15-319 / 15-619 Cloud Computing

Recitation 9

October 27th and 29th, 2015

Overview

- **Administrative issues**
  Office Hours, Piazza guidelines
- **Last week's reflection**
  Project 3.2, OLI Unit 4, Module 14, Quiz 7
- **This week's schedule**
  - 15619 Project - Query 1 & 2 - October 28th
  - Quiz 8 - October 30th (Unit 4, Module 15)
  - Project 3.3 - November 1st

Announcements

- Monitor AWS expenses regularly and tag all resource
  - Check your bill (Cost Explorer > filter by tags).
- Piazza Guidelines
  - Please tag your questions appropriately
  - Search for an existing answer first
- Provide clean, modular and well documented code
  - **Large** penalties for not doing so.
- Utilize Office Hours
  - We are here to help (but not to give solutions)
- Use the team AWS account and tag the 15619Project resources carefully. Otherwise, you might risk having them charged to your weekly projects.

Last Week : A Reflection

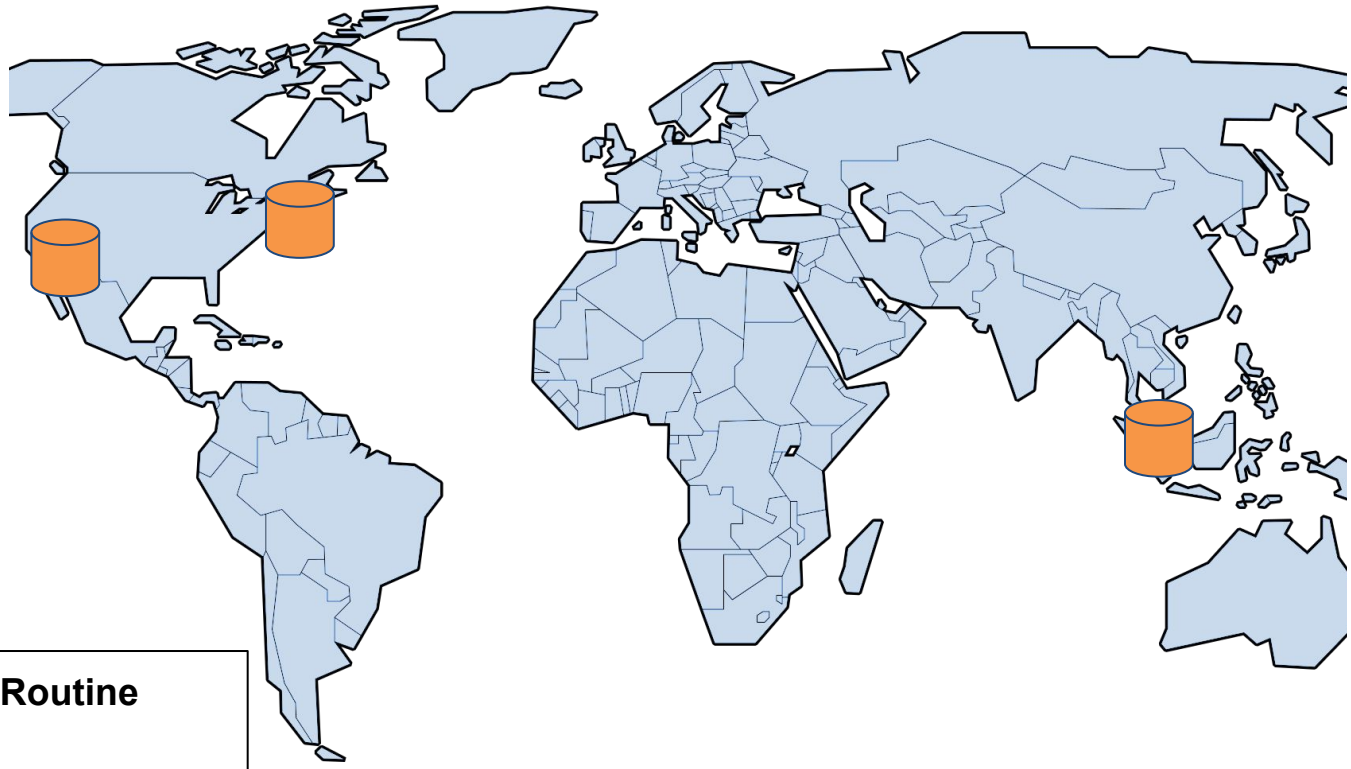
- Content, Unit 4 - Module 14:
  - Cloud Storage - Big Picture
  - Quiz 7 completed

- P3.2: You explored distributed databases:
  - Implemented a coordinator
    - Sharding
    - Replication

Project 3 Weekly Modules

- P3.1: Files, SQL and NoSQL
- P3.2: Sharding and Replication
- **P3.3: Consistency**
- P3.4: Social network and heterogeneous back end storage
- P3.5: Data warehousing and OLAP

P3.3: Motivation - Consistency



Withdrawal Routine

```
if(amt< balance):
  bal = bal - amt
  return amt
else:
  return 0
```

Account	Balance
xxxxx-4437	$100

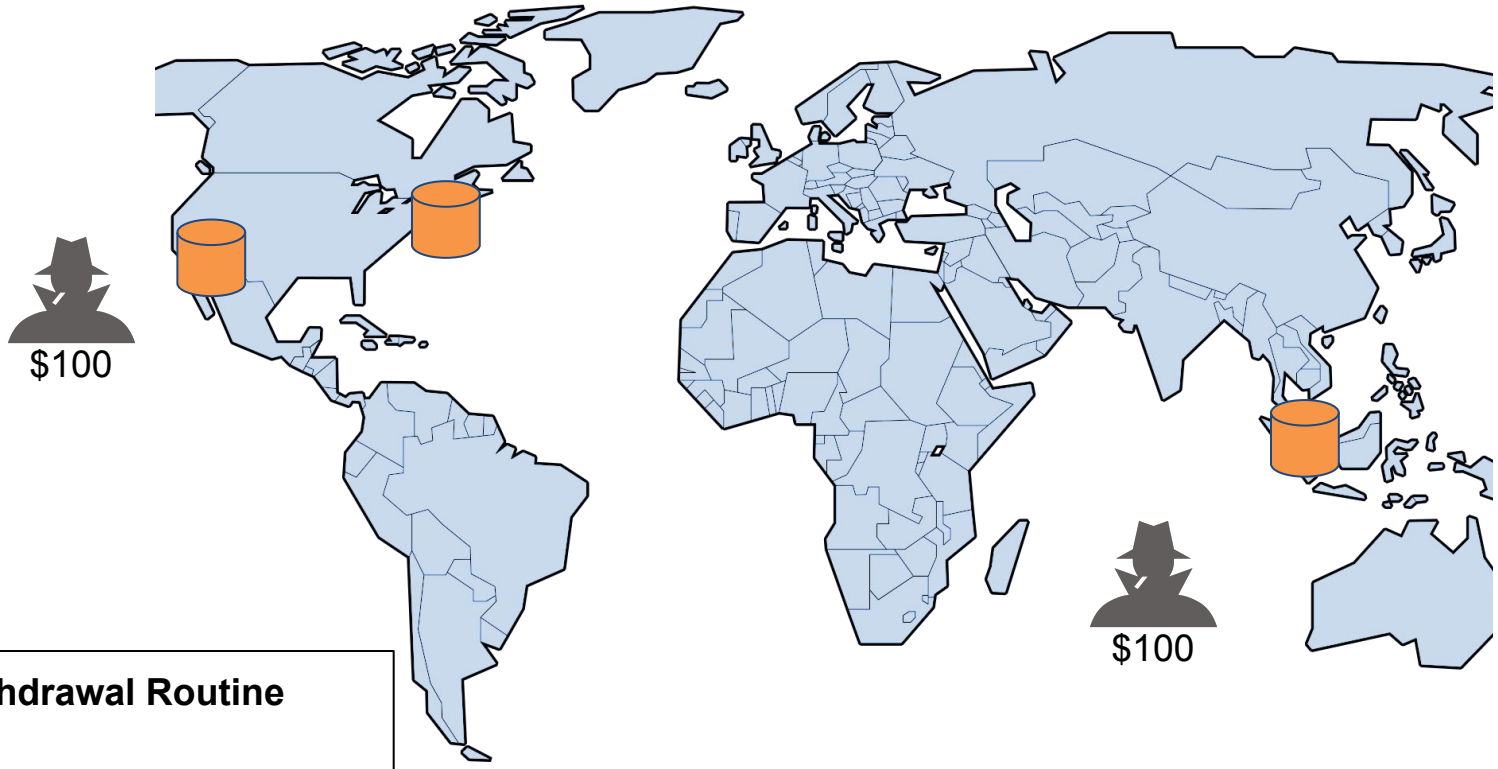
P3.3: Motivation - Consistency

Withdrawal Routine

```
if(amt< balance):
  bal = bal - amt
  return amt
else:
  return 0
```

Account	Balance
xxxxx-4437	$100

P3.3: Motivation - Consistency



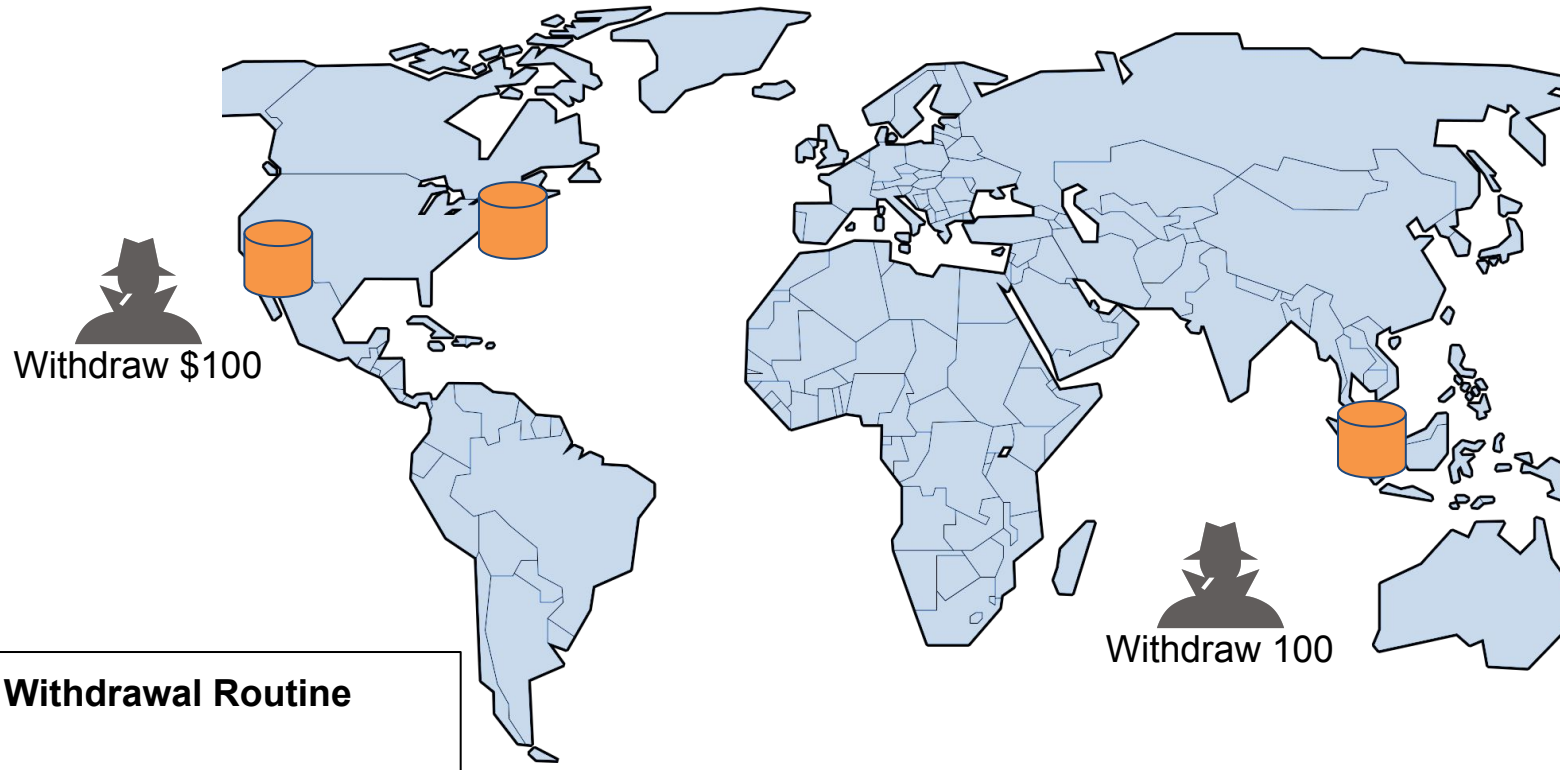
Withdrawal Routine

```
if(amt< balance):
  bal = bal - amt
  return amt
else:
  return 0
```

Account	Balance
xxxxx-4437	**$0**

Bank lost $100

P3.3: Motivation - Consistency



Withdrawal Routine

```
lock(balance)
if(amt< balance):
  bal = bal - amt
  return amt
else:
  return 0
unlock(balance)
```

Account	Balance
xxxxx-4437	**$100**

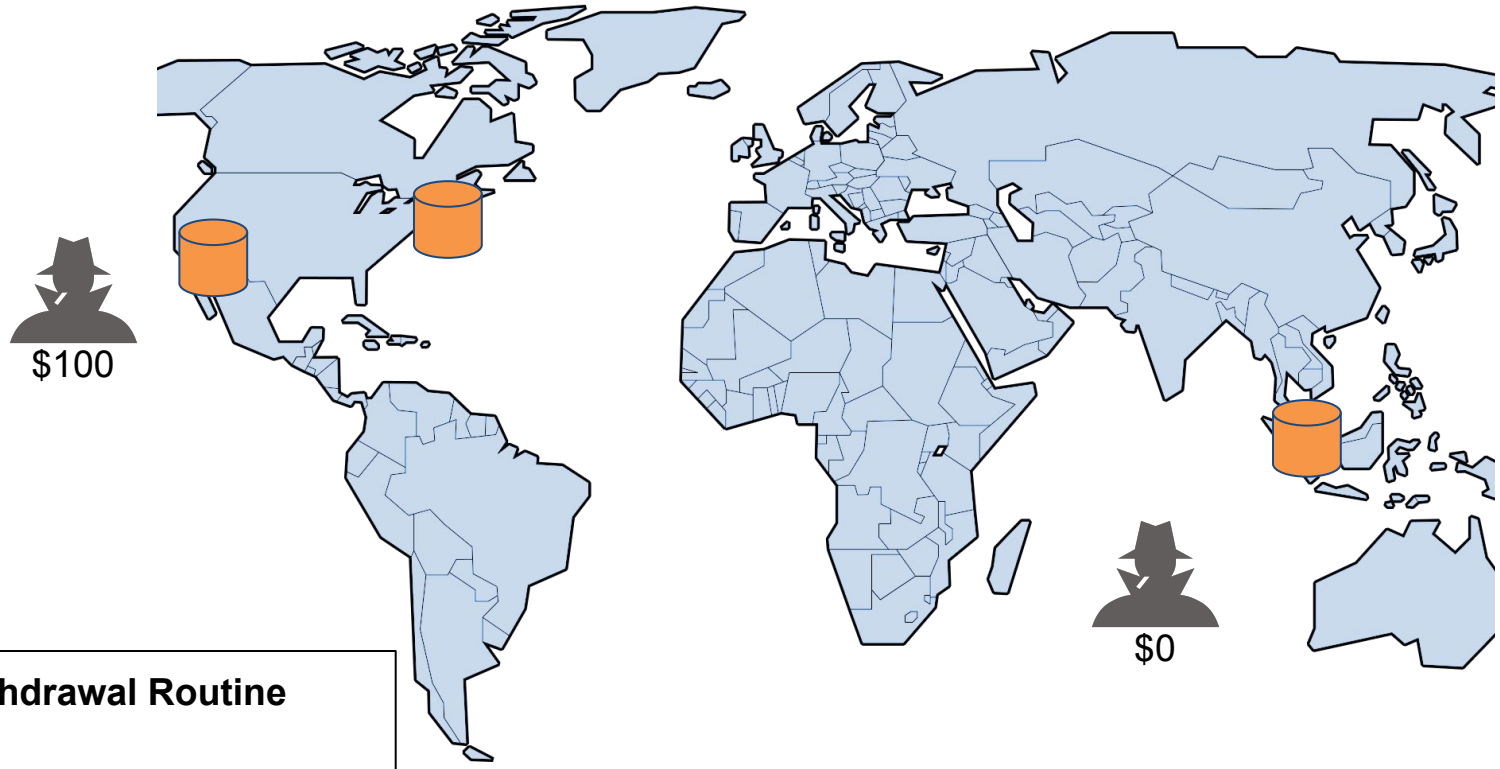
P3.3: Motivation - Consistency

Withdrawal Routine

```
lock(balance)
if(amt< balance):
  bal = bal - amt
  return amt
else:
  return 0
unlock(balance)
```

Account	Balance
xxxxx-4437	**$100**

P3.3: Motivation - Consistency

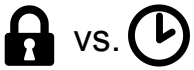


Withdrawal Routine

```
lock(balance)
if(amt< balance):
  bal = bal - amt
  return amt
else:
  return 0
unlock(balance)
```

Account	Balance
xxxxx-4437	**$0**

P3.3: Consistency Models

Tradeoff: 

- Strict
- Strong
- Sequential
- Causal
- Eventual

P3.3: Strong Consistency

- Every operation receives a global timestamp order
  - Typically the order in which they arrive at the coordinator
- Operations must be ordered according to timestamps
- At any given point of time, all clients should read the same data from any datacenter replica.

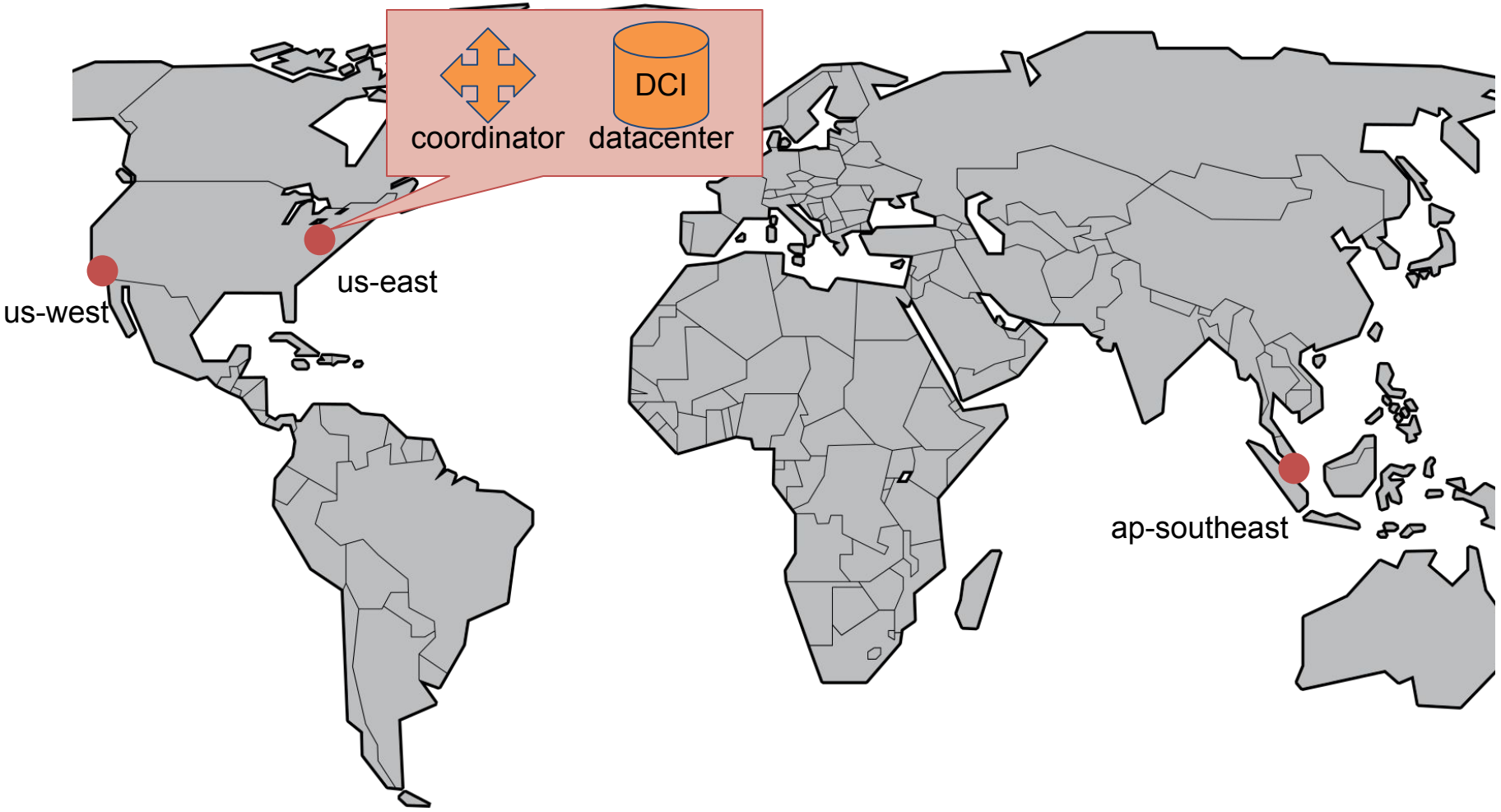
P3.3: Causal Consistency

- Causally-related operations must be ordered correctly
  - All other operations can be performed in any order
- Provides better performance than strong consistency

P3.3: Eventual Consistency

- Writes are performed in the order they are received at each replica
  - Operations may not be blocked for replica consensus
- Clients that request data may receive multiple versions of the data, or stale data
  - Left to the application to resolve

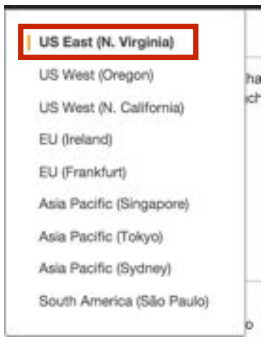
P3.3: Architecture



P3.3: Your Task

- Launch Coordinators and DCIs
  - All in **us-east**, we simulate global latencies
- Implement the Coordinators
  - Strong Consistency
  - Causal Consistency
  - Eventual Consistency

P3.3: Hints

- Launch a total of 7 machines (3 data centers, 3 coordinators and 1 client)
- 

  All machines should be launched in US East region.

  The “US East” here has nothing to do with the simulated location of datacenters and coordinators in the project.

US-EAST DATACENTER (*KeyValueStore.java*)

US-WEST DATACENTER (*KeyValueStore.java*)

SINGAPORE DATACENTER (*KeyValueStore.java*)

US-EAST COORDINATOR (*Coordinator.java*)

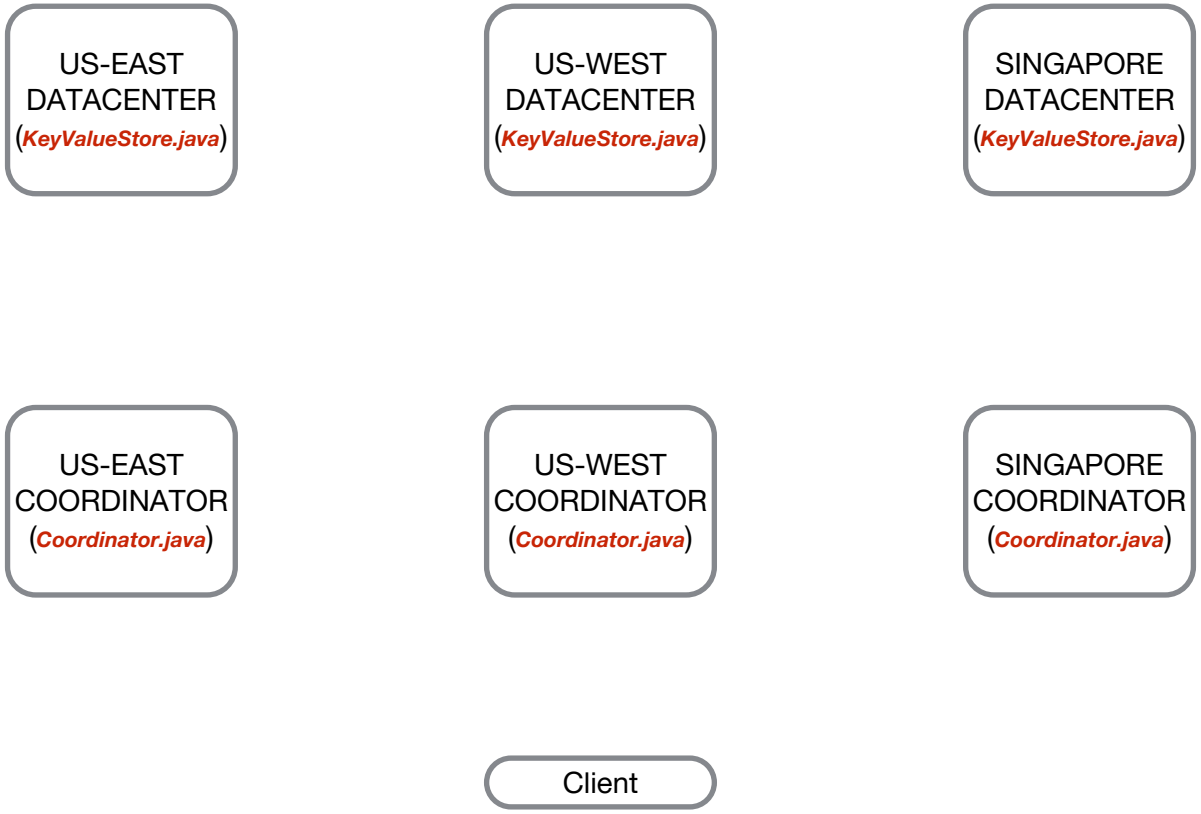
US-WEST COORDINATOR (*Coordinator.java*)

SINGAPORE COORDINATOR (*Coordinator.java*)

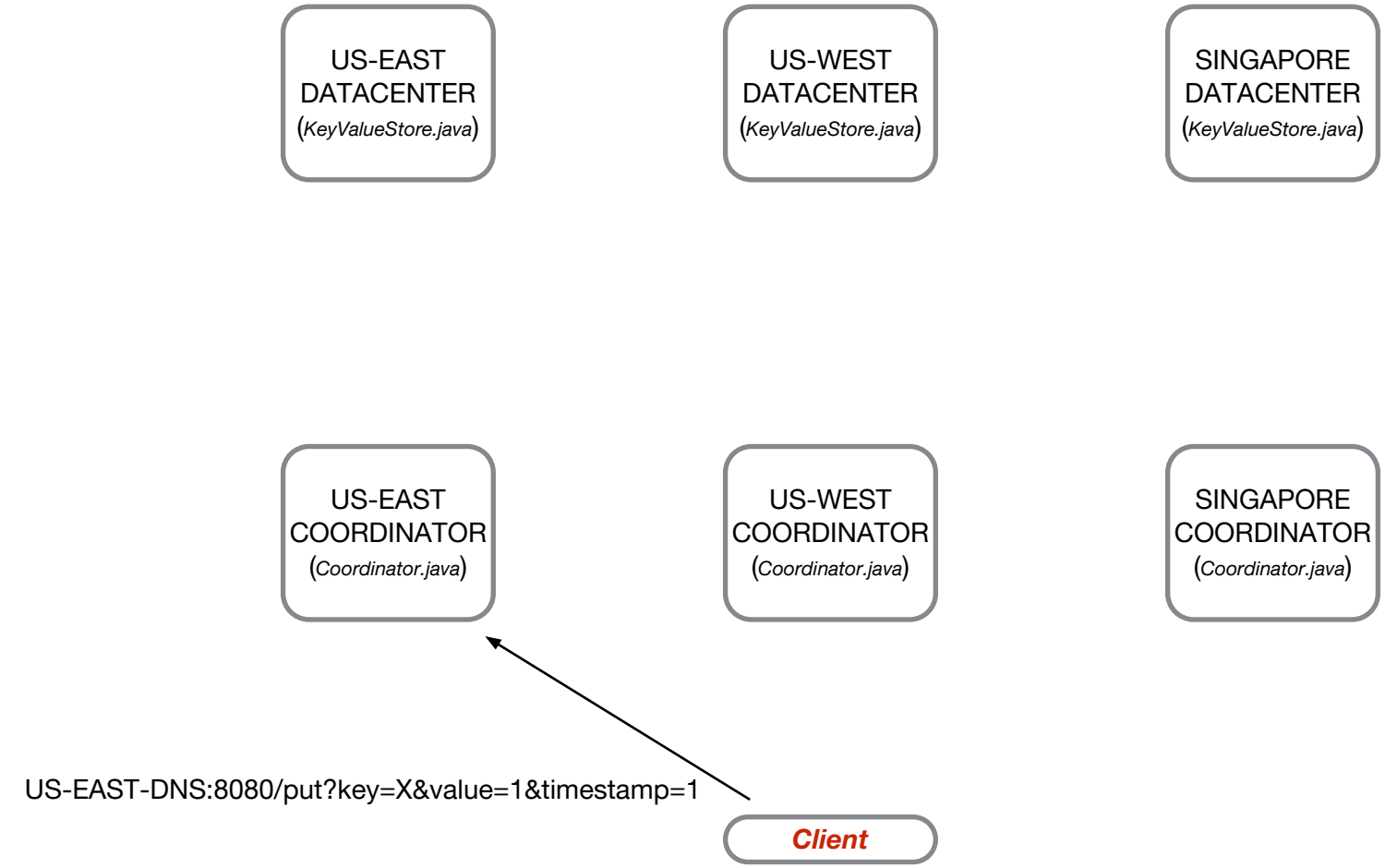
Client

P3.3 TODO:

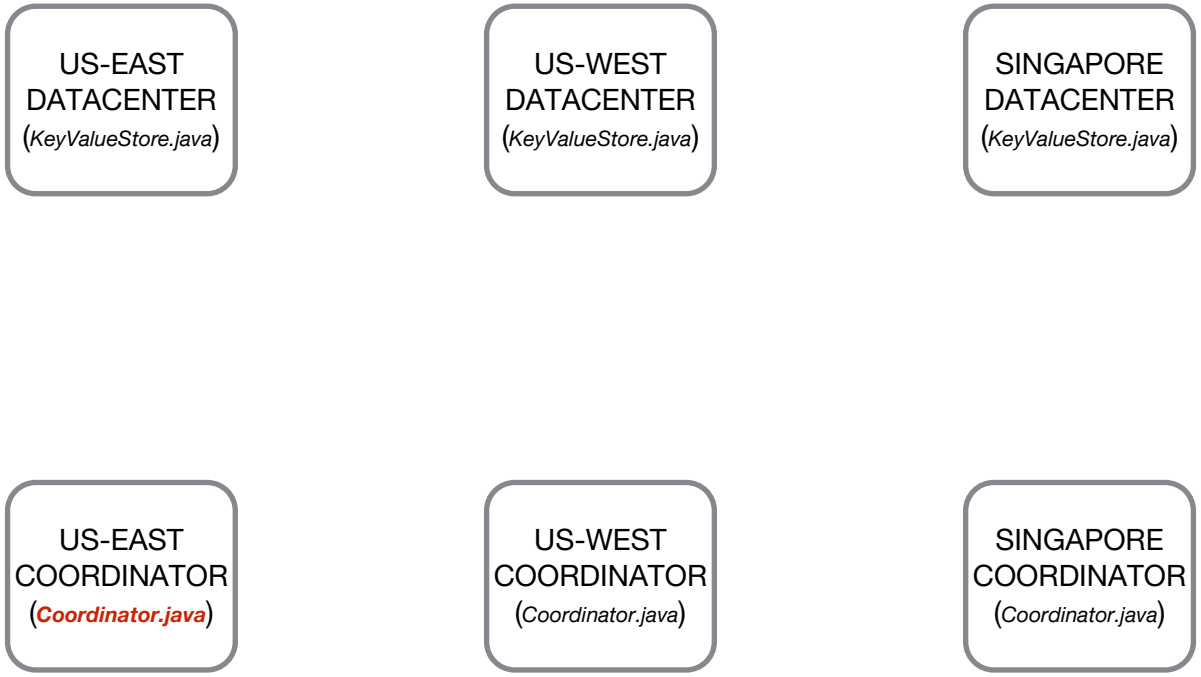
- Complete the KeyValueStore.java (on the datacenter instance) and Coordinator.java (on the coordinator instance).
- Support 3 consistencies for PUT/GET request: Strong, Causal and Eventual.



Example workflow for PUT request in strong consistency

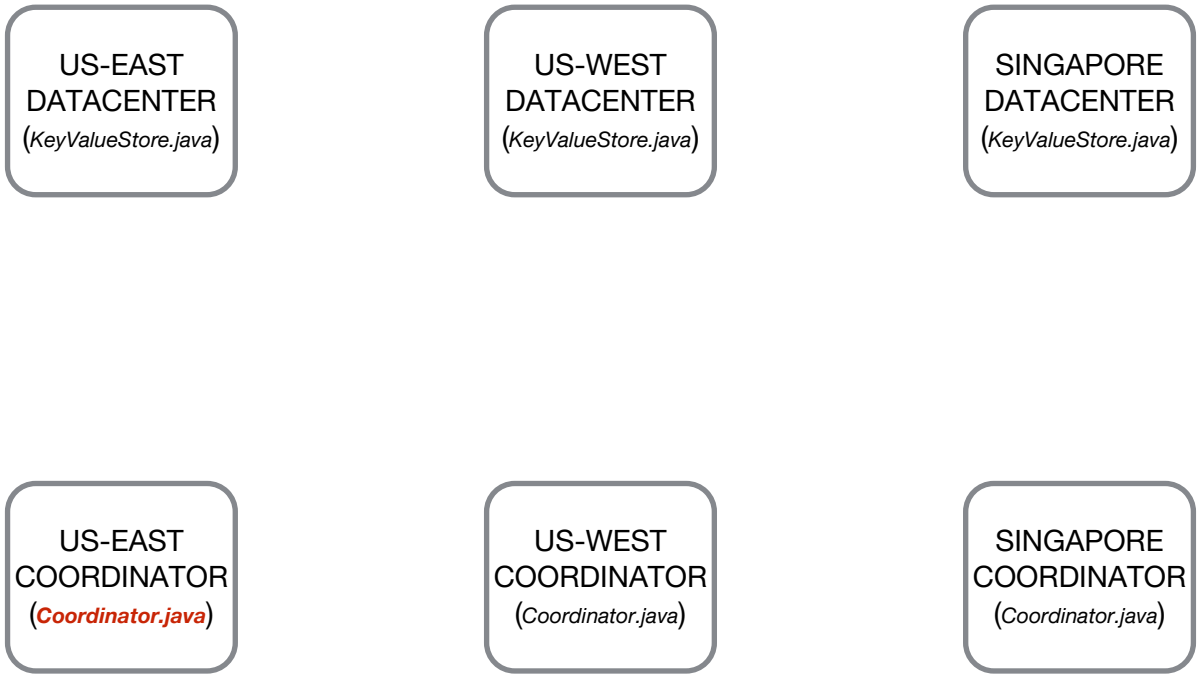


Example workflow for PUT request in strong consistency



Example workflow for a PUT request in strong consistency

- If US-EAST is responsible for key “X”

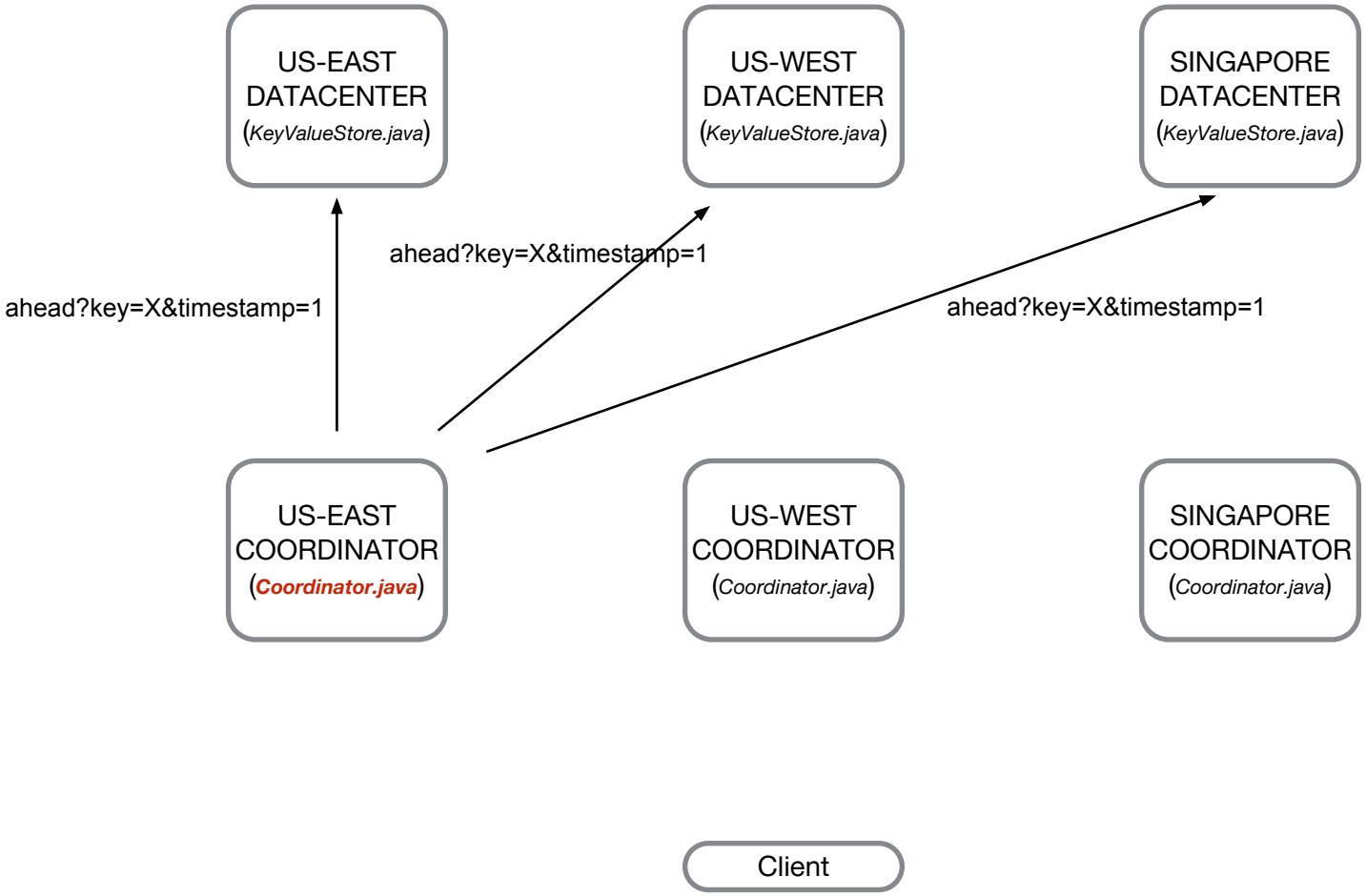


You should call KeyValueLib.AHEAD(“X”,1) to notify all 3 datacenters of this PUT request, and you should also do the following things:

- Lock PUT request involving “x” before the current PUT request completes.
- Lock GET request involving “x” before the current PUT request completes, you could choose to lock in either coordinator or data center.

Example workflow for PUT request in strong consistency

- If US-EAST is responsible for key “X”

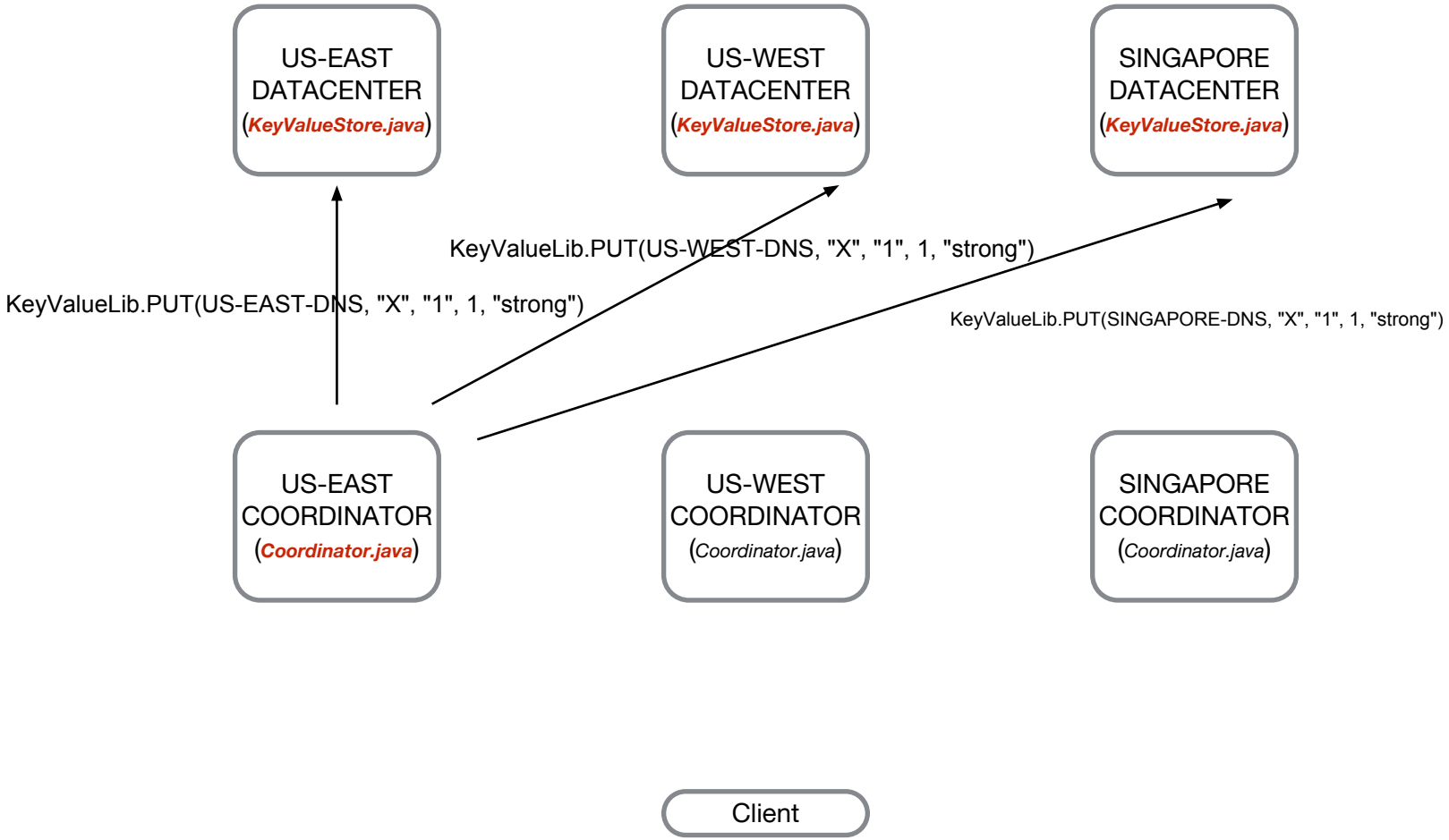


Example workflow for PUT request in strong consistency

- If US-EAST is responsible for key “X”

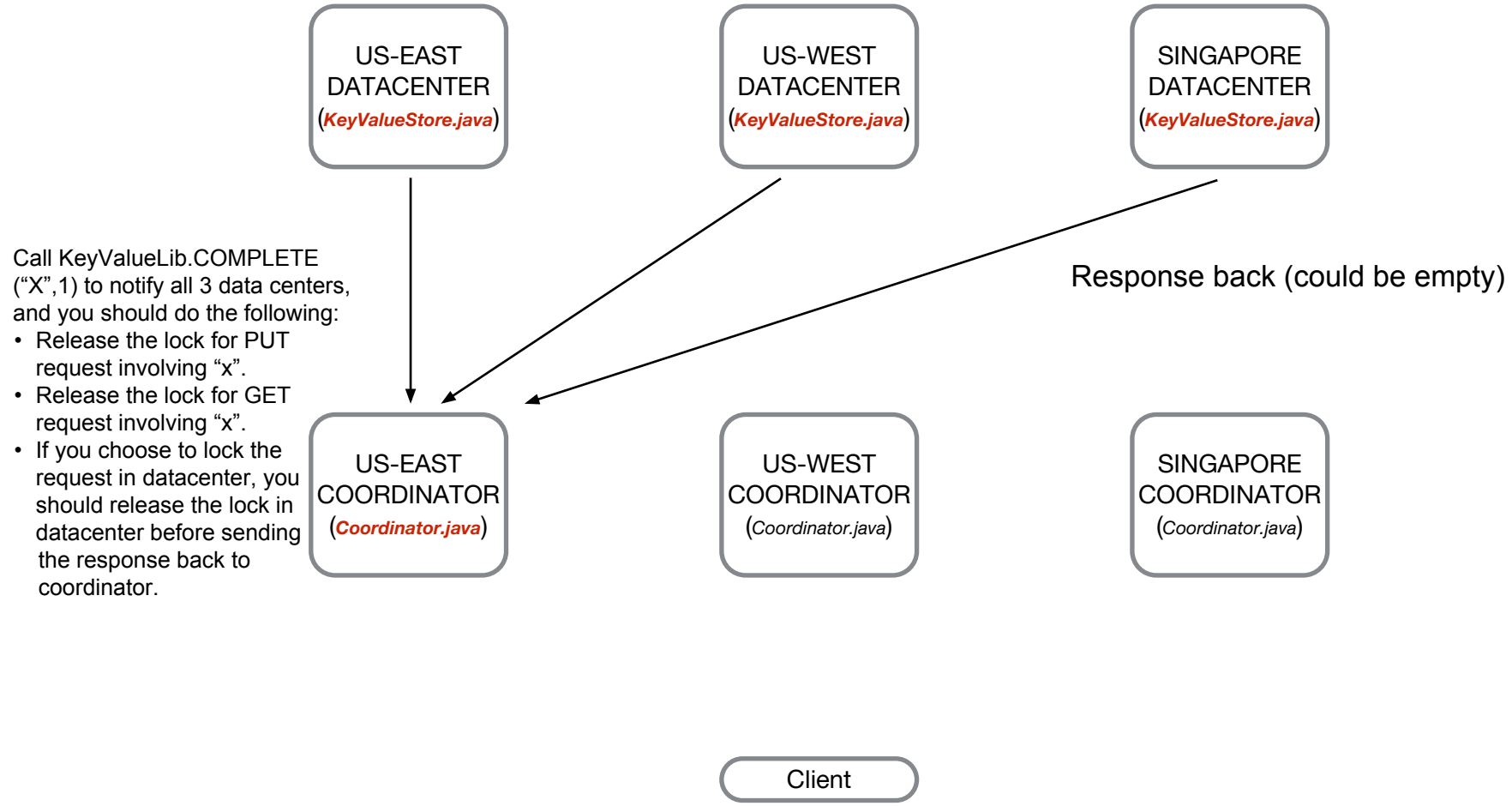
On receiving the request, your code in KeyValueStore.java should do following things:

- Store the data (you could use the StoreValue.java we provided)
- Remember to adjust the timestamp if the request is from a coordinator in a different region (you could use the Skews.java we provided)
- For strong consistency, ordering the request by timestamp is important. Maybe you need an additional data structure to keep track of this.



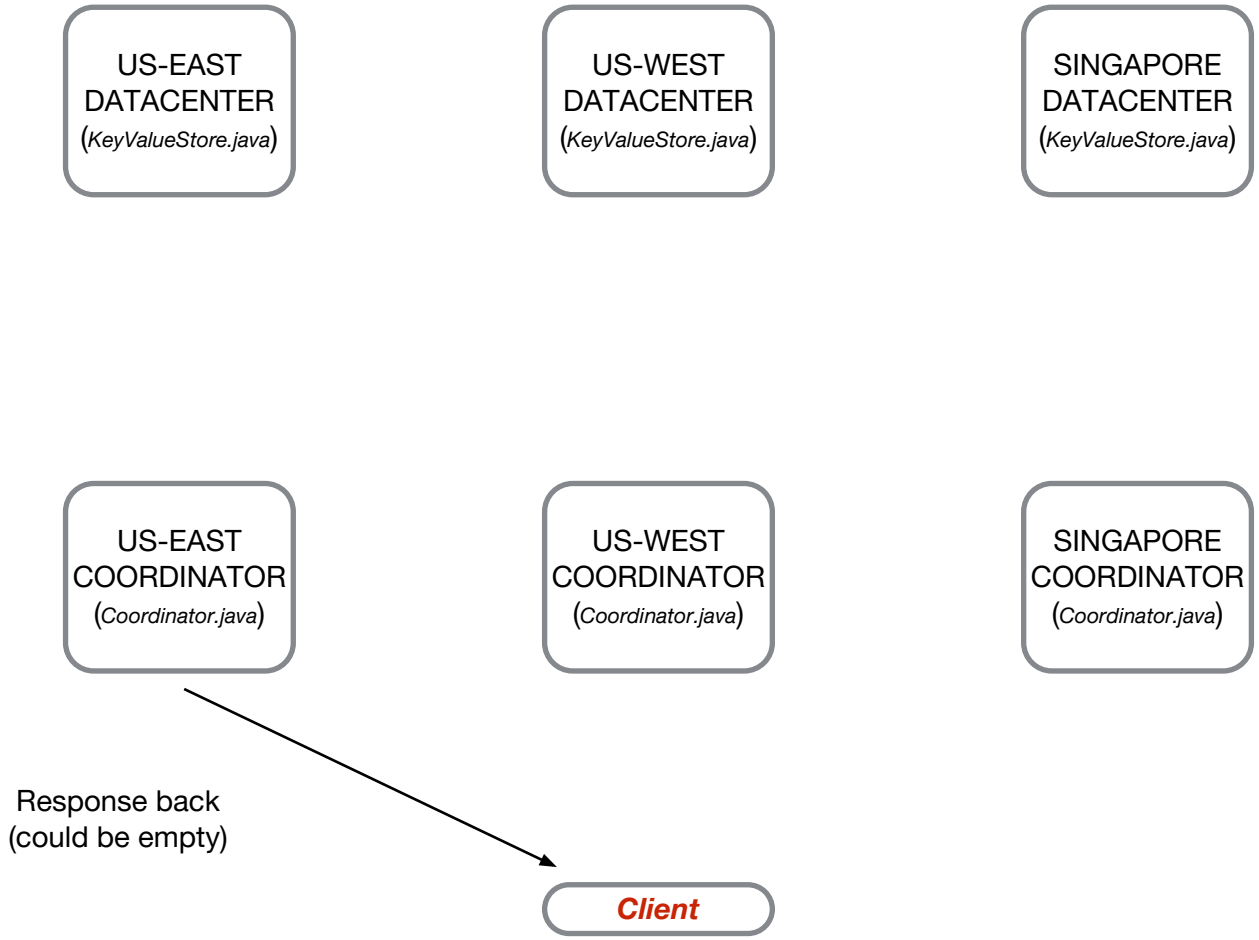
Example workflow for PUT request in strong consistency

- If US-EAST is responsible for key “X”



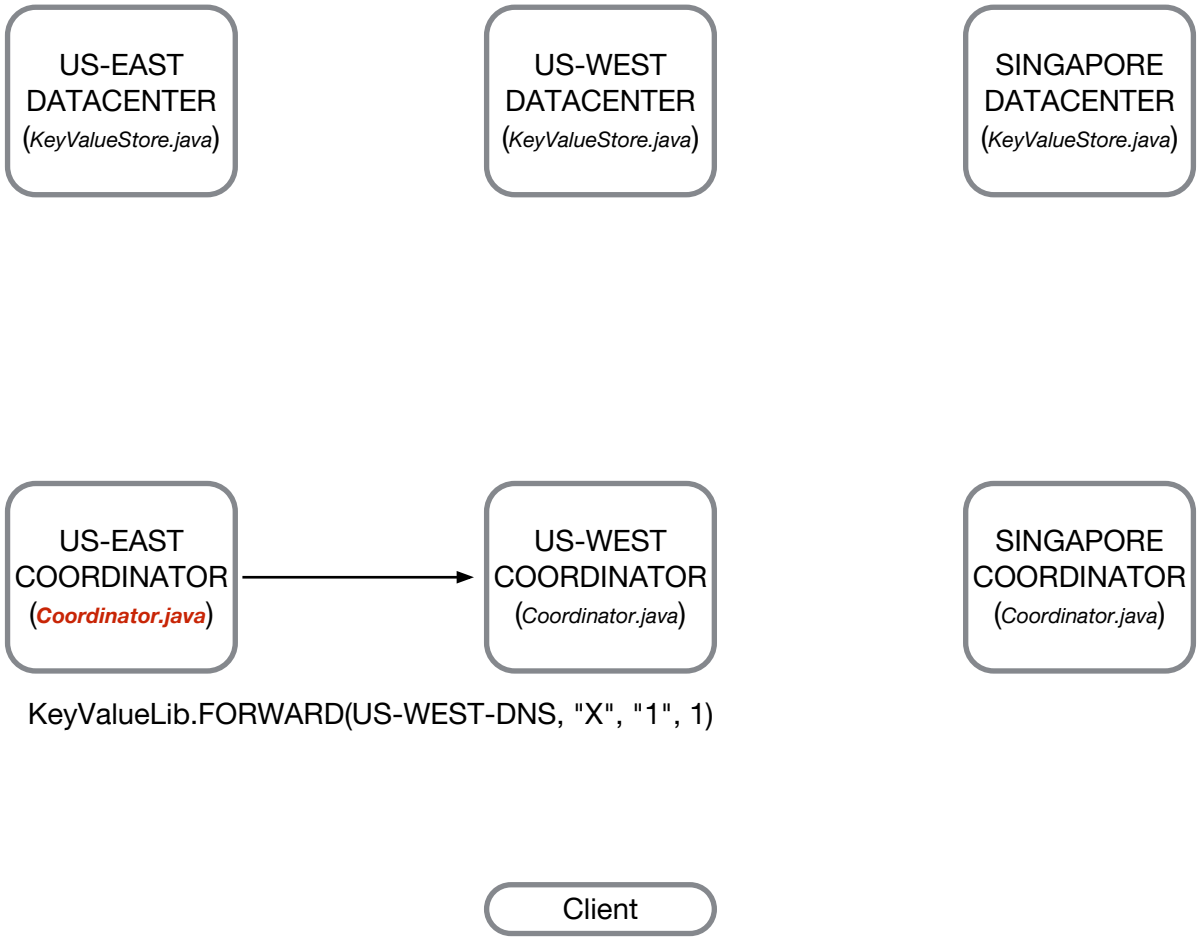
Example workflow for PUT request in strong consistency

- If US-EAST is responsible for key “X”



Example workflow for PUT request in strong consistency

- If US-WEST is responsible for key “X”



More Hints:

- Remember to adjust the timestamp in datacenter when the request is from a different region.
- In strong consistency, “AHEAD” and “COMPLETE” would help you to lock the GET request. You should think carefully of how they would work.
- Lock all datacenters in strong consistency and lock individual datacenters in causal consistency.
- In causal consistency, do NOT block a GET request.
- Eventual consistency could be trivial to implement.

Suggestions:

- You should first know the difference between the 3 policies before writing your code.
- Think about possible race conditions.
- Read the hints on the TPZ handout carefully.
- Don’t modify any class except Coordinator.java and KeyValueStore.java.
- You could optimize your hashing algorithm to reduce the number of forward operations.

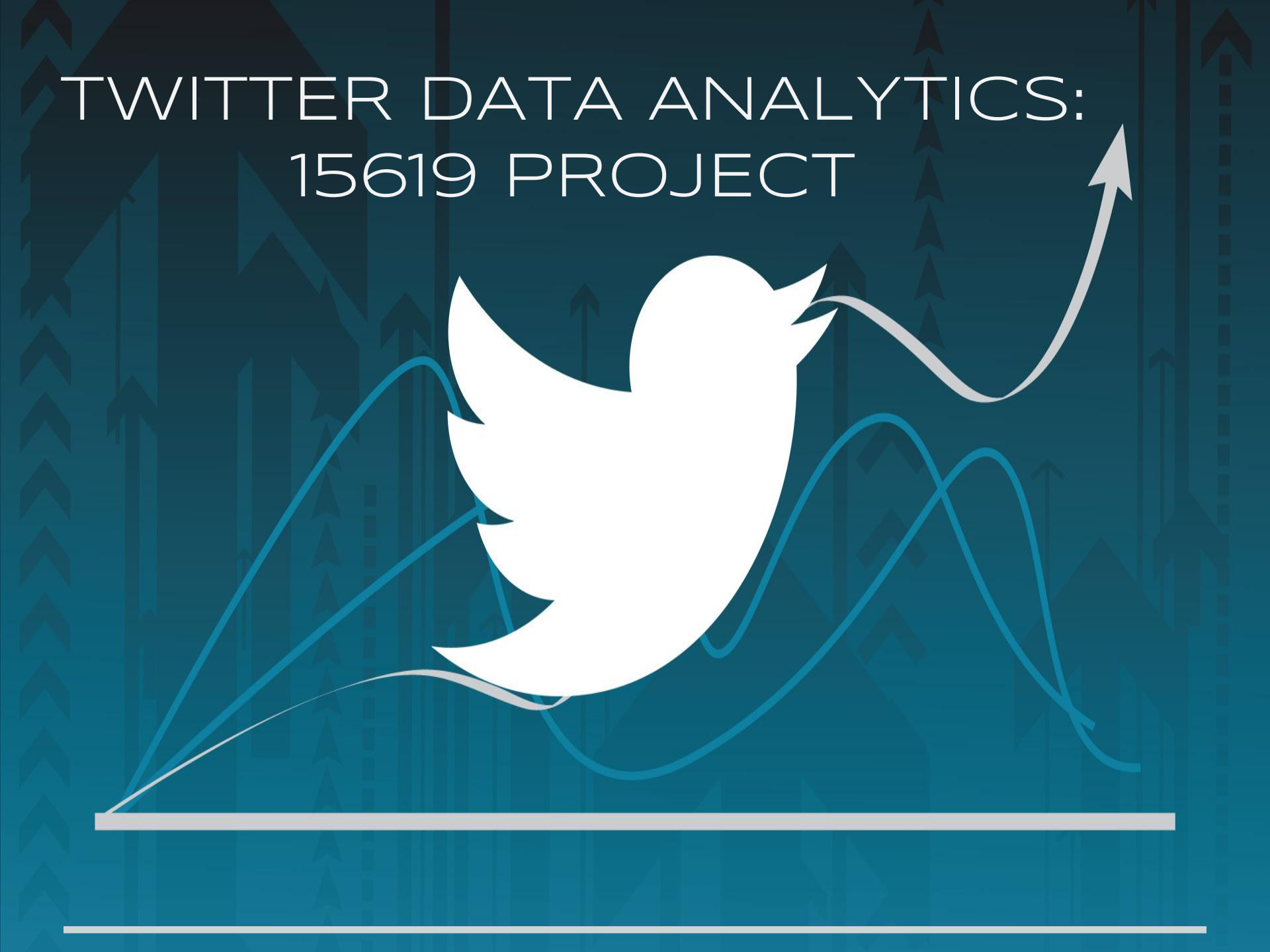
How To Test:

- Run “./vertx run Coordinator.java” and “./vertx run KeyValueStore.java” to start the vertx server on each of the data centers and coordinators. (You could use nohup to run it in background)

- Use “./consistency_checker strong”,“./consistency_checker causal” or “./consistency_checker eventual” to test your implementation of each consistency. (Our grader uses the same checker)

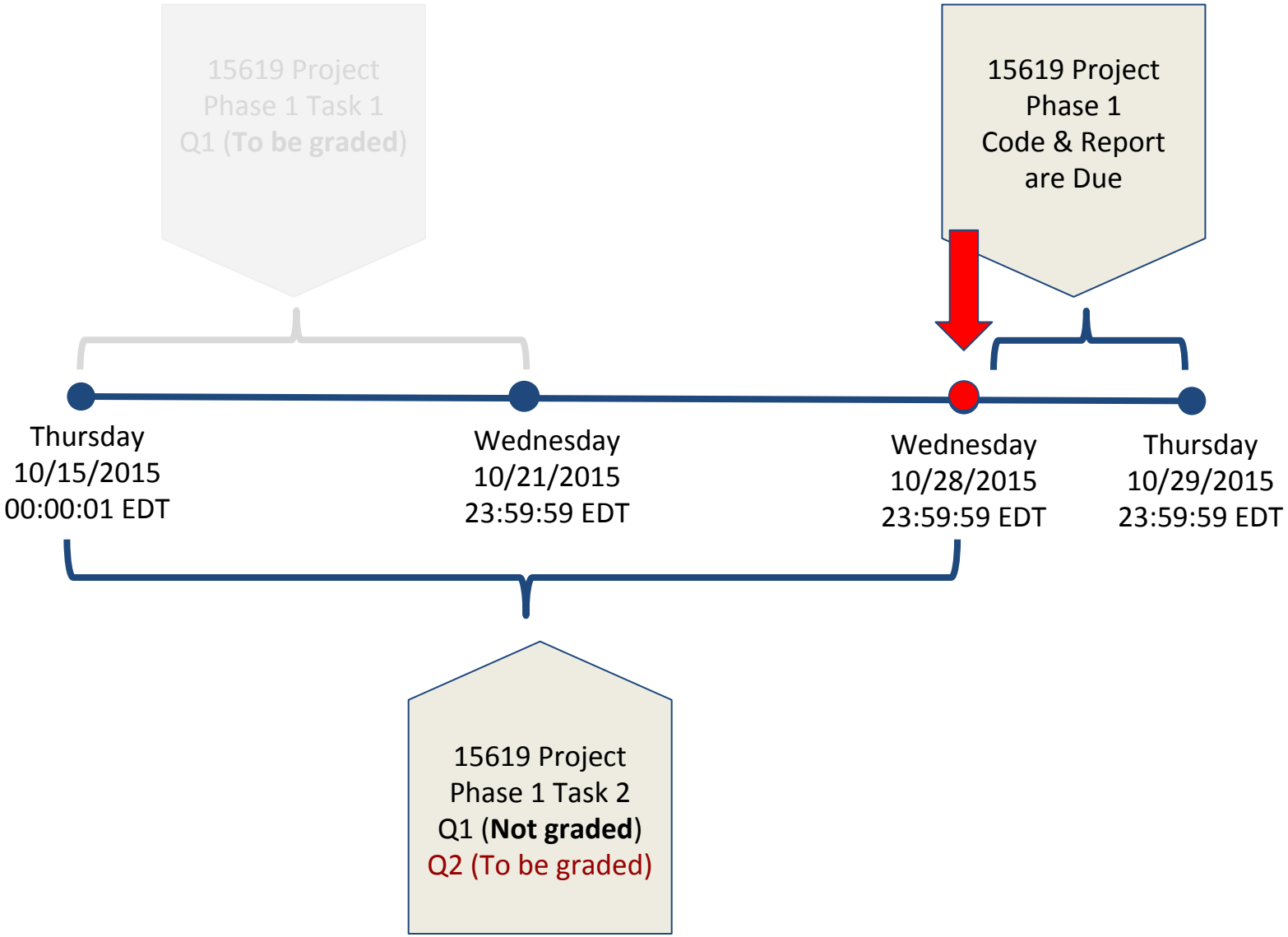
- If you want to test one simple PUT/GET request, you could directly enter the request in your browser.

TWITTER DATA ANALYTICS: 15619 PROJECT

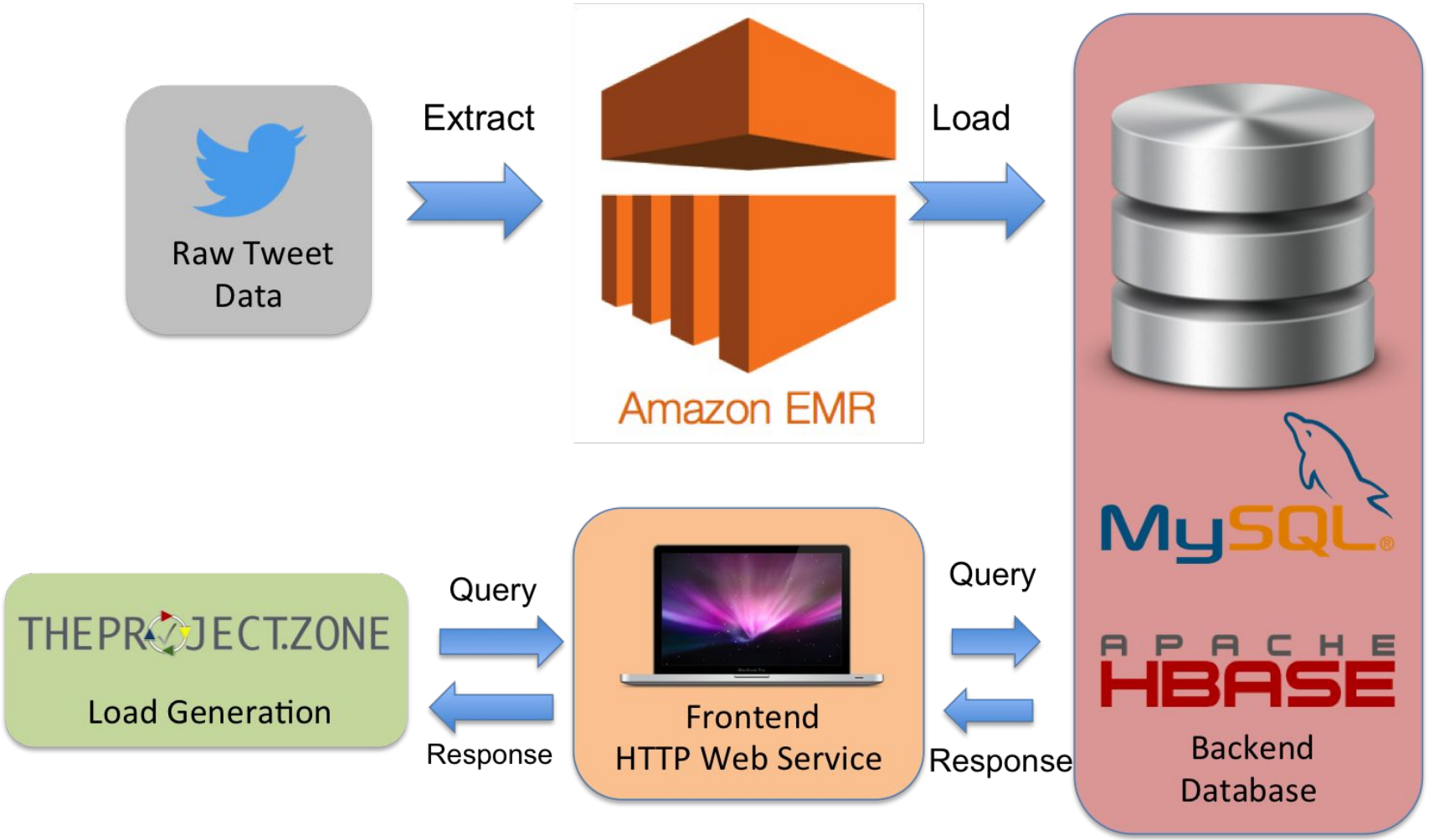


15619 Project Phase 1 Deadlines

- 1 week for Q1
- 2 weeks for Q2



15619 Project System Architecture



Q1 : Heartbeat and Authentication

- Task
  - Big integer division
  - Decryption
- Things to consider
  - Is framework selection important? Explore!
  - How important is it to minimize Latency?
  - What can I do if I want to use multiple front-end instances?

Q2 : Text Cleaning And Analysis

- ETL Task
  - Time filtering
  - Sentiment score calculation
  - Text censoring
- Request
  - Userid
  - Timestamp
- Return
  - TweetID, Sentiment score, Censored text

Q2: Sentiment score

Amazingly, despite the nice,cloudy weather, the BEST Hope for us to enjoy is to study CLOUD COMPUTING. Cloud is supper-interesting.

Sentiment score: ??

Word	**Score**		**Word**	**Score**	
amazing	4		interesting	3	
best	3		enjoy	1	
nice	2		super	7	
hope	2		study	-100	

Q2: Sentiment score

Amazingly, despite the **nice**,cloudy weather, the **BEST Hope** for us to **enjoy** is to **study** CLOUD COMPUTING. Cloud is supper-**interesting**.

Sentiment score: -89

Word	**Score**		**Word**	**Score**	
amazing	4		interesting	3	
best	3		enjoy	1	
nice	2		super	7	
hope	2		study	-100	

Q2: Text Censorship

Rot13ed Banned words
pybhq
vagrerfgvat

ROT13 →

Banned words
cloud
interesting

Amazingly, despite the nice,cloudy weather, the BEST Hope for us to enjoy is to study CLOUD COMPUTING. Cloud is supper-interesting.

↓

Amazingly, despite the nice,cloudy weather, the BEST Hope for us to enjoy is to study C***D COMPUTING. Cloud is supper-i*********g.

Q2: Other issues

- Unicode

  الحوسبة السحابية

  बादल कंप्यूटिंग

  云计算

  クラウドコ

  ンピューティング

  ಕೌಲ್ಡ್ ಕಂಪೂಯ್ಟಿಂ

  וואָלקן קאַמפּיוטינג

  облачныхвычислений

- Multiple tweets at the same time for a single user

What's due soon?

- Phase 1 Report & Code Deadline
  - **[11.59 PM Pitt Thursday 10/29]**
  - Upload to TheProject.Zone
  - No code ⇒ ZERO POINTS FOR ENTIRE PHASE 1
  - Missing files ⇒ ZERO POINTS FOR ENTIRE PHASE 1

- Very High Standard Expected in Report (25%)
  - Make sure you highlight failures and learning
  - If you didn't do well, explain why
  - If you did, explain how
  - A really good report showing effort can compensate for poor performance

Upcoming Deadlines

- Quiz 8: Unit 4 - Module 15 - Case Studies: DFSs

  Open: 10/30/2015 12:01AM Pittsburgh

  Due: 10/30/2015 11:59PM Pittsburgh

- Project 3.3: Consistency in Distributed K-V Stores

  Due: 11/01/2015 11:59PM Pittsburgh

- 15619Project: Phase 1, Task 2

  Due: 10/28/2015 11:59PM Pittsburgh

- 15619Project: Phase 1, Report

  Due: 10/29/2015 11:59PM Pittsburgh