

# 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



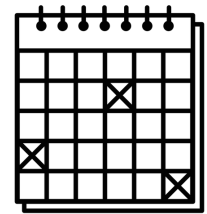
# Team Project

- Phase 1:
  - Q1
  - Q2 (MySQL **AND** HBase)
- Phase 2
  - Q1
  - Q2 & Q3 (MySQL **AND** HBase)
- Phase 3
  - Q1, Q2, & Q3 (Managed Cloud Services)

- Input your team account ID and GitHub username on TPZ

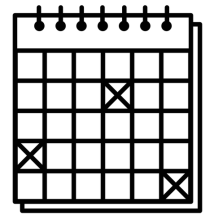


# Team Project Time Table



Phase	Deadline ( <u>11:59PM EST</u> )
<b>Phase 1 (20%)</b> <ul style="list-style-type: none"><li>- Query 1</li><li>- Query 2</li></ul>	<ul style="list-style-type: none"><li>● <b>Q1 CKPT (5%): Sun, 3/14</b></li><li>● <b>Q1 CKPT Report (5%): Sun, 3/14</b></li><li>● Q1 FINAL (10%): Sun, 3/21</li><li>● Q2 CKPT (10%): Sun, 3/21</li><li>● Q2M &amp; Q2H FINAL (50%): Sun, 3/28</li><li>● Final Report (20%): Tue, 3/30</li></ul>
<b>Phase 1 (20%)</b> <ul style="list-style-type: none"><li>- Query 1</li><li>- Query 2</li></ul>	<b>BONUSES:</b> <ul style="list-style-type: none"><li>● <b>Q1 Early Bird Bonus (5%): Sun, 3/14</b></li><li>● Q2M &amp; Q2H Early Bird Bonus (5%): Sun, 3/21</li><li>● Q2 Correctness Penalty Waiver: Sun, 3/21</li><li>● Q2M &amp; Q2H Performance Bonus (5%): Sun, 3/28</li></ul>

# Team Project Time Table



<b>Phase</b>	<b>Deadline (<u>11:59PM EST</u>)</b>
<b>Phase 2 (30%)</b> - <b>Add Query 3</b>	<ul style="list-style-type: none"><li>● Live Test on Sun, 4/18</li></ul>
<b>Phase 3 (50%)</b> - <b>Managed Services for Query 1-3</b>	<ul style="list-style-type: none"><li>● Live Test on Sun, 5/2</li></ul>

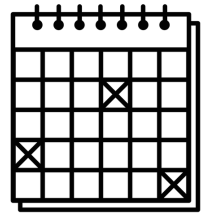
# Team Project Deadlines - Phase 1

- Writeup and queries released on Monday 3/8.
- Phase 1 milestones:
  - Q1 Checkpoint: **Sunday, 3/14**
    - A successful 10-min submission for Q1
  - Query 1 Checkpoint report: **Sunday, 3/14**
  - Q1 Final due: **Sunday, 3/21**
    - Achieve the Q1 target
  - Q2 Checkpoint: **Sunday, 3/21**
    - A successful 10-min submissions:
      - Q2 MySQL and Q2 HBase.
  - Q2 Final due: **Sunday, 3/28**
    - Achieve the Q2 target for Q2 MySQL and Q2 HBase.
  - Phase 1, code and report: **Tuesday, 3/30**

# Team Project Deadlines - Phase 1

- Phase 1 **bonus milestones**:
  - [5%] Q1 Early Bird Bonus: **Sunday, 3/14**
    - Reach target 32000 for Q1 (10-min submission)
    - Fill in the checkpoint report
  - [5%] Q2 Early Bird Bonus: **Sunday, 3/21**
    - Reach target 10000 for Q2 (10-min submission)
      - Q2 MySQL **and** Q2 HBase
  - [Waive Penalty] Q2 Correctness bonus: **Sunday, 3/21**
    - One 10-min submission with correctness above 95%
  - [5%] Q2 Performance bonus: **Sunday, 3/21**
    - Reach target 15000 for Q2 (10-min submission)
      - Q2 MySQL **and** Q2 HBase.
- **Start early**, read the report and earn **bonus** points!

# Suggested Tasks for Phase 1



Phase 1 weeks	Tasks	Deadline
<b>Week 1</b> <ul style="list-style-type: none"> <li>3/8 - 3/14</li> </ul>	<ul style="list-style-type: none"> <li>Team meeting</li> <li>Read Write Up &amp; Report</li> <li>Complete Q1 code &amp; achieve correctness</li> <li>Start ETL on mini dataset and design q2 schema</li> </ul>	<ul style="list-style-type: none"> <li><b>Q1 Checkpoint due on 3/14</b></li> <li><b>Checkpoint Report due on 3/14</b></li> </ul>
<b>Week 2</b> <ul style="list-style-type: none"> <li>3/15 - 3/21</li> </ul>	<ul style="list-style-type: none"> <li>Q1 target reached</li> <li>Q2 ETL &amp; Initial schema design completed</li> <li>Achieve Q2 basic correctness and submit to TPZ</li> </ul>	<ul style="list-style-type: none"> <li>Q1 final target due on 3/21</li> <li>Q2 MySQL Checkpoint due on 3/21</li> <li>Q2 HBase Checkpoint due on 3/21</li> </ul>
<b>Week 3</b> <ul style="list-style-type: none"> <li>3/22-3/28</li> </ul>	<ul style="list-style-type: none"> <li>Achieved correctness for both Q2 MySQL, Q2 HBase &amp; basic throughput</li> <li>Optimizations to achieve target throughputs for Q2 MySQL and Q2 HBase</li> </ul>	<ul style="list-style-type: none"> <li>Q2 MySQL final target due on 3/28</li> <li>Q2 HBase final target due on 3/28</li> <li>Final Report due on 3/30</li> </ul>



# Query 1 - CloudCoin

**Submission Budget: \$0.70/h**

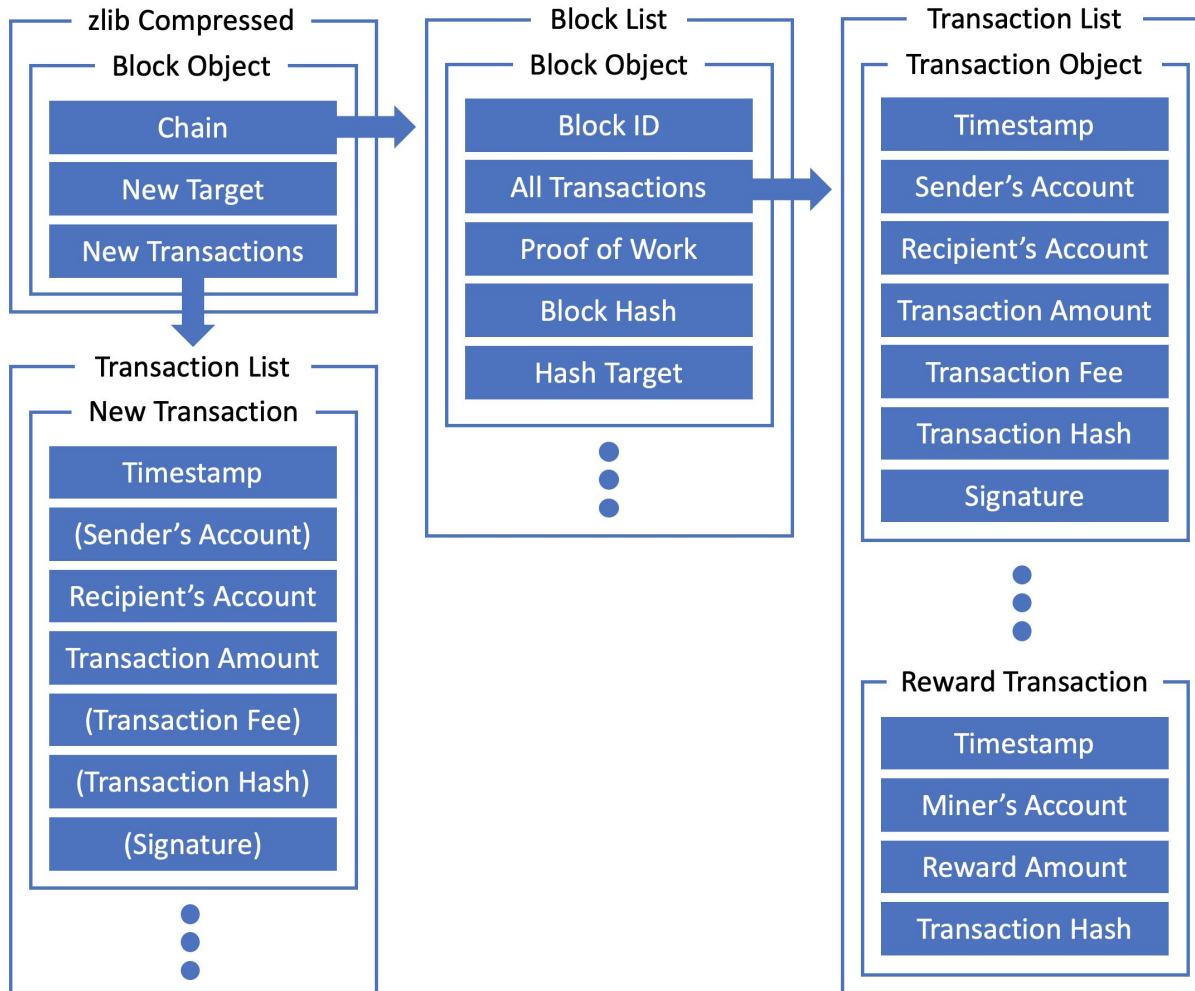
- Query 1 does not require a database (storage tier)
- Implement a web service that verifies and updates blockchains.
- You must explore different web frameworks
  - Get at least 2 different web frameworks working
  - [Techempower](#) will be a good resource to explore
  - Select the framework with the better performance
  - Provide evidence of your experimentations
  - Have support libraries to connect to mysql and Hbase
  - **Read the report template 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": [
        {
          "rec": 895456882897,
          "amt": 50000000,
          "time": "158252040000000000",
          "hash": "4b277860"
        }
      ],
      "pow": "0",
      "id": 0,
      "hash": "07c98747",
      "target": "1"
    },
    {
      "all_tx": [
        {
          "sig": 1523500375459,
          "rec": 831361201829,
          "fee": 2408,
          "amt": 126848946,
          "time": "1582520454597521976",
          "send": 895456882897,
          "hash": "c0473abd"
        }
      ],
      {
        "rec": 621452032379,
        "amt": 50000000,
        "time": "1582521002184738591",
        "hash": "ab56f1d8"
      }
    },
    {
      "pow": "202",
      "id": 1,
      "hash": "0055fd15",
      "target": "01"
    },
    {
      "all_tx": [
        {
          "sig": 829022340937,
          "rec": 905790126919,
          "fee": 78125,
          "amt": 4876921,
          "time": "1582521009246242025",
          "send": 831361201829,
          "hash": "46b61f8e"
        }
      ],
      {
        "sig": 295281186908,
        "rec": 1097844002039,
        "fee": 0,
        "amt": 83725981,
        "time": "1582521016852310220",
        "send": 895456882897,
        "hash": "b6c1b10f"
      }
    },
    {
      "rec": 905790126919,
      "amt": 250000000,
      "time": "158252160302667063",
      "hash": "b0750555"
    }
  ],
  "pow": "12",
  "id": 2,
  "hash": "00288a38",
  "target": "0a"
},
{
  "new_target": "007",
  "new_tx": [
    {
      "sig": 160392705122,
      "rec": 658672873303,
      "fee": 3536,
      "amt": 34263741,
      "time": "1582521636327155516",
      "send": 831361201829,
      "hash": "1fb48c71"
    }
  ],
  {
    "rec": 895456882897,
    "amt": 34263741,
    "time": "1582521645744862608"
  }
}
}

```

# Q1 Example

- **Q1 input:**
  - Several blocks, each with numerous transactions
  - A new transaction to be processed by you
- **Q1 output:**
  - Check each block and transaction for validity
  - If valid, find a POW that satisfy the hash requirement.
  - Add the completed transaction and new block into the blockchain.

(You could verify your understanding using our reference server)

# 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, plus a PoW chosen by the miner.

- Miner finds a PoW (Proof of Work) through brute forcing, to make the block hash lexicographically smaller than the hash target.
- Block hash formula:

```
{  
  "all_tx": [...],  
  "pow": "cloud",  
  "id": 2,  
  "hash": "09288a38",  
  "target": "0a"  
}
```

```
CCHash(SHA-256("block_id|previous_block_hash|tx1_hash|tx2_hash|tx3_hash...") + PoW)
```

# Q1 Example

- **Transfer Transaction:**

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

```
{  
  "send": 831361201829,  
  "recv": 905790126919,  
  "amt": 4876921,  
  "fee": 78125,  
  "time": "1582521009246242025",  
  "sig": 829022340937,  
  "hash": "46b61f8e"  
},
```

```
CCHash("timestamp|sender|recipient|amount|fee")
```

# Q1 Example

- **Reward Transaction:**

- Special type of transaction.
- Created by miner.
- Is the last transaction in
  - the block's transaction list.
- Reward amount determined by block id, 500,000,000 for the first two blocks, halved for any two following blocks.

```
{  
  "recv": 905790126919,  
  "amt": 250000000,  
  "time": "1582521603026667063",  
  "hash": "b0750555"  
}
```

# Q1 Example

- **New transactions:**
  - Contains transactions made by your team or by some other accounts.
  - Transaction made by some other account has the same format as any non-reward transaction in the block list.
  - For the transactions made by your team, you need to fill in missing fields and sign it using the key given to you.

```
"new_tx": [  
  {  
    "sig": 160392705122,  
    "recv": 658672873303,  
    "fee": 3536,  
    "amt": 34263741,  
    "time": "1582521636327155516",  
    "send": 831361201829,  
    "hash": "1fb48c71"  
  },  
  {  
    "recv": 895456882897,  
    "amt": 34263741,  
    "time": "1582521645744862608"  
  }  
]
```

# Q1 Example

- **Q1 Output:**
  - Collect the new transactions.
  - Create a reward transaction.
  - Include these transactions in a new block.
  - Compute a PoW that makes the new block hash satisfies the new hash target.
  - Append the block to the chain.
  - Respond with the zlib compressed and Base64 encoded new JSON.

# Q1 Example

- **Q1 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 a string that starts with INVALID.
  - You can append any debug info you want. Just make sure it does not start a new line.
  - E.g., INVALID|any\_debug\_info\_you\_like

# Query 2 - User Recommendation System

**Submission Budget: \$0.70/h MySQL, \$0.85/h HBase**

**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>\nuid\tname\tdescription\ttweet\nuid\tname\tdescription\ttweet
```

# Q2 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!
```

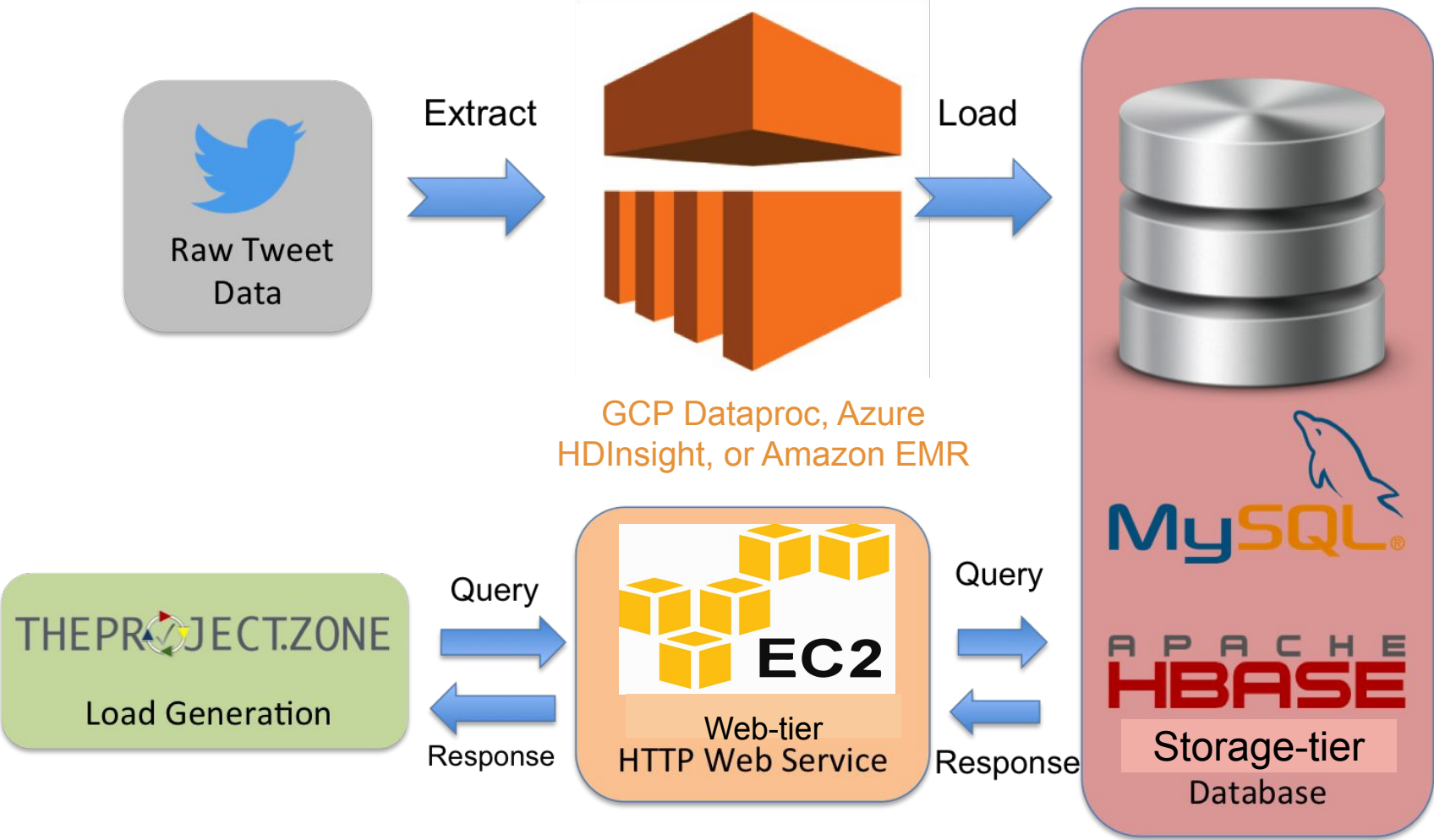
# Q2 ETL Recommendations

- Do the filtering on the first part of the dataset and make sure the result is **exactly the same** as the reference answer. (See **Reference the ETL result of a small dataset** at the end of the Query 2 write-up.)
- Start ETL on the mini dataset in GCP/Azure, pick some queries as the test cases and compare your result against the mini reference server (See **Phase1 - [Reference Server](#)** in the write-up)
  - You can also test locally for your result (See **Continuous testing** section)

# Phase 1 Budget

- Your web service should not cost more than **\$0.70/hour (Q1 and Q2 MySQL)** and **\$0.85/hour (Q2 HBase)** this includes:
  - 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
  - See writeup for details
- AWS total budget of \$55 for Phase 1

# 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
  - Make sure your final phase 1 code and reports are in **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. Different instance types requires different AMI.
4. Primers and individual projects are helpful.
5. You don't need all your hourly budget to get Q1 target.
6. Coding is the least time consuming part.
7. Think before you do esp. for ETL (Azure, GCP, or AWS).
8. Divide workload appropriately. Take up your responsibility.
9. Read the write-up.
10. Read the write-up again.
11. Read the report.
12. **Start early.** You cannot make-up the time lost
13. I'm not kidding. Drama happens frequently.
14. Don't forget to use the [reference server](#) to verify you result!

Good Luck!!!

