390416EF8D33A2F52602174C542C084E6D0E05ADC825EFA12C:
0BED8D02AFEE3B03450F4F29195539081CA48D28B0A12F47170943210754211852A6DB2EAF:
96D0E00E3C024B7E03E0F0A1E00260853280552ACC361B6F5270B0C185964084E294111AC:
3706553F0217B0836180ED2112014C432B045038151BADCA61AAEF3808091A45374944281:
60952320C523E4113ADC961B0EE2813120D5221494E281505ACDF2AB0A13A0F04180025054:
2F00004C412A0D00220F1EAAC324E3E02D04001F53641D4F6D021DAEDD34B7E43C4716095:
EB6A12F15004C412605456D151DE3C924A0F33717114C542C00536D0C17B0DE20A4E46247:
B7CC27A5A12808174C41645C056D031DADD832E3EE20471104456408533E0815ADC024ADF:
80517ADD932E3F32B04000556214954250801E3CF2EADF43D49452F4F2A0E522C1507AFCC:
064E600D1BADDC861A0EE3B15160900230056281252B0D934A7E42013164C412A494F3B040:
944642A4F201107B7C42FA4AD6E0E111F002107412F0D1BADCA61B7E42D0F0B034C2B0E49:
47040244640141230501EEC22FE3E43617001E49210743284106ABDF2EB6E62647151E4F2:
0ED2112014C08050441370E1CE3FA24A1A11D02171A49270C53644F5280C12EB6E56E040A:
CA61A2E52117110944641E4929041EBA8D20A0F32114164C41641F413F0817B7D461ACE76:
852A7C22CA2E820144B;Target 2 Abner Felipe Souza De's encrypted conversations
150A0716524F284800001E000030000008124564024937091E451A1014250700110100270:
352632A0A190D50264F3E1F000E084E23490028181E450C0A00230E070B150032000F011E:
0A0F020349003201115A09412A01536C0303450C1C11241007001C4323451801020A55340:
20949304F16160E5520450800010C1F060A4116070C4521453413050A1516097D4F36160E:
4F171F084E23454125031D110C0041360B0A001E5966040F1B1F1653730E550C00522D0054
```

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!

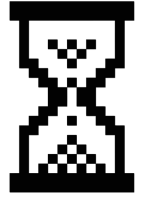
```
/*
 * retrieveDetails - you have to modify this function to achieve a higher RPS value
 * Input: the targetID
 * Returns: The result from querying the database instance
 */
private String retrieveDetails(String targetID) {
    try{
        return sendRequest(generateURL(0, targetID));
    } catch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

Resources



1. Brewer, Eric A. "Lessons from giant-scale services." *Internet Computing, IEEE* 5.4 (2001): 46-55.
2. Sivasubramanian, Swaminathan, et al. "Analysis of caching and replication strategies for web applications." *Internet Computing, IEEE* 11.1 (2007): 60-66.
3. Karger, David, et al. "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web." *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. ACM, 1997.
4. Fan, Bin, et al. "Small cache, big effect: Provable load balancing for randomly partitioned cluster services." *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011.

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

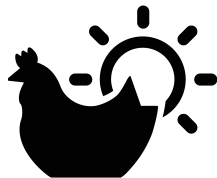


- Caches will be effective if the memory accesses exhibit good locality of references
- Temporal locality - resources accessed in close points in time.
- Spatial Locality - likelihood that a resource will be accessed that is near a resource that was just accessed

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



Violation	Penalty of the project grade
Spending more than \$10 for this project phase	-10%
Spending more than \$20 for this project phase	-100%
Failing to tag all your resources for this project	-10%
Submitting your AWS credentials in your code for grading	-10%
Using instances other than the ones specified	-100%
Caching more than 1000 records in the front end	-100%

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

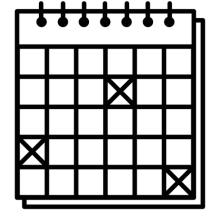


Form your 15619Project teams and submit your team info to:

<http://bit.ly/1EFAsdO>

- Once submitted, teams are final and binding

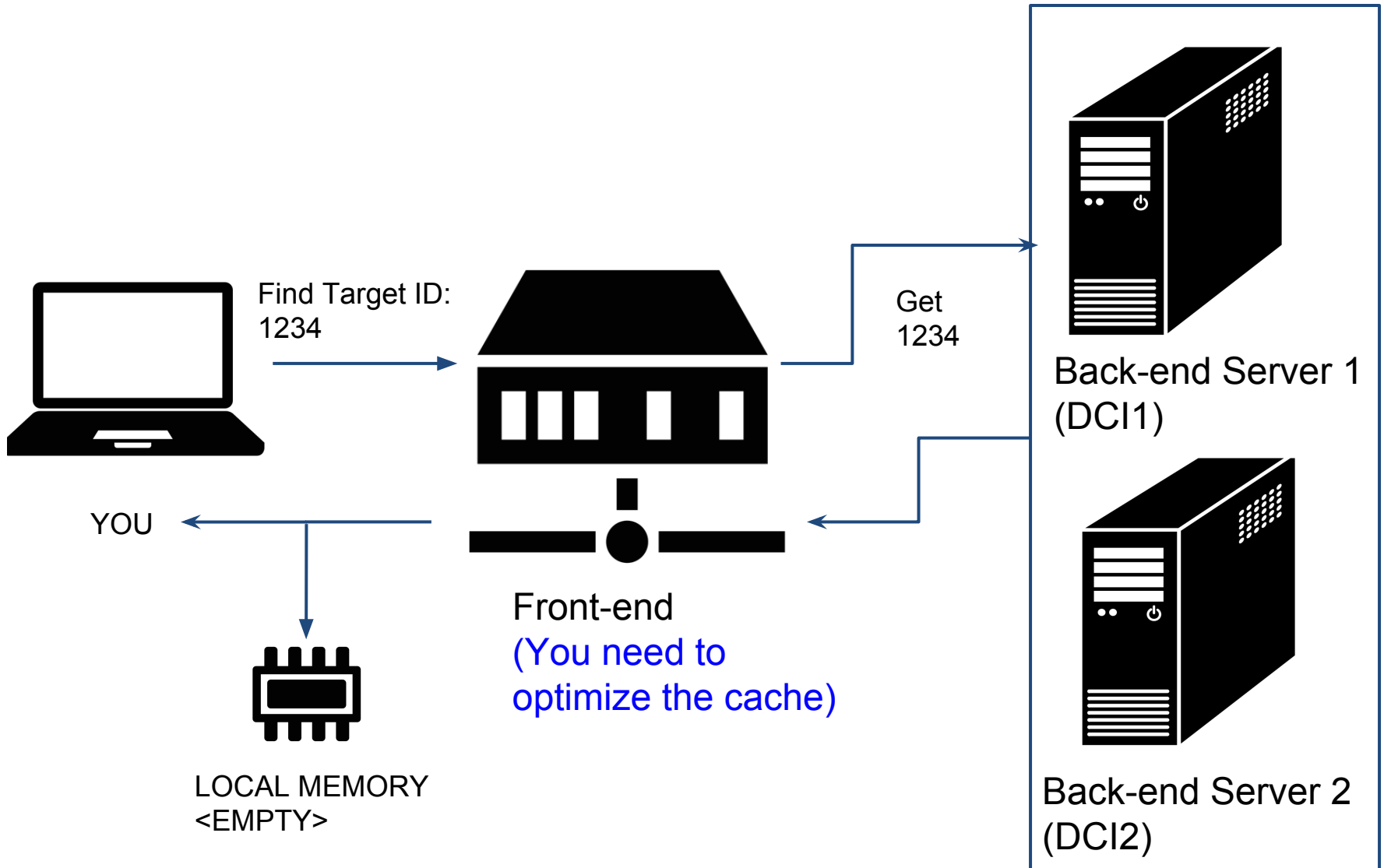
15619 Project Time Table



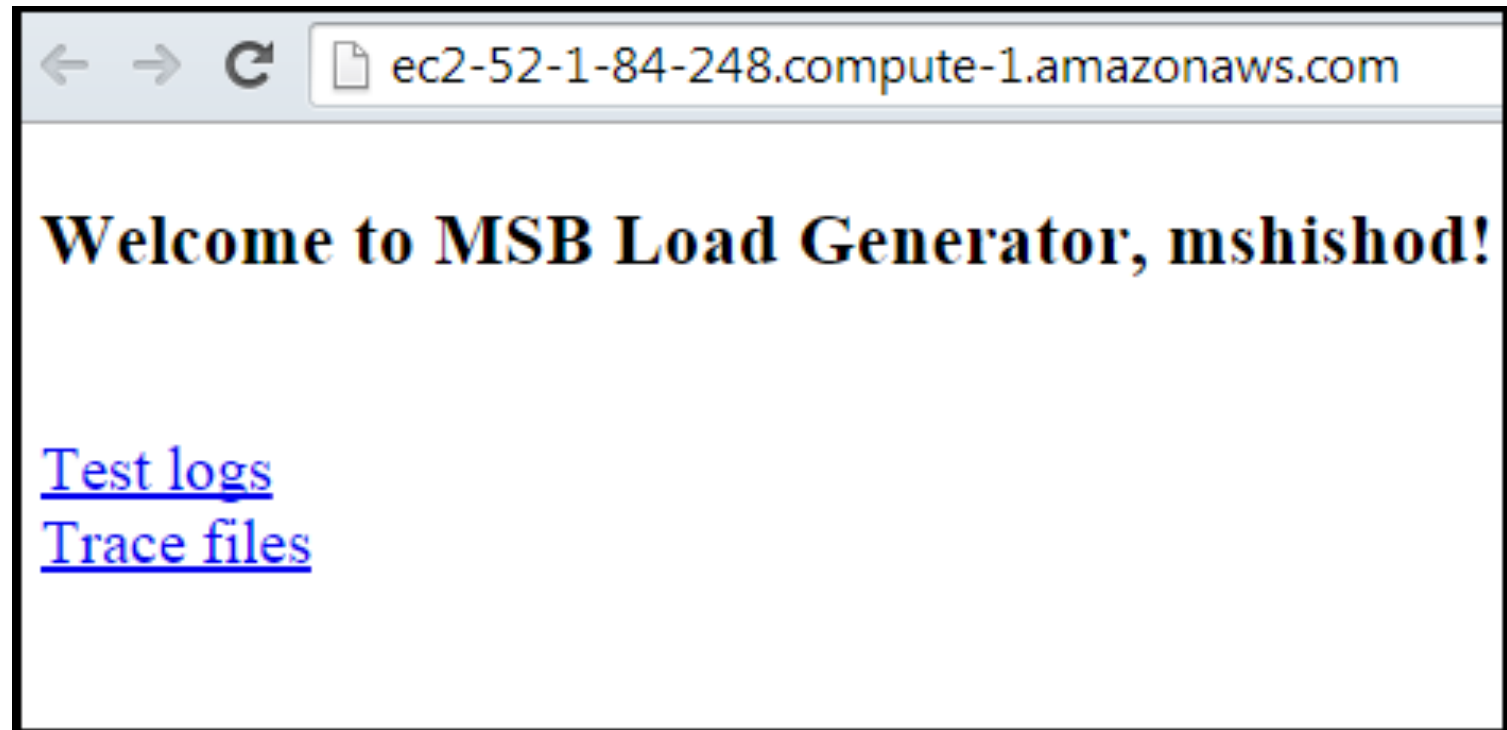
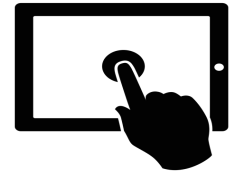
Phase (and query due)	Start	End
Phase 1 Part 1 <ul style="list-style-type: none"> Q1 (due), Q2 (not yet due) 	Thursday 2/26/2015 00:00:01 EST	Wednesday 3/4/2015 23:59:59 EST
Phase 1 Part 2 <ul style="list-style-type: none"> Q1, Q2 (due) 	Thursday 3/26/2015 00:00:01 EST	Wednesday 3/18/2015 23:59:59 <u>EDT</u>
Phase 2 <ul style="list-style-type: none"> Q1, Q2, Q3, Q4 	Thursday 3/19/2015 00:00:01 <u>EDT</u>	Wednesday 3/25/2015 16:59:59 EDT
Phase 2 Live Test <ul style="list-style-type: none"> Q1, Q2, Q3, Q4 	Wednesday 3/25/2015 18:00:01 EDT	Wednesday 3/25/2015 23:59:59 EDT
Phase 3 <ul style="list-style-type: none"> Q1, Q2, Q3, Q4, Q5, Q6 	Thursday 3/26/2015 00:00:01 EDT	Wednesday 4/8/2015 18:59:59 EDT
Phase 3 Live Test <ul style="list-style-type: none"> Q1, Q2, Q3, Q4, Q5, Q6 	Wednesday 4/8/2015 20:00:01 EDT	Wednesday 4/8/2015 23:59:59 EDT

There will also be a report due at the end of each phase, where you are expected to discuss optimizations

P2.3 - This Week



Project 2.3: Load Generator UI



Project 2.3: Data Center UI

