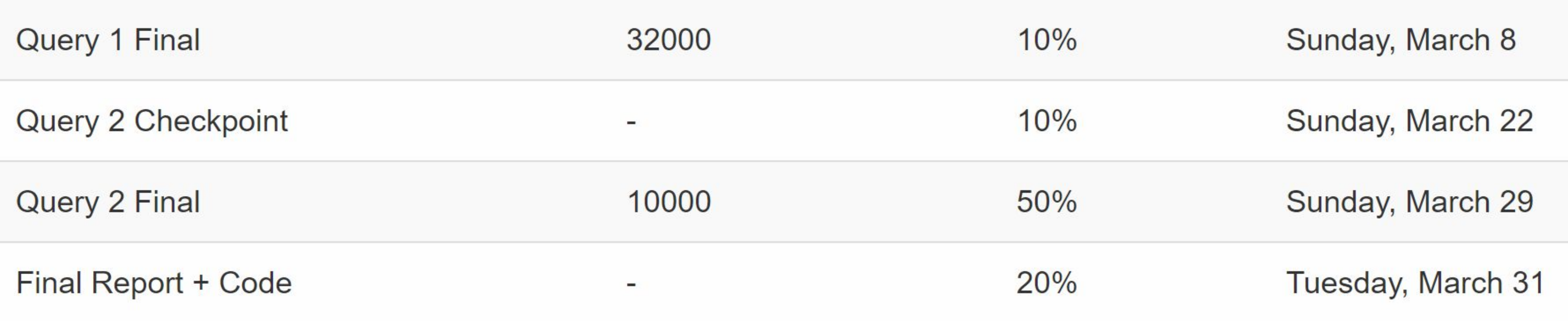
Read about Query 2 Now.
Start ETL Now.



Query 1 Final	32000	10%	Sunday, March 8
Query 2 Checkpoint	-	10%	Sunday, March 22
Query 2 Final	10000	50%	Sunday, March 29
Final Report + Code	-	20%	Tuesday, March 31

After spring break, you got one week
to meet the Query 2 checkpoint.

Question: Is 1 week enough time for that?
Hint: No. Start now.

15-319 / 15-619
Cloud Computing

Recitation 8

Mar 3, 2020

Overview

- **Last week's reflection**
  - Project 3.1
  - OLI Unit 3 - Module 13
  - Quiz 6
- **This week's schedule**
  - Project 3.2
  - OLI Unit 4 - Module 14
  - Quiz 7 (Due Thursday 3/5)
  - Online Programming Exercise for Multi-Threading
- **Team Project, Twitter Analytics**
  - Phase 1 is out! Q1 final due on 3/8.
  - Phase 1 due, 3/29.

Last Week

- **Unit 3: Virtualizing Resources for the Cloud**
  - Module 13: Storage and network virtualization
- **Quiz 6**
- **Project 3.1**
  - Files v/s Databases (SQL & NoSQL)
    - Flat files
    - MySQL
    - Redis & Memcached
    - HBase
      - Read the NoSQL and HBase basics primer

This Week

- **OLI : Unit 4 Module 14 - Cloud Storage**
- **Quiz 7** - **Thursday**, March 5
- **Project 3.2** - Sunday, March 8
  - Social Networking Timeline with Heterogeneous Backends
    - MySQL
    - Neo4j
    - MongoDB
    - Choosing Databases, Storage Types & Tail Latency
- **Online Programming Exercise for Multi-Threading on Cloud9**
  - This week
- **Team Project, Phase 1 released**

Conceptual Topics - OLI Content

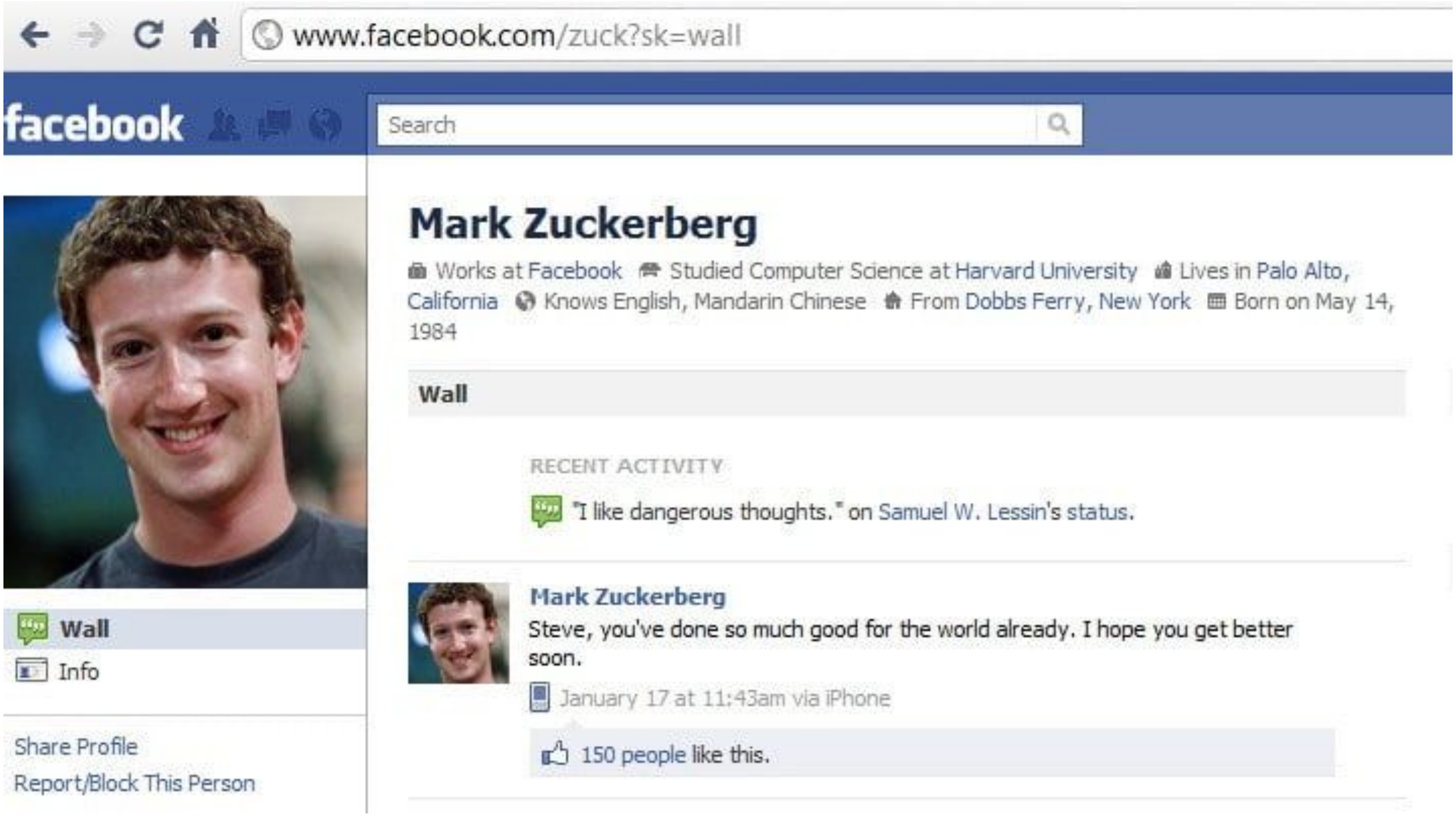
- **OLI Unit 4 - Module 14: Cloud Storage**
  - File Systems and Databases
  - Scalability and Consistency
  - NoSQL, NewSQL and Object Storage
  - CAP theorem

- **Quiz 7**
  - **DUE on Thursday, March 05**
    - **Remember to click submit**
      - **Within 2 hours, and**
      - **Before the deadline!**

Individual Projects

- DONE
  - P3.1: Files vs Databases - comparison and Usage of flat files, MySQL, Redis, and HBase
  - NoSQL Primer
  - HBase Basics Primer
  - MongoDB Primer
- **NOW**
  - P3.2: Social networking with heterogeneous backends
- Coming Up
  - P3.3: Multi-threading Programming and Consistency

A Social Network Service



www.facebook.com/zuck?sk=wall

Mark Zuckerberg

Works at Facebook · Studied Computer Science at Harvard University · Lives in Palo Alto, California · Knows English, Mandarin Chinese · From Dobbs Ferry, New York · Born on May 14, 1984

Wall

RECENT ACTIVITY

"I like dangerous thoughts." on Samuel W. Lessin's status.

Mark Zuckerberg

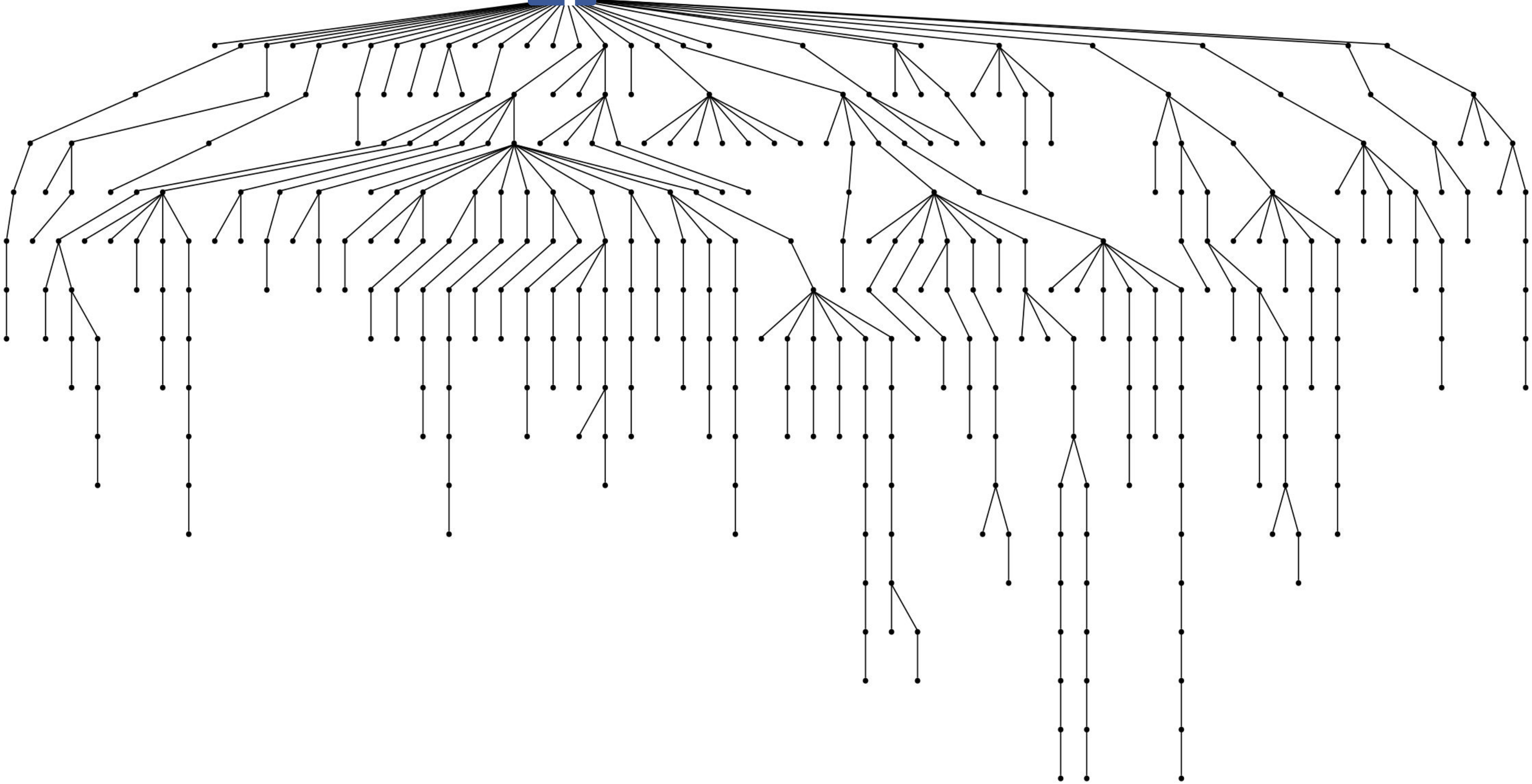
Steve, you've done so much good for the world already. I hope you get better soon.

January 17 at 11:43am via iPhone

150 people like this.

High Fanout in Data Fetching

A single f page, requires many data fetch operations



Nishtala, R., Fugal, H., Grimm, S., Kwiatkowski, M., Lee, H., Li, H. C., ... & Venkataramani, V. (2013, April). Scaling Memcache at Facebook. In *nsdi* (Vol. 13, pp. 385-398).

Graph Database Neo4j

- Designed to treat the relationships between data as equally important as the data
  - Relationships are very important in social graphs
- Property graph model
  - Nodes
  - Relationships
  - Properties
- Cypher query language
  - Declarative, SQL-inspired language for describing patterns in graphs visually

MongoDB

- Document Database
  - Schema-less model
- Highly Scalable
  - Automatically shards data among multiple servers
  - Does load-balancing
- Allows for Complex Queries
  - MapReduce style filter and aggregations
  - Geospatial queries

mongoDB®

P3.2 - Overview

- Build a social network about Reddit comments
- Dataset generated from Reddit.com
  - **users.csv**, **links.csv**, **posts.json**
- Build a social network timeline on the Reddit.com data
  - **Task 1**: Basic login
  - **Task 2**: Social graph
  - **Task 3**: Rank user comments
  - **Task 4**: Generate user timeline
  - **Task 5**: Caching mechanism
- **Task 6: Understanding Tail Latency, BLOBs, Storage Types, and Selecting Databases**
  - Answer questions on relevant topics and choose the right database and storage type for a given scenario

TDD with Mockito

- Mockito is an open-source testing framework that allows the creation of test double objects (mock objects).
- It is used to mock interfaces so that the specific functionality of an application can be tested without using real resources such as databases, expensive API calls, etc.
- You are required to understand the given implementation, and may use it to quickly debug your solution for Task 1.

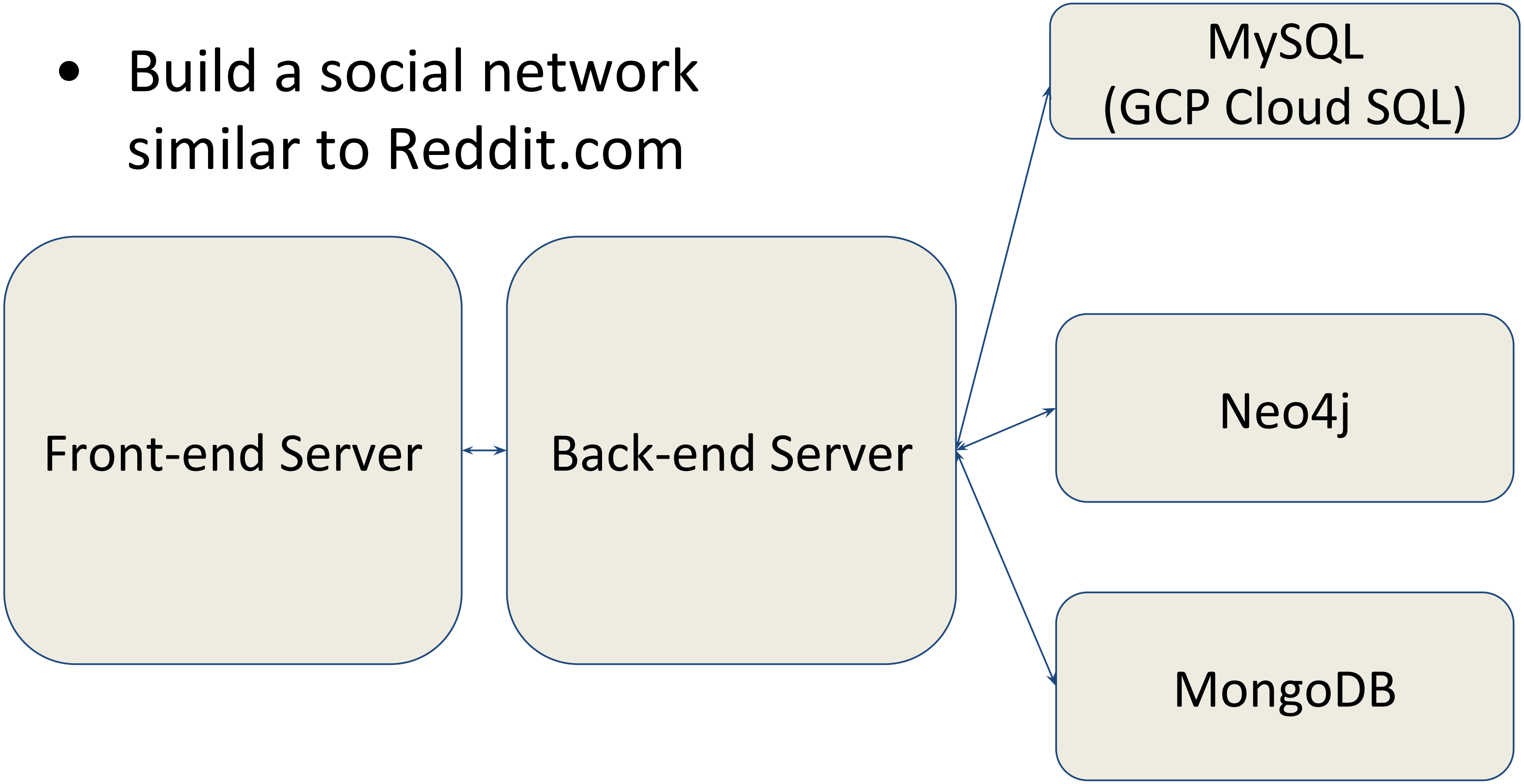
P3.2 - Reddit Dataset

- Task 1: User profiles
  - User authentication system : GCP Cloud SQL(users.csv)
  - User info / profile : GCP Cloud SQL
- Task 2: Social graph of the users
  - Follower, followee : Neo4j (links.csv)
- Task 3: User activity system
  - All user generated comments : MongoDB (posts.json)
- Task 4: User timeline
  - Put everything together
- Task 5: Caching Mechanism
  - Cache the requests

reddit

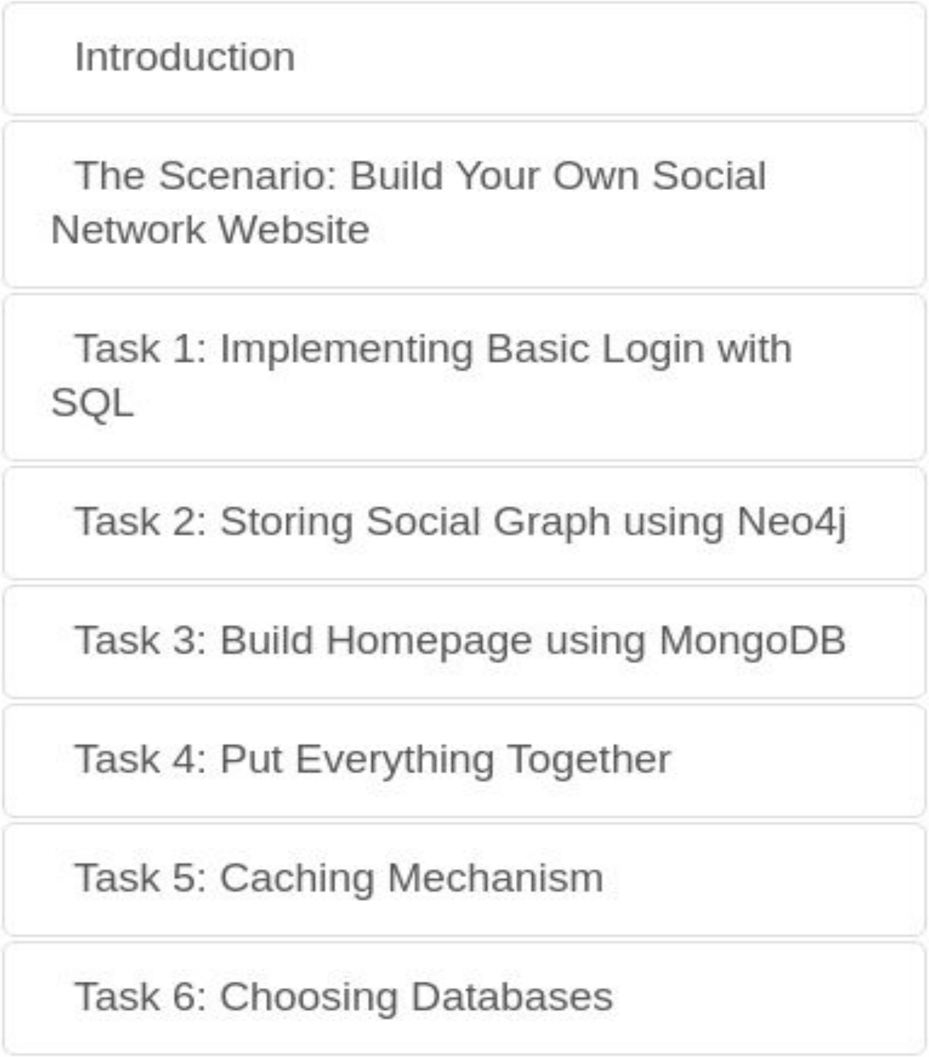
P3.2 - Architecture

- Build a social network similar to Reddit.com

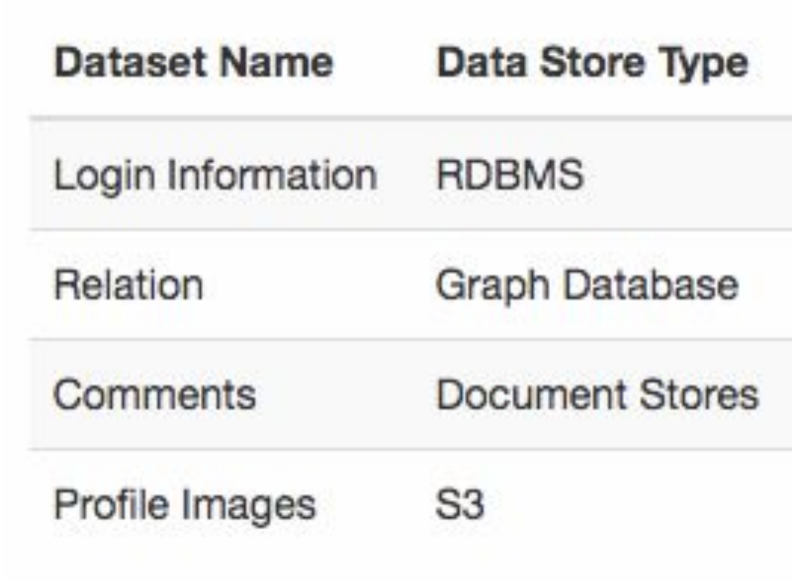


○ Some images in the front-end are broken. No worries as long as you can get valid responses using “curl” command.

Tasks, Datasets & Storage



- Introduction
- The Scenario: Build Your Own Social Network Website
- Task 1: Implementing Basic Login with SQL
- Task 2: Storing Social Graph using Neo4j
- Task 3: Build Homepage using MongoDB
- Task 4: Put Everything Together
- Task 5: Caching Mechanism
- Task 6: Choosing Databases



Dataset Name	Data Store Type
Login Information	RDBMS
Relation	Graph Database
Comments	Document Stores
Profile Images	S3

P3.2 - Task 6

- **Issues of dealing with Scale**
  - An overview of the systems issues that arise with scale and how they were addressed in the context of Facebook.
    - Tail Latency and Fanout
    - BLOBs and Storage Types
      - Cost and performance
    - Learn how popularity and freshness of data plays a role in designing efficient social networking backends.

P3.2 - Task 6

- **Choosing Databases & Storage Types**
  - Use your knowledge and experience gained working with the databases in the project to
    - Identify advantages and disadvantages of various DBs
    - Pick suitable DBs for particular application requirements
    - Provide reasons on why a certain DB is suitable under the given constraints
  - Instructions provided in **runner.sh**

Terraform

- **Required in P3.2**
- **Required in the team project, get some practice**
- Files provided
- Use **‘terraform destroy’** to terminate resources
- This project is on GCP, so apply the following tag
  - The tag is “3-2” instead of “3.2” (for GCP only)

P3.2 - Reminders and Suggestions

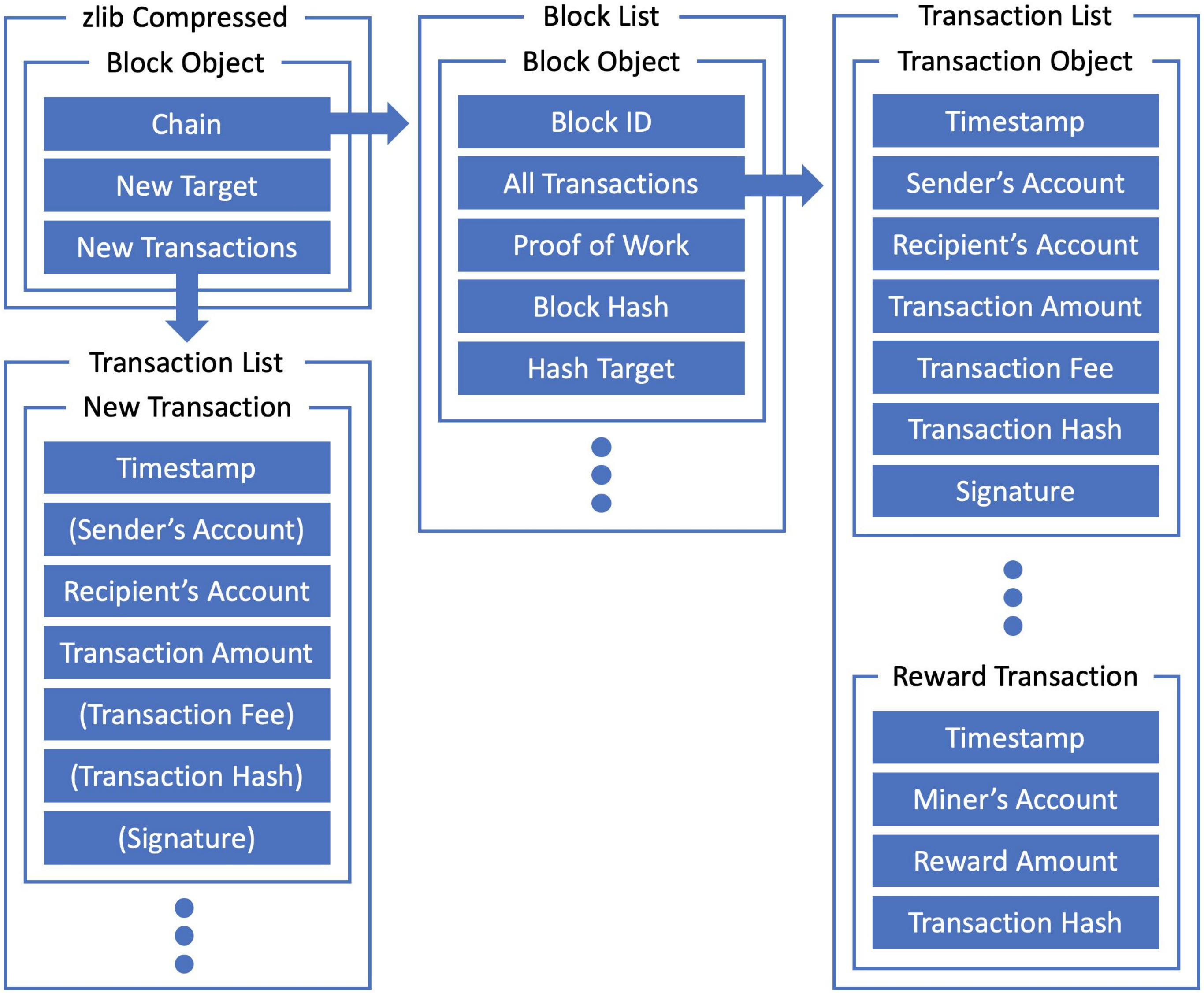
- Set up a budget alarm on GCP
  - Suggested budget: $15
  - No penalties
- Learn and practice using a standard JSON Library. This will prove to be valuable in the Team Project
  - **Google GSON** - Recommended for Java
- Set up Gcloud in your environment
- No AWS instances on your individual AWS account are allowed
  - Otherwise you will receive warning emails and penalties

P3.2 - Reminders and Suggestions

- In Task 4 and 5, you will use the databases from all previous tasks. Make sure to have **all** the databases loaded and ready when working on Task 4 and 5.
- You can submit one task at a time using the submitter. Remember to have your Back-end Server VM running when submitting.
- Make sure to terminate **all** resources using “terraform destroy” after the final submission. Double check on the GCP console that all resources were terminated.

TEAM PROJECT
Twitter Data Analytics

Query 1 Recap



zlib Compressed

Block Object

Chain

New Target

New Transactions

Transaction List

New Transaction

Timestamp

(Sender’s Account)

Recipient’s Account

Transaction Amount

(Transaction Fee)

(Transaction Hash)

(Signature)

Block List

Block Object

Block ID

All Transactions

Proof of Work

Block Hash

Hash Target

Transaction List

Transaction Object

Timestamp

Sender’s Account

Recipient’s Account

Transaction Amount

Transaction Fee

Transaction Hash

Signature

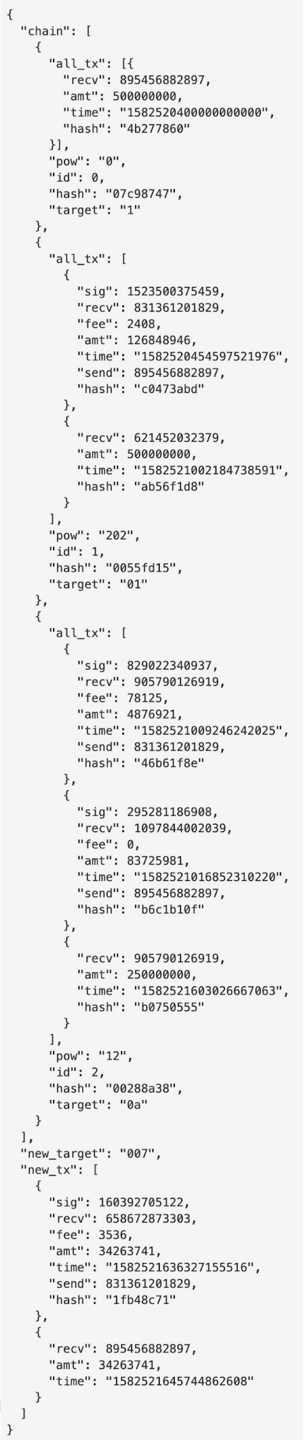
Reward Transaction

Timestamp

Miner’s Account

Reward Amount

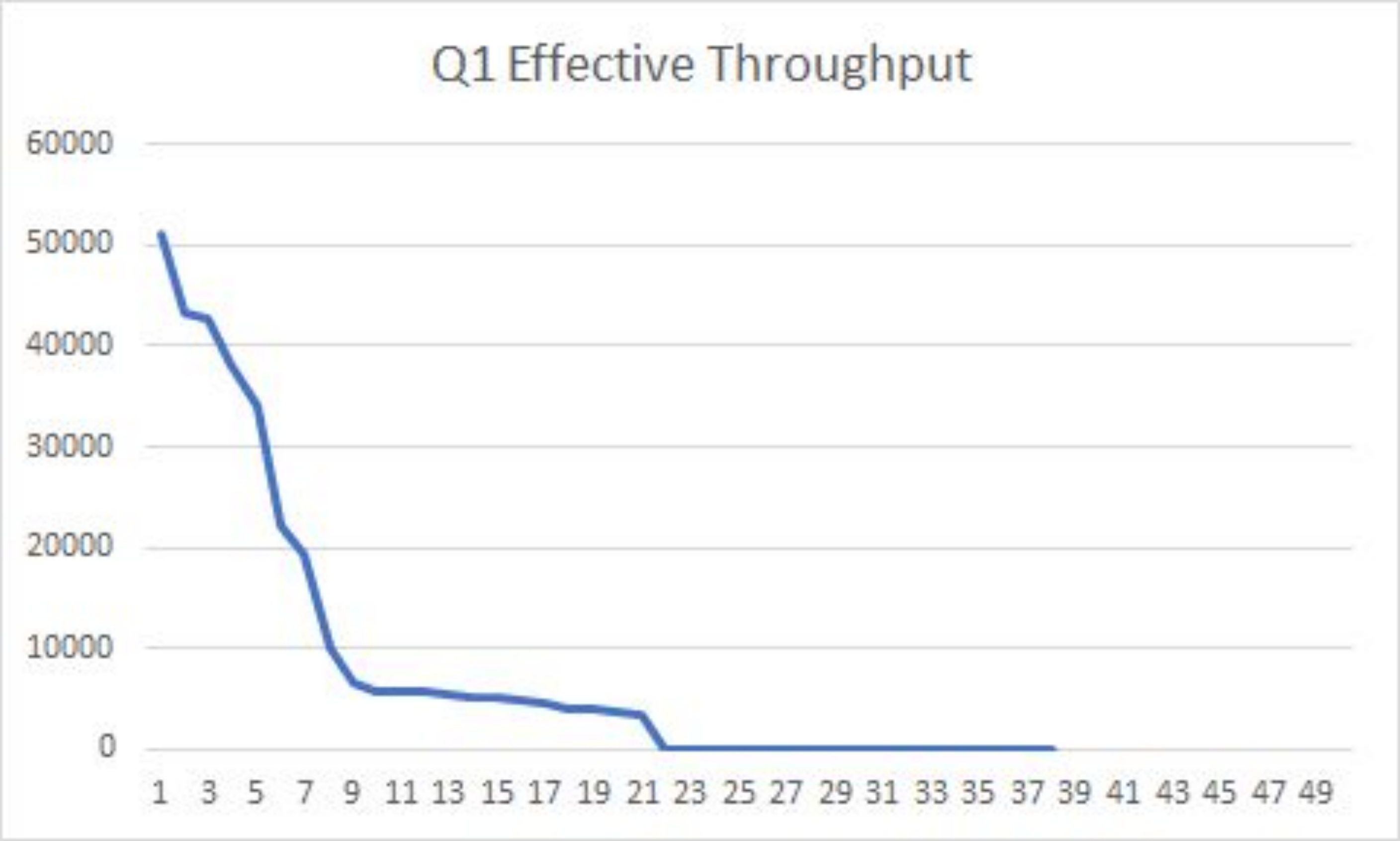
Transaction Hash



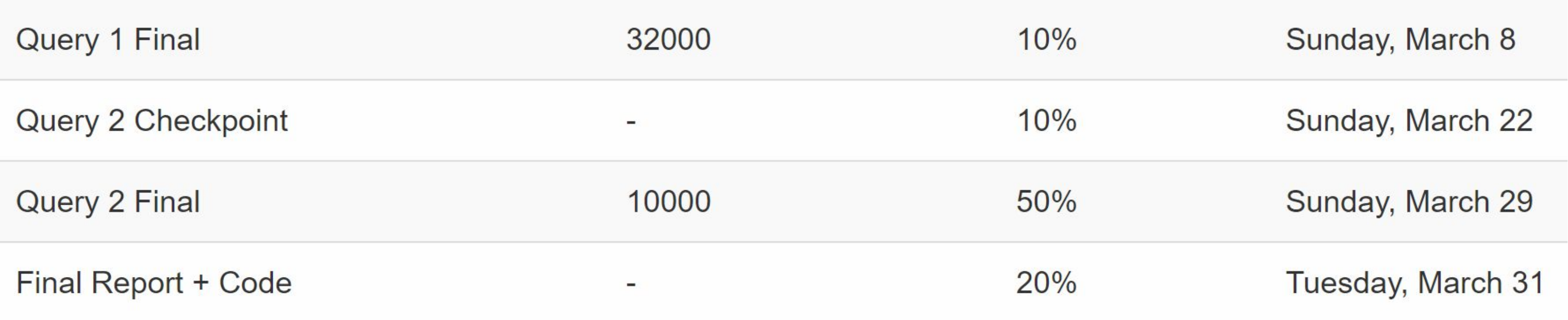
```
{
  "chain": [
    {
      "all_tx": [{
        "recv": 895456882897,
        "amt": 500000000,
        "time": "1582520400000000000",
        "hash": "4b277860"
      }],
      "pow": "0",
      "id": 0,
      "hash": "07c98747",
      "target": "1"
    },
    {
      "all_tx": [
        {
          "sig": 1523500375459,
          "recv": 831361201829,
          "fee": 2408,
          "amt": 126848946,
          "time": "1582520454597521976",
          "send": 895456882897,
          "hash": "c0473abd"
        },
        {
          "recv": 621452032379,
          "amt": 500000000,
          "time": "1582521002184738591",
          "hash": "ab56f1d8"
        }
      ],
      "pow": "202",
      "id": 1,
      "hash": "0055fd15",
      "target": "01"
    },
    {
      "all_tx": [
        {
          "sig": 829022340937,
          "recv": 905790126919,
          "fee": 78125,
          "amt": 4876921,
          "time": "1582521009246242025",
          "send": 831361201829,
          "hash": "46b61f8e"
        },
        {
          "sig": 295281186908,
          "recv": 1097844002039,
          "fee": 0,
          "amt": 83725981,
          "time": "1582521016852310220",
          "send": 895456882897,
          "hash": "b6c1b10f"
        },
        {
          "recv": 905790126919,
          "amt": 250000000,
          "time": "1582521603026667063",
          "hash": "b0750555"
        }
      ],
      "pow": "12",
      "id": 2,
      "hash": "00288a38",
      "target": "0a"
    }
  ],
  "new_target": "007",
  "new_tx": [
    {
      "sig": 160392705122,
      "recv": 658672873303,
      "fee": 3536,
      "amt": 34263741,
      "time": "1582521636327155516",
      "send": 831361201829,
      "hash": "1fb48c71"
    },
    {
      "recv": 895456882897,
      "amt": 34263741,
      "time": "1582521645744862608"
    }
  ]
}
```

Team Project - Q1 CKPT1

- 38 teams attempted a Query 1 submission.
- 20 teams got a 10-minute submission
- 5 teams reached 32,000 RPS

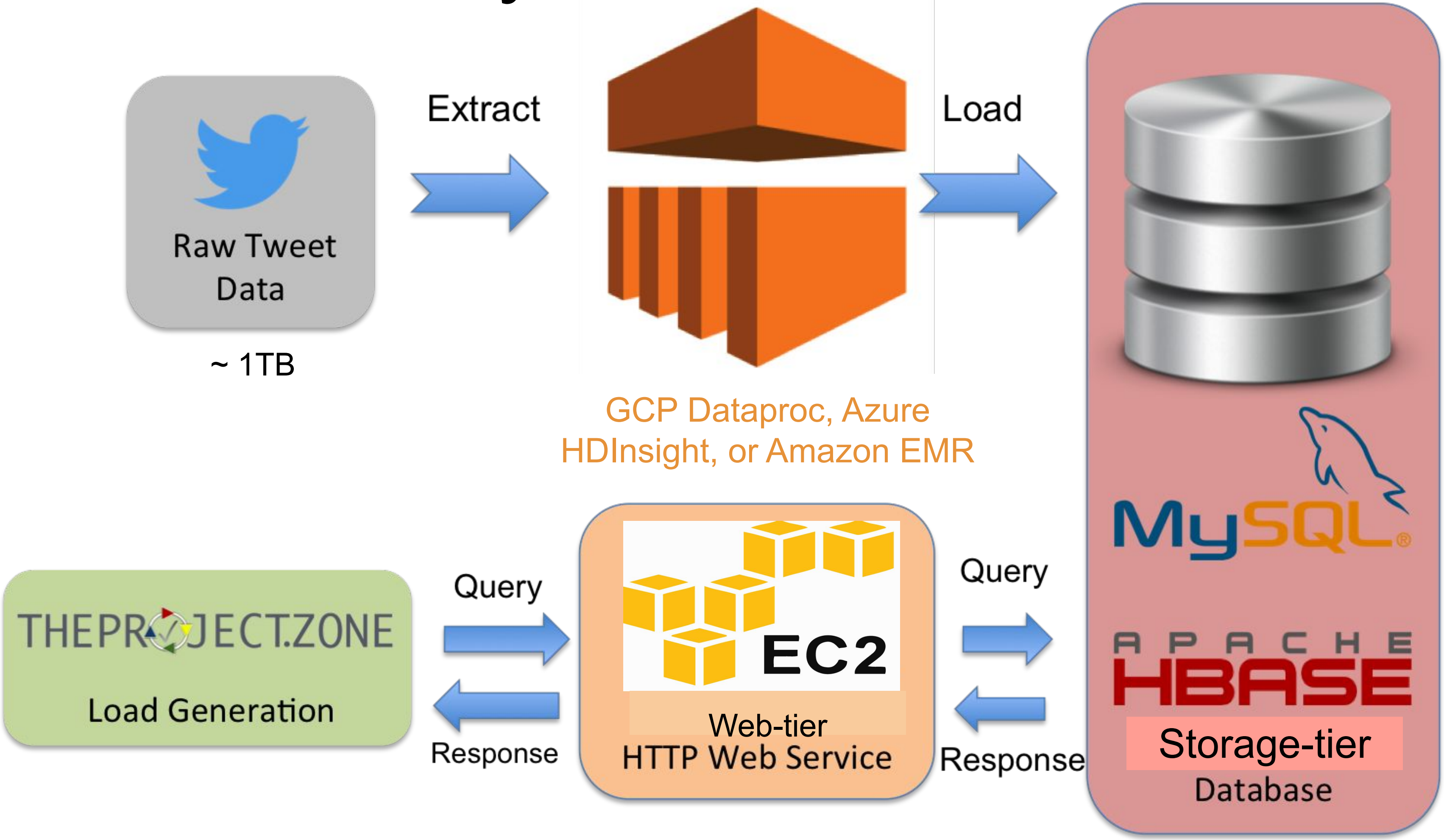


Read about Query 2 Now.
Start ETL Now.



Query 1 Final	32000	10%	Sunday, March 8
Query 2 Checkpoint	-	10%	Sunday, March 22
Query 2 Final	10000	50%	Sunday, March 29
Final Report + Code	-	20%	Tuesday, March 31

Twitter Analytics System Architecture



Raw Tweet Data

~ 1TB

Extract

Load

GCP Dataproc, Azure HDInsight, or Amazon EMR

Storage-tier

Database

Load Generation

Query

Response

Web-tier

HTTP Web Service

Query

Response

- Building a performant web service
- Dealing with large scale real world tweet data
- HBase and MySQL optimization

Query 2 - User Recommendation System

Use Case: When you follow someone on twitter, recommend close friends.

Query: GET /q2?**user_id**=<ID>&**type**=<TYPE>&**phrase**=<PHRASE>&**hashtag**=<HASHTAG>

Response:

<TEAMNAME>,<AWSID>\n
uid\tname\tdescription\ttweet\n
uid\tname\tdescription\ttweet

Three Scores:

- Interaction Score - closeness
- Hashtag Score - common interests
- Keywords Score - match specific interests

Final Score: Interaction Score * Hashtag Score * Keywords Score

Q2 target throughput: 10,000 RPS **for both MySQL and HBase**

Reminders on penalties

- M family instances **only**, smaller than or equal to **large** type
- Other types are allowed (e.g., t2.micro) **but only for testing**
    - Using these for any submissions = 100% penalty
- Only General Purpose (gp2) SSDs are allowed for storage
    - so **m5d is not allowed** since it uses NVMe storage
- AWS endpoints only (EC2/ELB).
- **$0.85/hour** applies to every submission

Phase 1 Budget

- Your web service should not cost more than **$0.85 per hour** this includes (see write-up for details):
  - EC2 cost (Even if you use spot instances, we will calculate your cost using the **on-demand** instance price)
  - **EBS cost**
  - **ELB cost**
  - We will not consider the cost of data transfer and EMR
- AWS total budget of $55 for Phase 1

Q2 Tips

- Start early! Start early! Start early!
- Consider doing ETL on GCP/Azure MySQL first
- Be careful about encoding 😒 (use utf8mb4 in MySQL)
- Use stable version of MySQL and HBase (use HBase 1.4.8)
- ETL can be expensive, so read the write-up carefully
- Pre-compute as much as possible

Spark, Scala and Zeppelin Primers

- Primers for Apache Spark/Scala/Zeppelin are now available
- You'll learn more about Spark in 3rd OPE, Project 4.1, and OLI Module 20 (which is a month away)
- Spark stores data in **memory**, allowing it to run an order of magnitude **faster** than Hadoop
- An alternative to Hadoop, but you have total freedom in ETL frameworks

Suggested Tasks for Phase 1

Phase 1 weeks	Tasks	Deadline
Week 1 ● **2/24 - 3/1**	● **Team meeting** ● **Writeup** ● **Complete Q1 code & achieve correctness** ● **Q2 Schema, think about ETL**	● **Q1 Checkpoint due on 3/1** ● **Checkpoint Report due on 3/1**
Week 2 ● 3/2 - 3/8	● **Q1 target reached** ● **Q2 ETL & Initial schema design completed**	● **Q1 final target due on 3/8**
Week 3 ● Spring Break	● Take a break or make progress (up to your team)	
Week 4 ● 3/16 - 3/22	● Achieve correctness for both Q2 MySQL, Q2 HBase & basic throughput	● Q2 MySQL Checkpoint due on 3/22 ● Q2 HBase Checkpoint due on 3/22
Week 5 ● 3/23 - 3/29	● Optimizations to achieve target throughputs for Q2 MySQL and Q2 HBase	● Q2 MySQL final target due on 3/29 ● Q2 HBase final target due on 3/29 ● Final Report due on 3/31

This Week's Deadlines

- Quiz 7:

  Due: **Thursday**, March 5th, 2020 11:59PM ET

- Complete Multi-Threading OPE task

  Due: This week (date varies)

- Project 3.2: Social Networking Timeline

  Due: Sunday, March 8th, 2020 11:59PM ET

- Team Project Phase 1 Q1 Final

  Due: Sunday, March 8th, 2020 11:59PM ET

Q&A