

15-319 / 15-619

Cloud Computing

Recitation 6

February 17th, 2016

Overview

- **Administrative issues**
Office Hours, Piazza guidelines
- **Last week's reflection**
Project 2.2, OLI unit 3 module 7, 8 and 9
- **This week's schedule**
 - Project 2.3 - Sunday, February 21st
 - Unit 3 - Modules 10, 11 and 12
 - Quiz 5 - Friday, February 19th
- **Demo**

Project 2.2



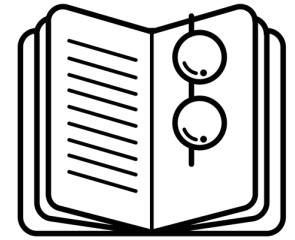
- Manual Grading: 20% for code
 - Always make sure that your code is readable
 - Follow style presented in Recitation 2

Project Grading Penalties

Besides the penalties mentioned in recitation and/or on Piazza, penalties accrue for the following:

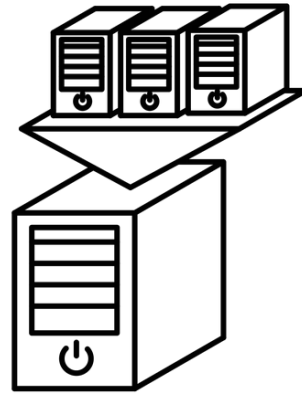
Violation	Penalty of the project grade
Spending more than \$5 for this project phase on AWS	-10%
Spending more than \$10 for this project phase on AWS	-100%
Failing to tag all your AWS resources in either parts (EC2 instances, etc) for this project	-10%
Submitting your AWS credentials in your code for grading	-100%
Submitting your Azure credentials in your code for grading	-100%
Submitting the Azure part with AWS instances or the AWS part with Azure VMs.	-100%
Attempting to hack/tamper the autograder in any way	-100%
Using virtual machines other than Standard_A1(DC) and Standard_D1(LG and LB) in the Azure part	-100%

This Week: Content



- UNIT 3: Virtualizing Resources for the Cloud
 - Module 7: Introduction and Motivation
 - Module 8: Virtualization
 - Module 9: Resource Virtualization - CPU
 - Module 10: Resource Virtualization - Memory
 - Module 11: Resource Virtualization – I/O
 - Module 12: Case Study
 - Module 13: Storage and Network Virtualization

OLI, Unit 3: Modules 10, 11, 12



- Understand two-level page mappings from virtual to real to physical
- Learn how memory is overcommitted and reclaimed using ballooning
- Study how I/O requests are intercepted at different interfaces
- Map these concepts into your practical exploration with AWS

This Week: Project



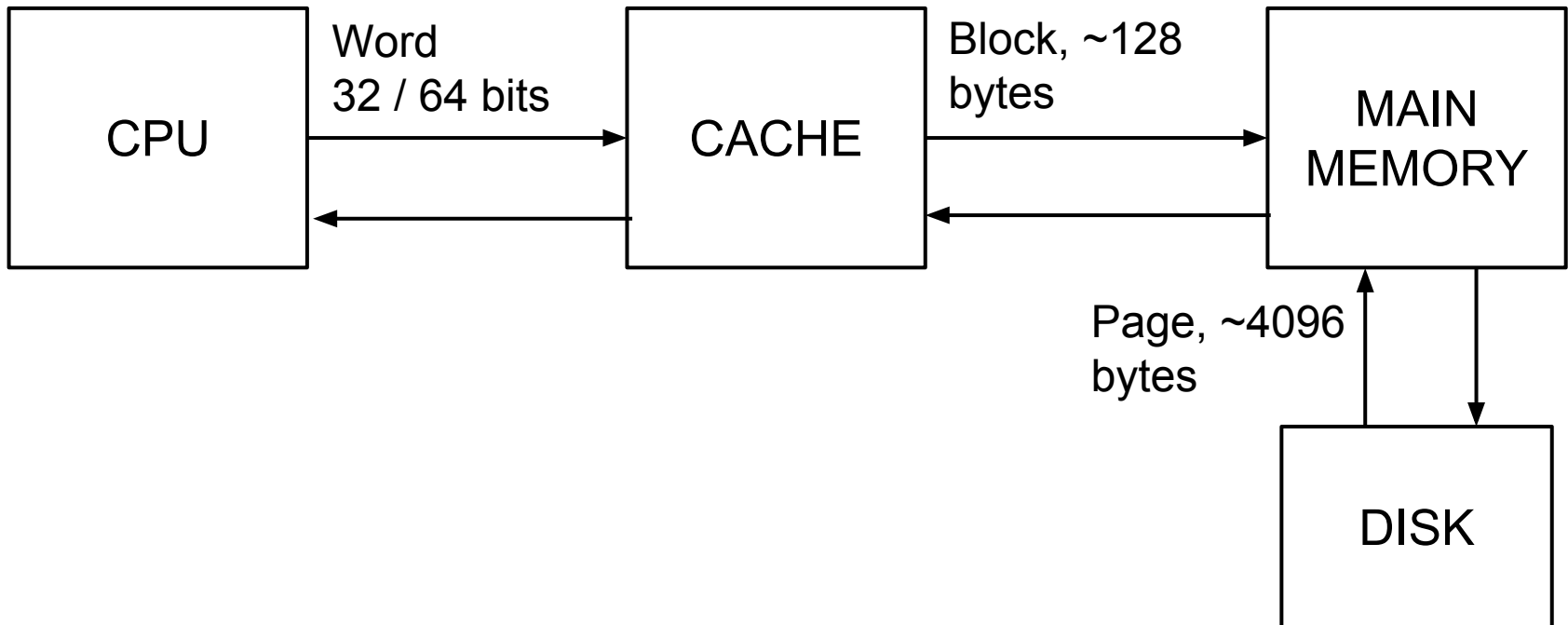
- P2.1: Horizontal Scaling and Advanced Resource Scaling
 - MSB Recruitment Exam
- P2.2: The Internals and Strategies of a Load Balancer
 - Junior System Architect at the MSB
- P2.3: Advanced Scaling Concepts: Caching
 - Speed up a web-service using caching

Caching

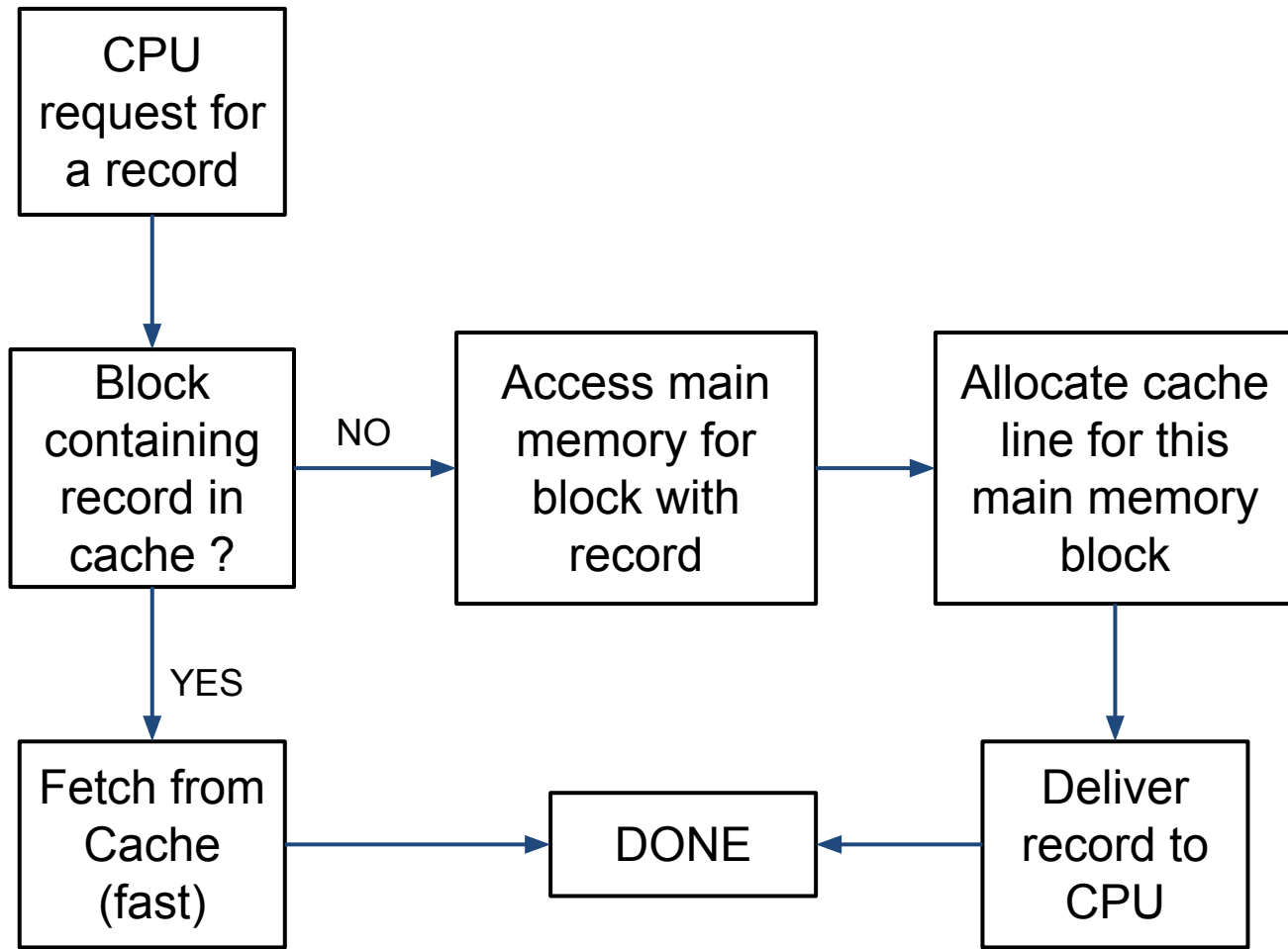
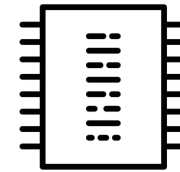
- Technique for hiding latency
- Move data closer (faster) to client
- Leverages hierarchies of access speeds
- Benefits from bias towards locality of access
- Generally a transparent abstraction layer
- Introduces many different inconsistencies
 - Page replacement and invalidation
 - Cache coherence
- Highly beneficial, but needs careful tuning

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

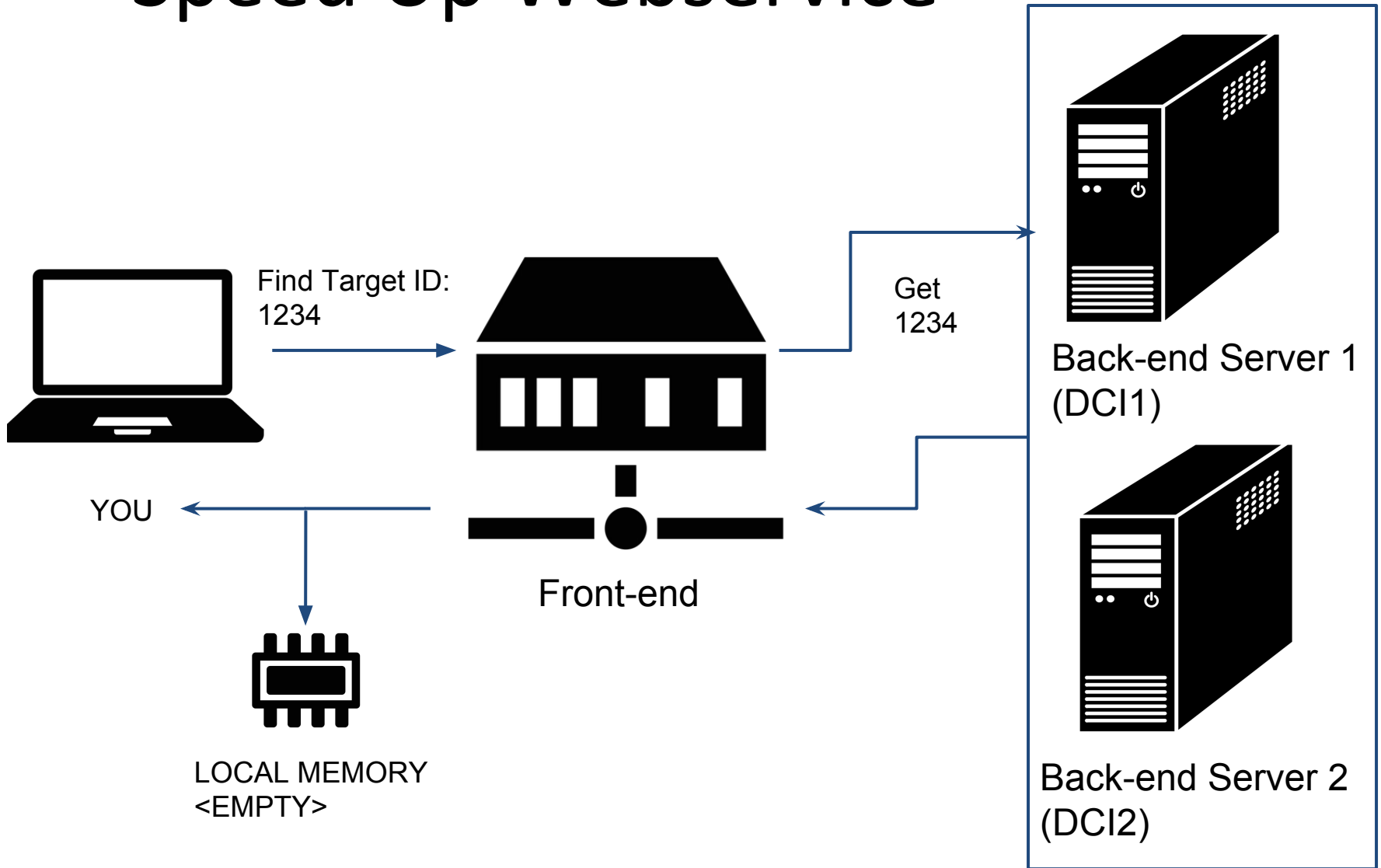


Caching - III

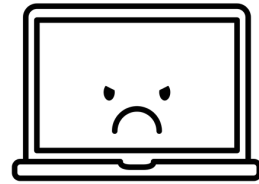
- Exploiting locality
 - Temporal locality
 - Spatial locality
- Things to consider in caching
 - Which data to (pre-?)fetch?
 - Which block to invalidate during writing?
 - Eviction policy, when you run out of space?

P2.3 - This Week

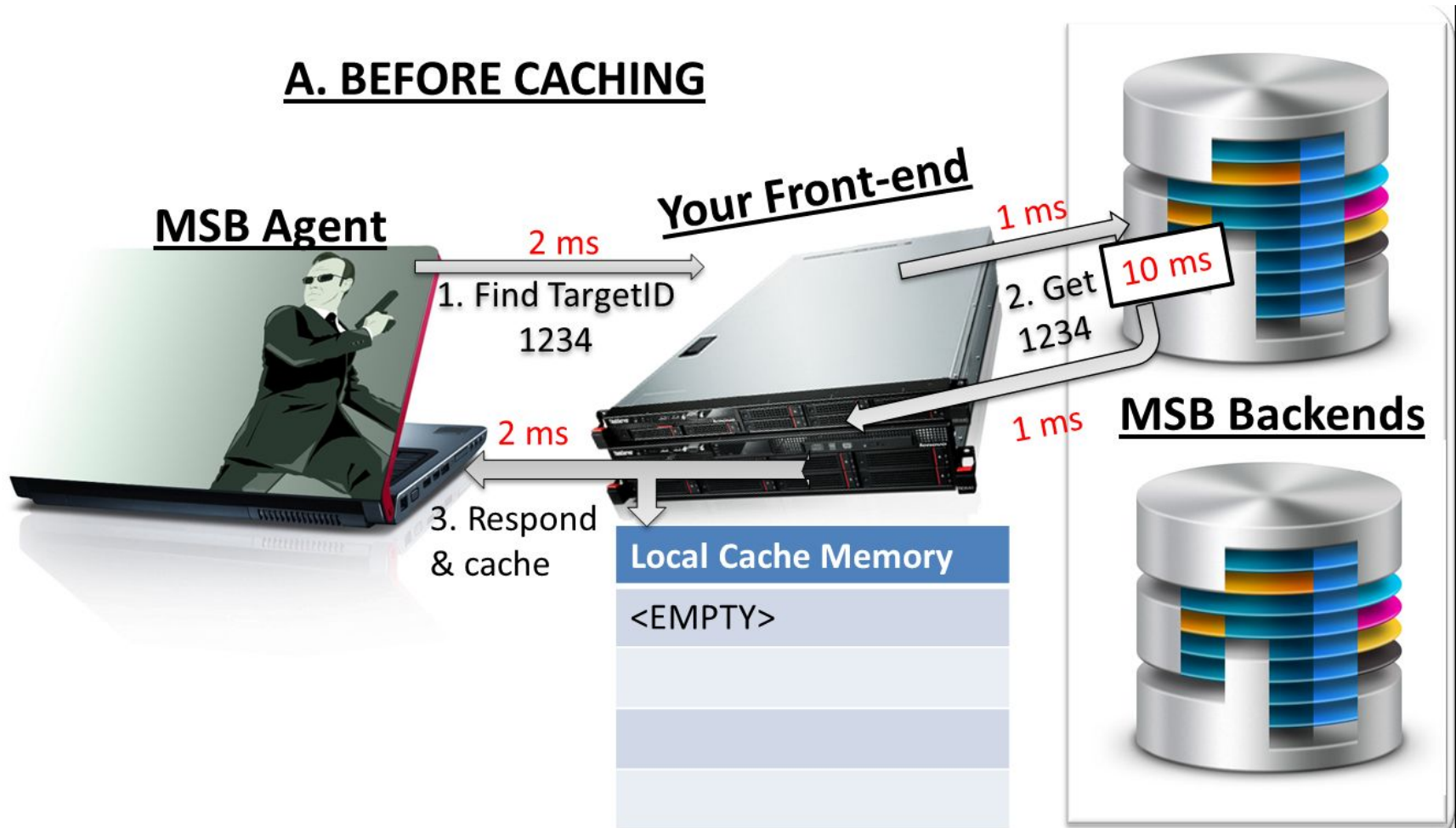
Speed Up Webservice



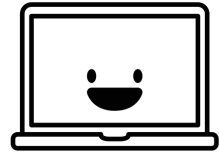
Web Service: Before Caching



A. BEFORE CACHING



Web Service: After Caching



B. AFTER CACHING

MSB Agent



Your Front-end



MSB Backends



2 ms

1. Find TargetID
1234

2 ms

2. Respond
from cache

Local Cache Memory

1234:<1234_DATA>

P2.3 - what you have to do



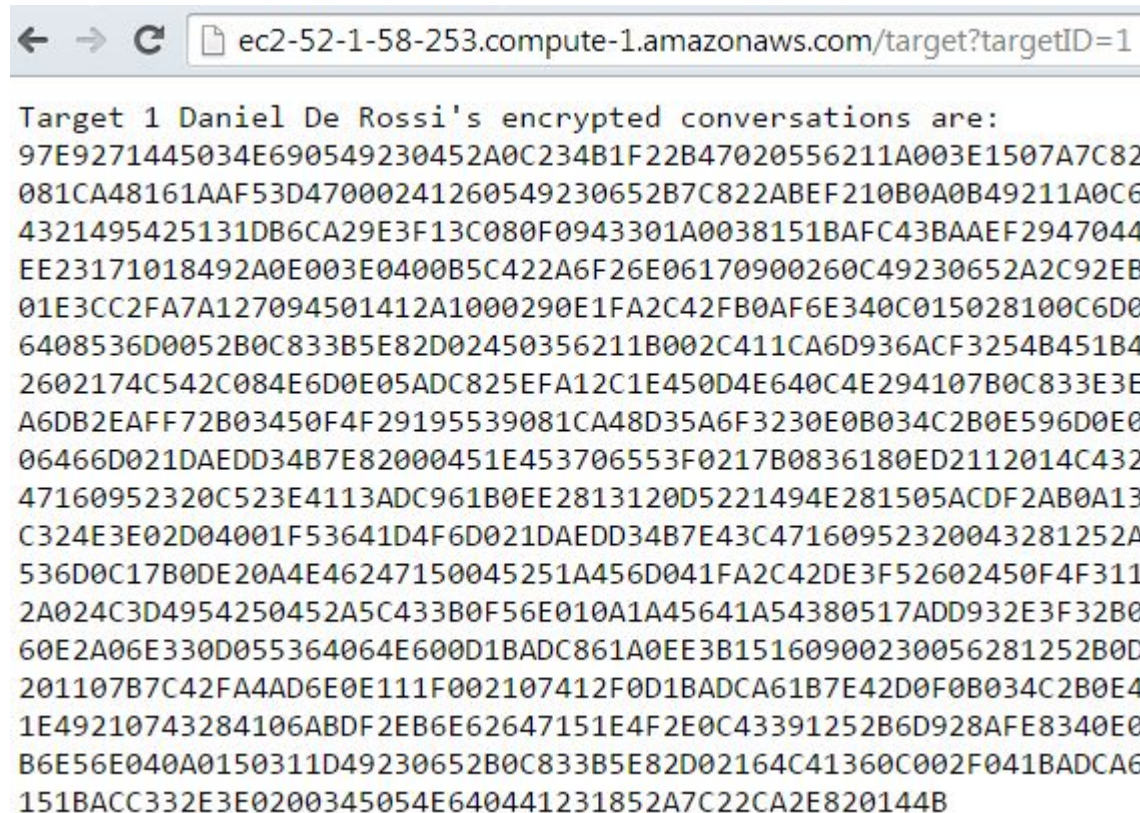
- Serve more web requests in the same time period using a cache to save on the number of hops
 - Size of cache on backend and frontend is predetermined
 - Learn the characteristics of the traces
 - Skeleton code provided gives low RPS
 - Trace 1: Skeleton code RPS ~ 25
- Target RPS ~ 145

Note: You are allowed to 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:  
97E9271445034E690549230452A0C234B1F22B47020556211A003E1507A7C82  
081CA48161AAF53D47000241260549230652B7C822ABEF210B0A0B49211A0C6  
4321495425131DB6CA29E3F13C080F0943301A0038151BAFC43BAAEF2947044  
EE23171018492A0E003E0400B5C422A6F26E06170900260C49230652A2C92EB  
01E3CC2FA7A127094501412A1000290E1FA2C42FB0AF6E340C015028100C6D0  
6408536D0052B0C833B5E82D02450356211B002C411CA6D936ACF3254B451B4  
2602174C542C084E6D0E05ADC825EFA12C1E450D4E640C4E294107B0C833E3E  
A6DB2EAF72B03450F4F29195539081CA48D35A6F3230E0B034C2B0E596D0E0  
06466D021DAEDD34B7E82000451E453706553F0217B0836180ED2112014C432  
47160952320C523E4113ADC961B0EE2813120D5221494E281505ACDF2AB0A13  
C324E3E02D04001F53641D4F6D021DAEDD34B7E43C47160952320043281252A  
536D0C17B0DE20A4E46247150045251A456D041FA2C42DE3F52602450F4F311  
2A024C3D4954250452A5C433B0F56E010A1A45641A54380517ADD932E3F32B0  
60E2A06E330D055364064E600D1BADDC861A0EE3B15160900230056281252B0D  
201107B7C42FA4AD6E0E111F002107412F0D1BADCA61B7E42D0F0B034C2B0E4  
1E49210743284106ABDF2EB6E62647151E4F2E0C43391252B6D928AFE8340E0  
B6E56E040A0150311D49230652B0C833B5E82D02164C41360C002F041BADCA6  
151BACC332E3E0200345054E640441231852A7C22CA2E820144B
```

Querying multiple records from the DCI

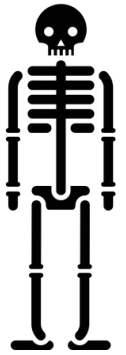
URL: <DCI-DNS>/range?
start_range=1&end_range=3

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

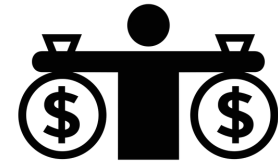


Skeleton code in java (**MSB.java**)

- You can get a full score by optimizing this function.
- Feel free to optimize other parts too!

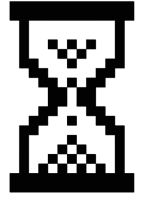
```
/*
 * retrieveDetails - you have to modify this function to achieve a higher RPS value
 * Input: the targetID
 * Returns: The result from querying the database instance
 */
private String retrieveDetails(String targetID) {
    try{
        return sendRequest(generateURL(0, targetID));
    } catch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

Resources



1. Brewer, Eric A. "Lessons from giant-scale services." IC, 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 front-end web-service:

Get record for targetID = 2000

Without the record cached: 0.135s

With the record cached at back-end: 0.030s

With the record cached at front-end: 0.011s

Project 2.3 Hints - 1

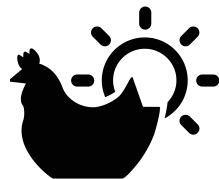


- Caches will be effective if the memory accesses exhibit good locality of references
 - Temporal locality - resources accessed in close points in time.
 - Spatial Locality - likelihood that a resource will be accessed that is near a resource that was just accessed

Project 2.3 Hints - 2



- Read the project description more than once.
- Think about the 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



Project Grading Penalties

Besides the penalties mentioned in recitation and/or on Piazza, penalties accrue for the following:

Violation	Penalty of the project grade
Spending more than \$5 for this project phase on AWS	-10%
Spending more than \$10 for this project phase on AWS	-100%
Failing to tag all your AWS resources in either parts (EC2 instances, etc) for this project	-10%
Submitting your AWS credentials in your code for grading	-100%
Submitting your Azure credentials in your code for grading	-100%
Submitting the Azure part with AWS instances or the AWS part with Azure VMs.	-100%
Attempting to hack/tamper the autograder in any way	-100%
Using virtual machines other than Standard_D1(DC and LG) and Standard_A0(FE) in the Azure part	-100%
Caching more than 1000 records in the front end.	-100%

Upcoming Deadlines



- Project 2.3: Advanced Scaling Concepts - Caching

Due: 02/21/16 11:59PM Pittsburgh

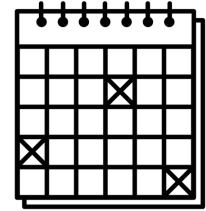


- Quiz 5: Unit 3 - Virtualizing IO, Memory; Cases

Due: 02/19/16 11:59PM Pittsburgh



15619 Project Time Table



Phase (and query due)	Start	Deadline	Code and Report Due
Phase 1 Part 1 • Q1, Q2	Thursday 02/25/2016 00:00:01 EST	Wednesday 03/16/2016 23:59:59 EDT	Thursday 03/17/2016 23:59:59 EDT
Phase 2 • Q1, Q2, Q3	Thursday 03/17/2016 00:00:01 EDT	Wednesday 03/30/2016 15:59:59 EDT	
Phase 2 Live Test (Hbase/MySQL) • Q1, Q2, Q3	Wednesday 03/30/2016 18:00:01 EDT	Wednesday 03/30/2016 23:59:59 EDT	Thursday 03/31/2016 23:59:59 EDT
Phase 3 • Q1, Q2, Q3, Q4	Thursday 03/31/2016 00:00:01 EDT	Wednesday 04/13/2016 15:59:59 EDT	
Phase 3 Live Test • Q1, Q2, Q3, Q4	Wednesday 04/13/2016 18:00:01 EDT	Wednesday 04/13/2016 23:59:59 EDT	Thursday 04/13/2016 23:59:59 EDT

Note:

- There will also be a report due at the end of each phase, where you are expected to discuss optimizations
- **WARNING: Check your AWS instance limits on the new account (should be > 10 instances)**

Demo