15-319 / 15-619
Cloud Computing

Recitation 7

Oct 8, 2019

Overview

- **Last week's reflection**
  - Project 2.3 and 3.1 (ongoing)
  - OLI Unit 3 - Module 10-12
  - Quiz 5
- **This week's schedule**
  - Project 3.1 and 3.2
  - OLI Unit 3 - Module 13
  - Quiz 6 (Due Friday 10/11)
  - Online Programming Exercise for Multi-Threading
- **Team Project, Twitter Analytics**
  - Phase 1 is out! Q1 final and Q2 checkpoint due on 10/20.
  - Phase 1 due, 10/27.

Last Week

- **Unit 3: Virtualizing Resources for the Cloud**
  - Module 10: Resource virtualization (Memory)
  - Module 11: Resource virtualization (I/O devices)
  - Module 12: Case Study
- **Quiz 5**
- **Project 2.3, Functions as a Service (FaaS)**
  - Task 1, Explore functions on various CSPs
    - Azure Functions, GCP Cloud Functions, AWS Lambda
  - Task 2, Extract thumbnails from video stream
    - Azure Functions and FFmpeg
  - Task 3, Get image labels and index
    - Azure Computer Vision, Azure Search

Last Week

- **Project 3.1**
  - Files v/s Databases (SQL & NoSQL)
    - Flat files
    - MySQL
    - HBase
      - Read the NoSQL and HBase basics primer

This Week

- **OLI : Unit 4 Module 14 - Cloud Storage**
- **Quiz 6** (Due Friday 10/11)
- **Project 3.2** - Sunday, March 13
  - Social Networking Timeline with Heterogeneous Backends
    - MySQL
    - Neo4j
    - MongoDB
    - Choosing Databases, Storage Types & Tail Latency
  - MongoDB Primer
- **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

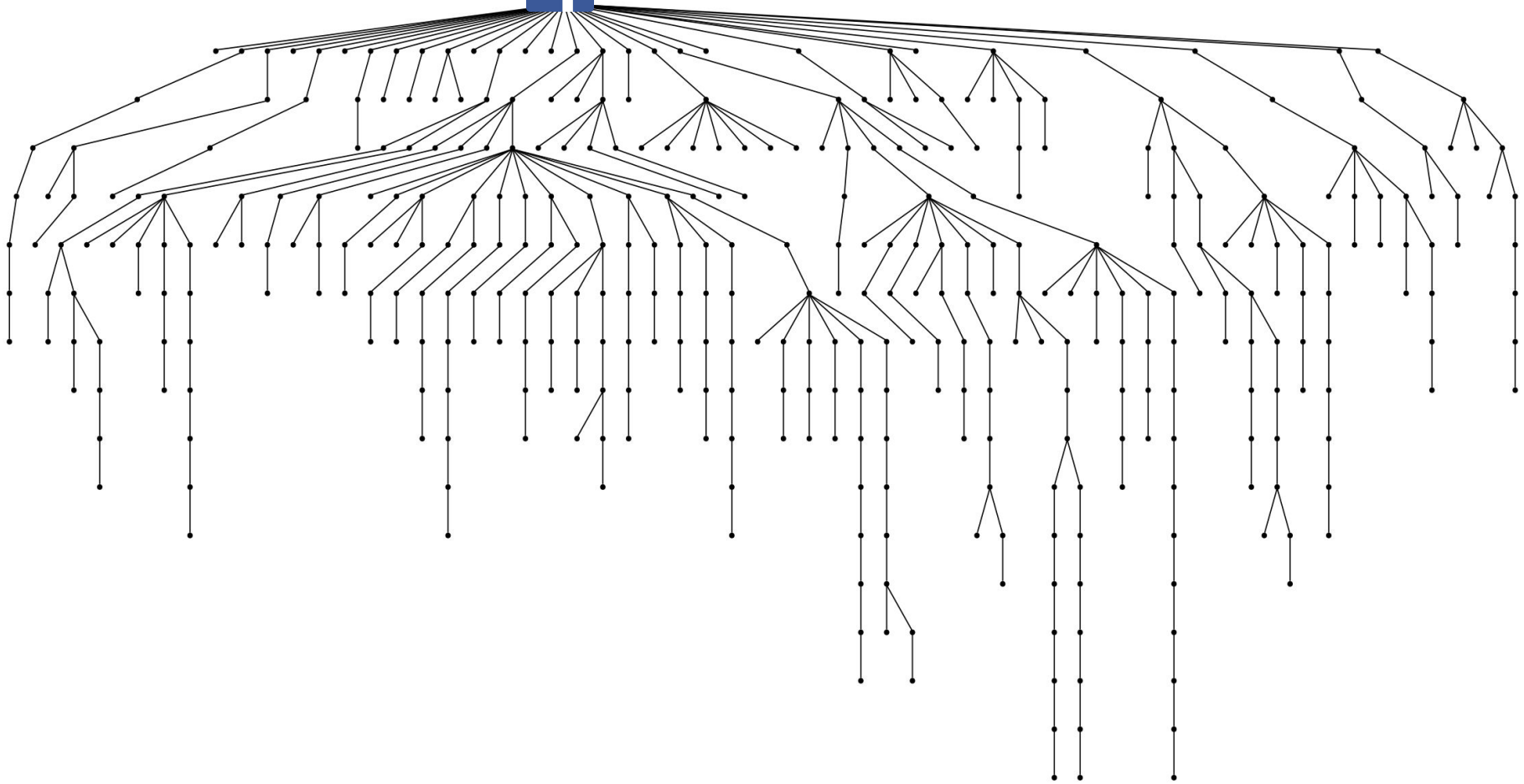
Individual Projects

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

A Social Network Service

High Fanout in Data Fetching

A single 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).

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
  - Geo-spatial queries

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: 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.

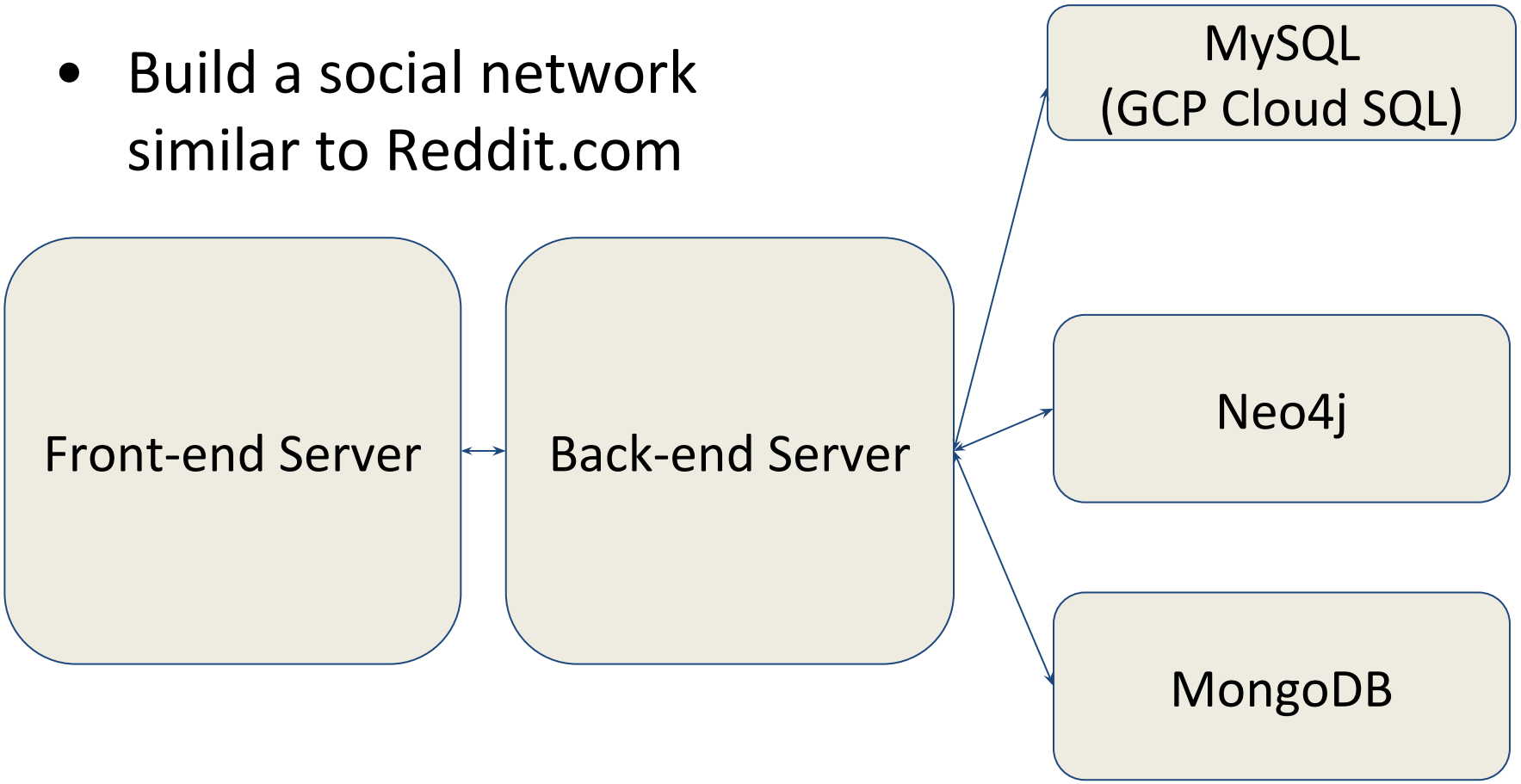
P3.2 - Reddit Dataset

- Task 1: User profiles
  - User authentication system : GCP Cloud SQL(users.csv)
  - User info / profile : GCP Cloud SQL
  - Follow TDD using Mockito
- 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

reddit

P3.2 - Architecture

- Build a social network similar to Reddit.com



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: Choosing Databases

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

P3.2 - Task 5

- **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 5

- **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
  - Mandatory for this project
- 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

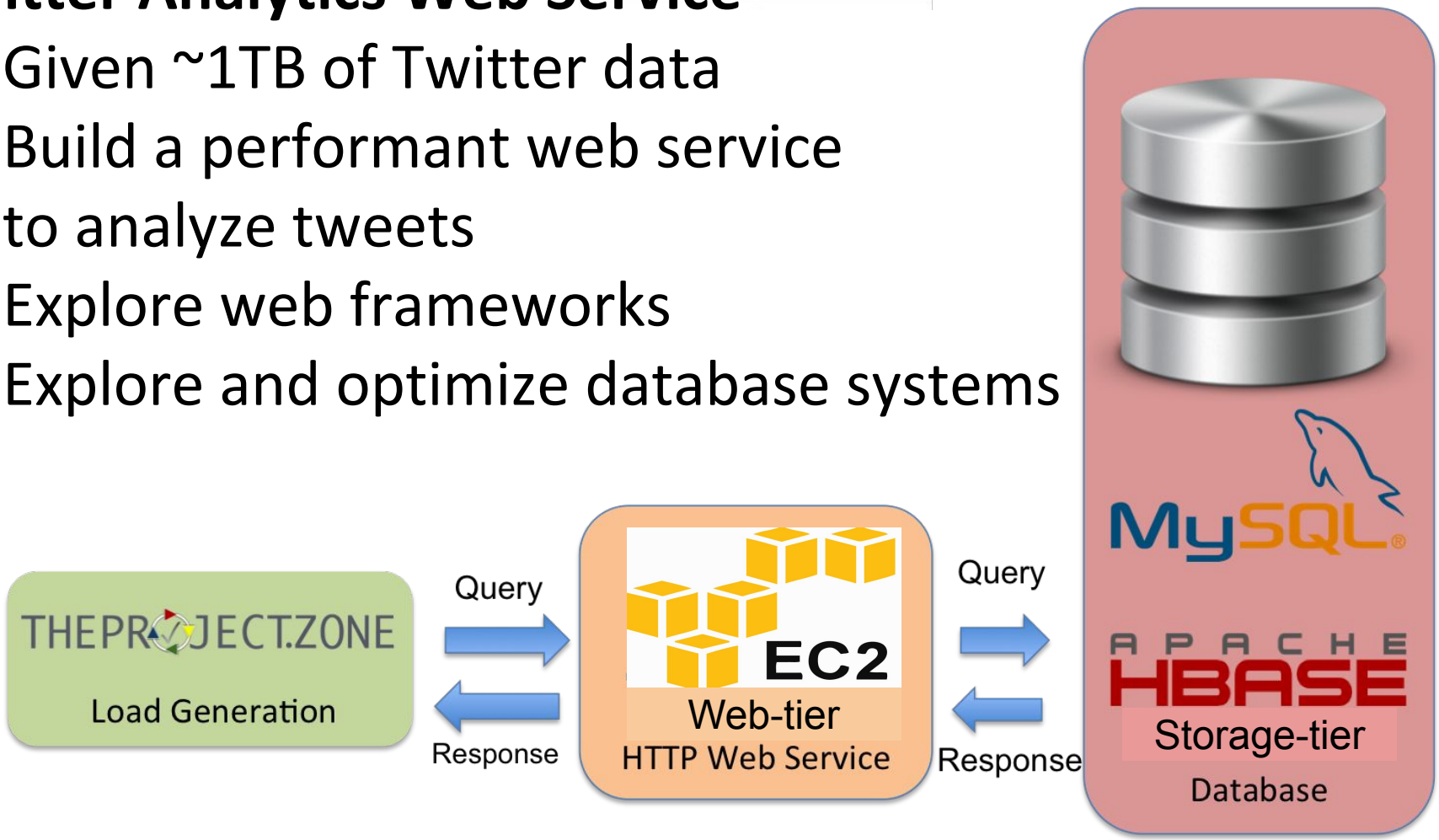
- In Task 4, you will use the databases from all previous tasks. Make sure to have **all** the databases loaded and ready when working on Task 4.
- You can submit one task at a time using the submitter. Remember to have your Back-end Server VM running when submitting.
- Make sure to terminate **all**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

TEAM PROJECT
Twitter Data Analytics

Team Project

Twitter Analytics Web Service

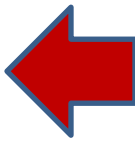
- Given ~1TB of Twitter data
- Build a performant web service to analyze tweets
- Explore web frameworks
- Explore and optimize database systems



Team Project

- Phase 1:
  - Q1
  - Q2 (MySQL **AND** HBase)

Input your team account ID and GitHub username on TPZ



- Phase 2
  - Q1
  - Q2 & Q3 (MySQL **AND** HBase)

- Phase 3
  - Q1, Q2, & Q3 (Managed Cloud Services)

Query 1 (CloudCoin)

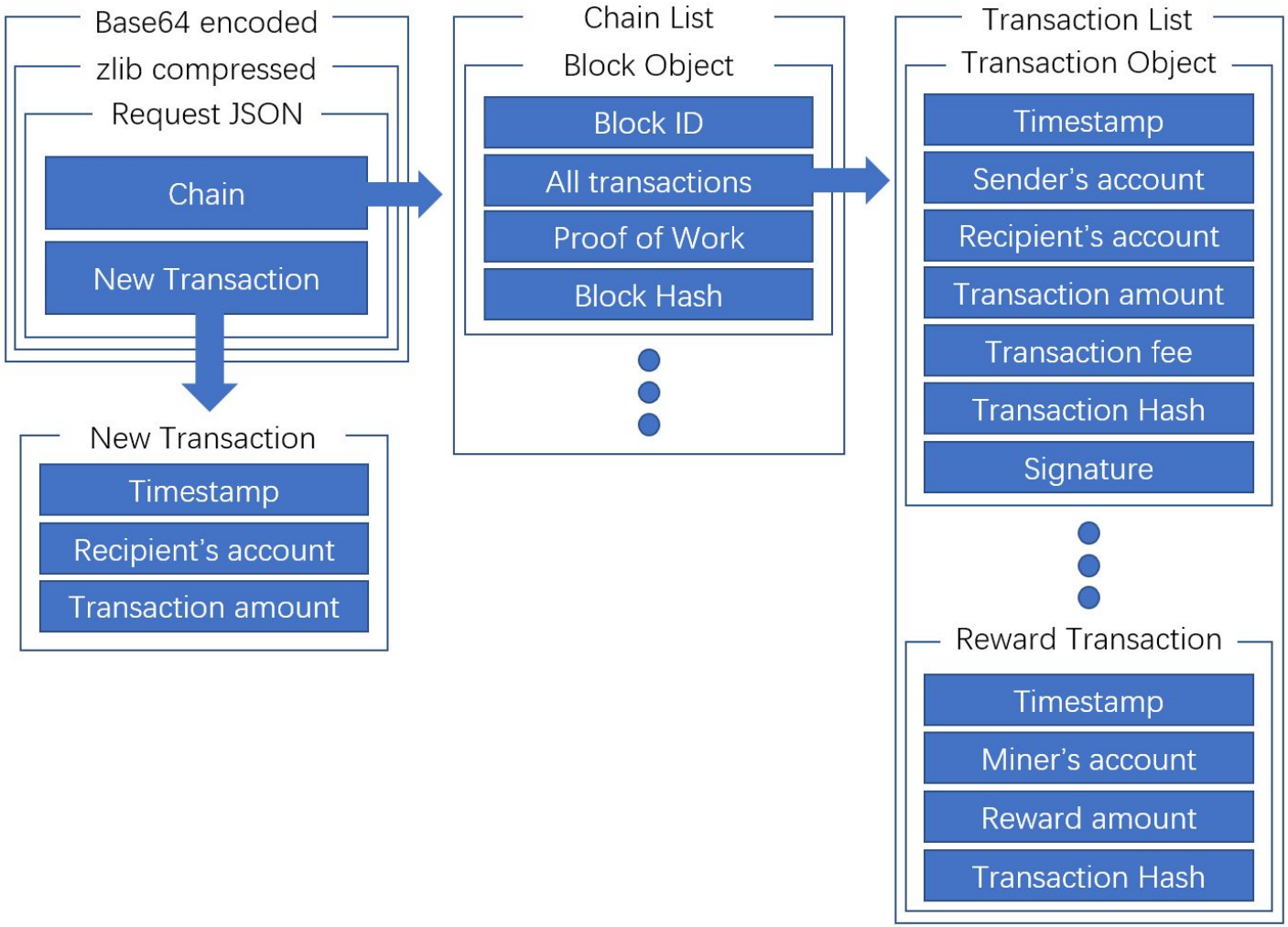
- Query 1 does not require a database (storage tier)
- Implement a web service that verifies and updates blockchains.
- You must explore different web frameworks
  - Get at least 2 different web frameworks working
  - Select the framework with the better performance
  - Provide evidence of your experimentations
  - Read the report first

What is a blockchain, though?

- Data structure that supports digital currency.
- Designed to be untamperable.
- Distributed. Shared among all user nodes.
  - Decentralized
  - Fault Tolerant.
- Consists of chained blocks.
- Each block consists of transactions.

Q1 Example

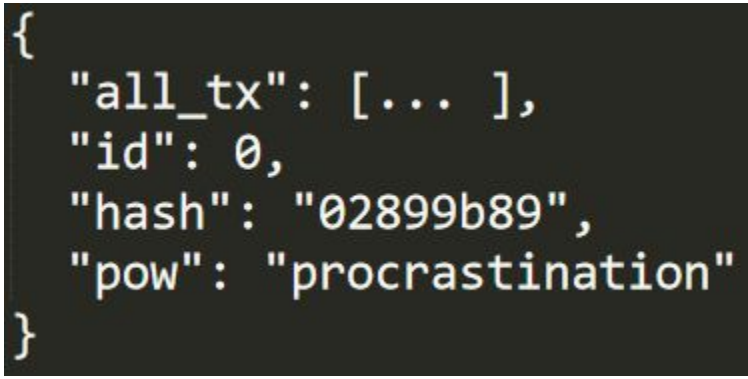
- Q1 input




```
{
  "chain": [
    {
      "all_tx": [
        {
          "recv": 509015179679,
          "amt": 500000000,
          "time": "1550721967779362304",
          "hash": "d50e5266"
        }
      ],
      "id": 0,
      "hash": "02899b89",
      "pow": "postpone"
    },
    {
      "all_tx": [
        {
          "send": 509015179679,
          "recv": 484054352161,
          "amt": 126848946,
          "fee": 12488,
          "time": "1550721967779391744",
          "hash": "5a2b4d71",
          "sig": 463884077351
        },
        {
          "recv": 1284110893049,
          "amt": 500000000,
          "time": "1550721967779424000",
          "hash": "7924c55e"
        }
      ],
      "id": 1,
      "hash": "0fce51c1",
      "pow": "fountain"
    },
    {
      "all_tx": [
        {
          "send": 1284110893049,
          "recv": 484054352161,
          "amt": 58759591,
          "fee": 5048,
          "time": "1550721967779447040",
          "hash": "b43737af",
          "sig": 1084970046728
        },
        {
          "recv": 34123506233,
          "amt": 500000000,
          "time": "1550721967779474176",
          "hash": "d705e74e"
        }
      ],
      "id": 2,
      "hash": "03635f77",
      "pow": "jeans"
    }
  ],
  "new_tx": {
    "recv": 837939704897,
    "amt": 430642077,
    "time": "1550721967779486720"
  }
}
```

Q1 Example

- Block:
  - Created by “miners”.
  - Has a list of transactions.
  - Block hash encapsulates all transaction info and block Metadata, as well as the hash of the previous block.
  - Block hash, required to start with a 0.
  - PoW (Proof of Work), which makes the hash start with a 0.
  - PoW is found by miner through brute forcing.



Q1 Example

- Transaction:
  - Hash value computed using all info in the blue box.
  - Signature is computed with hash value using RSA.
    sig=RSA(hash, key)


```
{
  "send": 1284110893049,
  "recv": 484054352161,
  "amt": 58759591,
  "fee": 5048,
  "time": "15507219677794470 40",
  "hash": "b43737af",
  "sig": 1084970046728
},
```

Q1 Example

- Reward:
  - Special type of transaction.
  - Created by miner.
  - Is the last transaction in the block's transaction list.


```
{
  "recv": 34123506233,
  "amt": 500000000,
  "time": "15507219677794741 76",
  "hash": "d705e74e"
}
```

Q1 Example

- New transaction:
  - You need to fill in missing fields.
  - You also need to sign the transaction using the key given to you.

```
"new_tx": {
  "recv": 837939704897,
  "amt": 430642077,
  "time": "15507219677794867 20"
}
```

Q1 Example

- Output:
  - Complete the new transaction.
  - Create a reward transaction.
  - Mine a new block that only has those two transactions.
  - Return the new transaction signature and new block PoW.
  - E.g. <125648413415l|i_love_cc>

Q1 Example

- Output:
  - There will be malicious attempts to break the blockchain.
  - You need to check the validity of the chain.
  - If the chain is not valid, return INVALID.
  - E.g. <INVALID|any_debug_info_you'd_like>

Query 2 - User Recommendation System

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

Three Scores:

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

Final Score: Interaction Score * Hashtag Score * Keywords Score

Query:

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

Response:

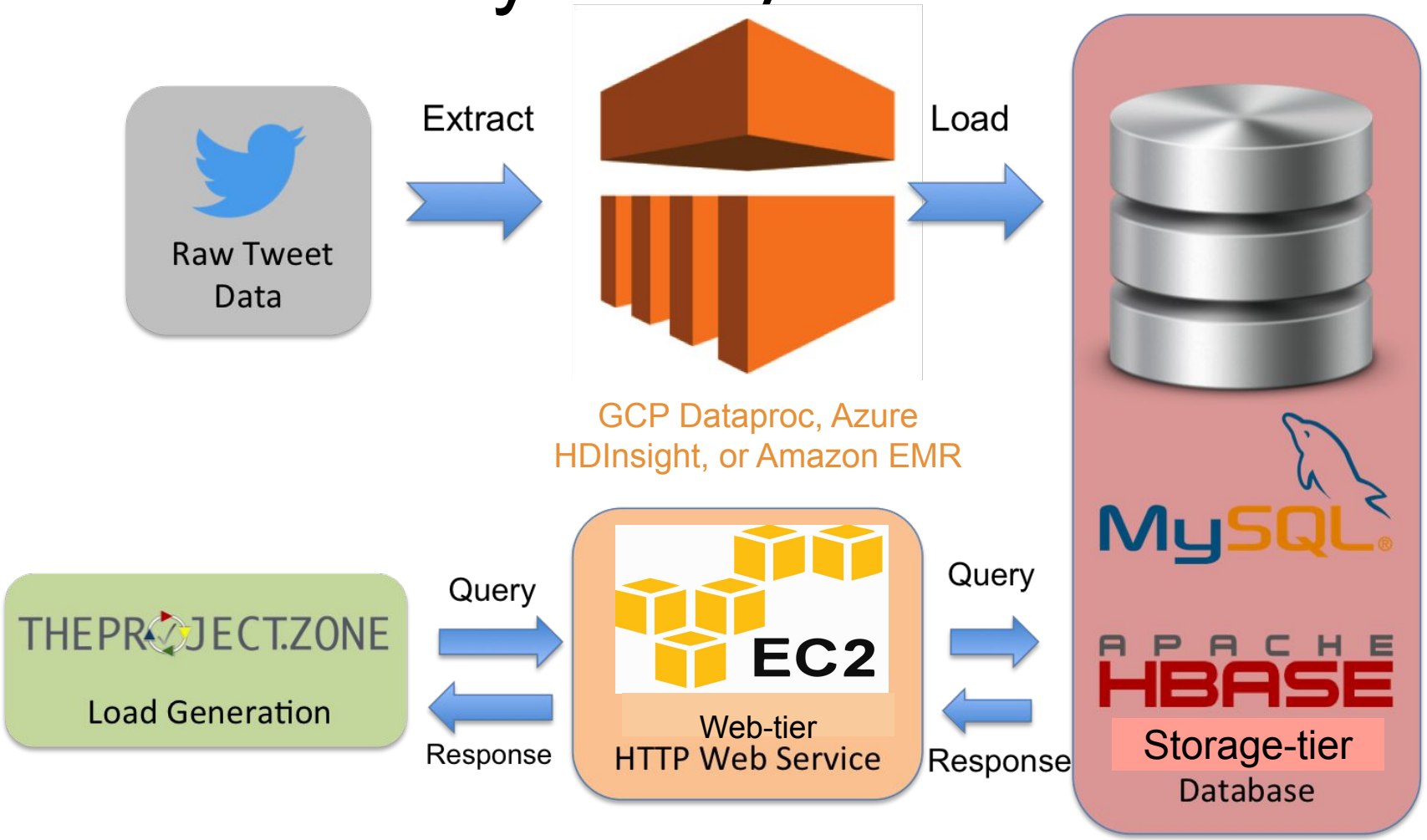
<TEAMNAME>,<AWSID>\n
uid\tname\tdescription\ttweet\n
uid\tname\tdescription\ttweet

Query 2 Example

```
GET /q2?
user_id=100123&
type=retweet&
phrase=hello%20cc&
hashtag=cmu

TeamCoolCloud,1234-0000-0001
100124\tAlan\tScientist\tDo machines think?\n
100125\tKnuth\tprogrammer\thello cc!
```

Twitter Analytics System Architecture



- Web server architectures
- Dealing with large scale real world tweet data
- HBase and MySQL optimization

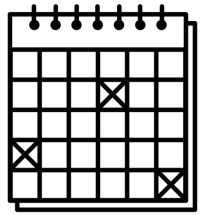
Git workflow

- Commit your code to the private repo we set up
  - Update your GitHub username in TPZ!
- Make changes on a new branch
  - Work on this branch, commit as you wish
  - Open a pull request to merge into the master branch
- Code review
  - Someone else needs to review and accept (or reject) your code changes
  - This process will allow you to capture bugs and remain informed on what others are doing

Heartwarming Tips from Your Beloved TAs

1. Design your architecture early and apply for limit increase.
2. EC2 VM is not the only thing that costs money.
3. Primers and individual projects are helpful.
4. You don’t need all your hourly budget to get Q1 target.
5. Coding is the least time consuming part.
6. Think before you do. Esp. for ETL (Azure, GCP, or AWS).
7. Divide workload appropriately. Take up your responsibility.
8. Read the write-up.
9. Read the write-up again.
10. Start early. You cannot make-up the time lost. Lots to finish.
11. I’m not kidding. Drama happens frequently.

Team Project Time Table

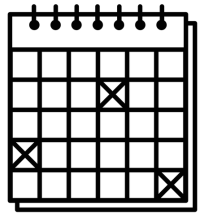


Phase	Deadline (11:59PM EST)
Phase 1 (20%) **- Query 1** **- Query 2**	• **Q1 CKPT (5%): Sun, 10/13** • **Report1 (5%): Sun, 10/13** • Q1 FINAL (10%): Sun, 10/20 • Q2 CKPT (10%): Sun, 10/20 • Q2M & Q2H FINAL (50%): Sun, 10/27 • Report2 (20%): Tue, 10/29
Phase 2 (30%) **- Add Query 3**	• Live Test on Sun, 11/11
Phase 3 (50%) **- Managed Services**	• Live Test on Sun, 12/02

Team Project Deadlines - Phase 1

- Writeup and queries were released on Monday.
- Phase 1 milestones:
  - Q1 Checkpoint: **Sunday, 10/13**
    - A successful 10-min submission for Q1
    - Checkpoint 1 Report
  - Q1 final due: **Sunday, 10/20**
    - Achieve the Q1 target
  - Q2 Checkpoint: **Sunday, 10/20**
    - A successful 10-min submissions:
      - Q2 MySQL **and** Q2 HBase.
  - Q2 final due: **Sunday, 10/27**
    - Achieve the Q2 target for Q2 MySQL **and** Q2 HBase.
  - Phase 1, code and report: **10/29**
- Start early, read the report and earn bonus points!

Suggested Tasks for Phase 1



Phase 1 weeks	Tasks	Deadline
Week 1 ● **10/7**	● **Team meeting** ● **Writeup** ● **Complete Q1 code & achieve correctness** ● **Q2 ETL & Initial schema design completed**	● **Q1 Checkpoint due on 10/13** ● **Checkpoint Report due on 10/13**
Week 2 ● 10/14	● Q1 target reached ● Achieve correctness for Q2 & basic throughput	● **Q1 final target due on 10/20** ● **Q2 checkpoint due on 10/20**
Week 3 ● 10/21	● Optimizations to achieve target throughputs for both Q2 MySQL and Q2 HBase	● Q2 MySQL final target due on 10/27 ● Q2 HBase final target due on 10/27

This Week's Deadlines

- Quiz 6: OLI Module 13

  Due: Friday, Oct 11th, 2019 11:59PM ET

- Project 3.1: Files v/s Databases

  Due: Sunday, Oct 13th, 2019 11:59PM ET

- Project 3.2: Social Networking Timeline with Heterogeneous Backends

  Due: Sunday, Oct 13th, 2019 11:59PM ET

- Team Project Phase 1 Q1 Checkpoint 1

  Due: Sunday, Oct 13th, 2019 11:59PM ET

Q&A