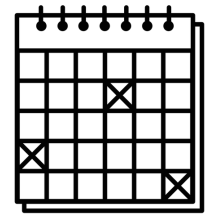


TEAM PROJECT

Twitter Data Analytics

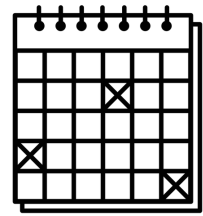


Team Project Time Table



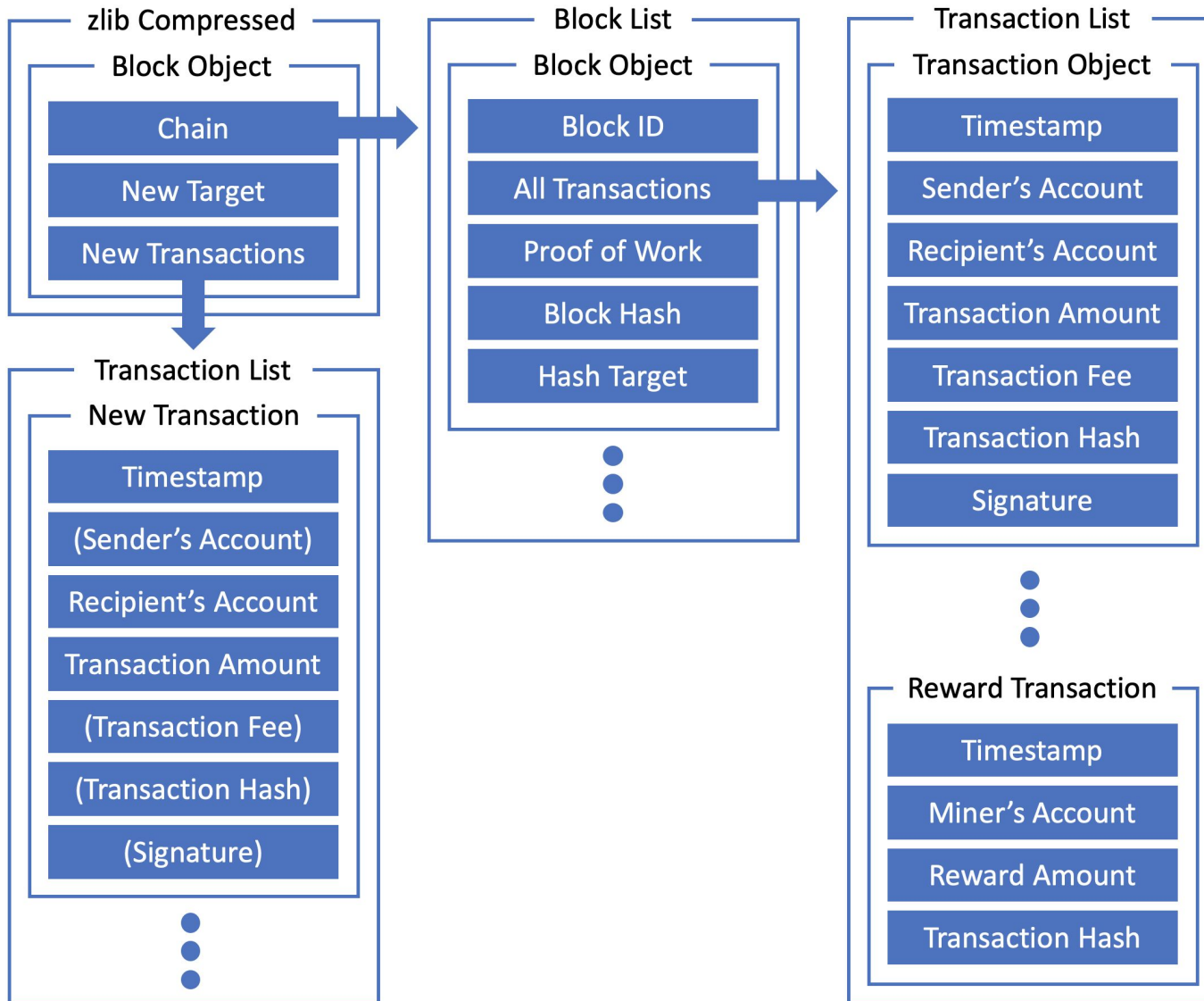
Phase	Deadline (<u>11:59PM EST</u>)
Phase 1 (20%) <ul style="list-style-type: none">- Query 1- Query 2	<ul style="list-style-type: none">● Q1 CKPT (5%): Sun, 3/14● Q1 CKPT Report (5%): Sun, 3/14● Q1 FINAL (10%): Sun, 3/21● Q2 CKPT (10%): Sun, 3/21● Q2M & Q2H FINAL (50%): Sun, 3/28● Final Report (20%):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

Team Project Time Table



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

Query 1 Recap

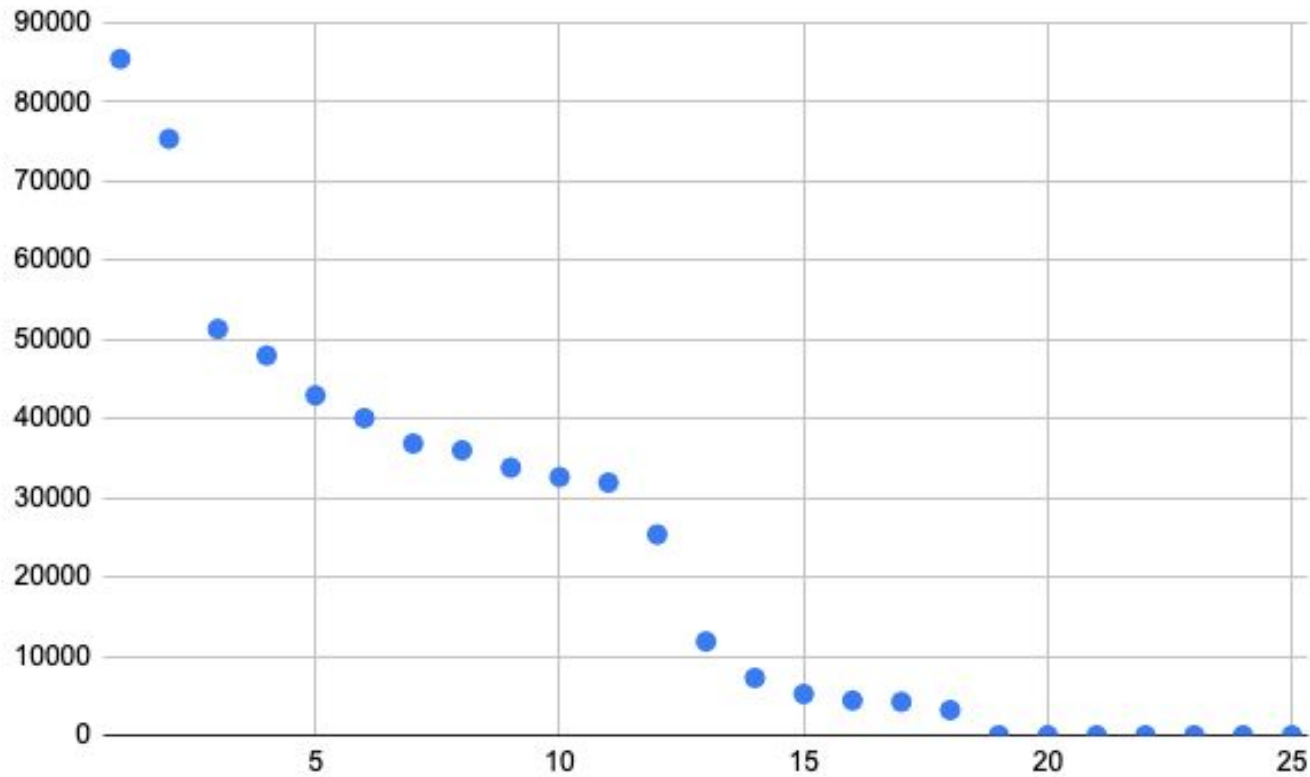


```

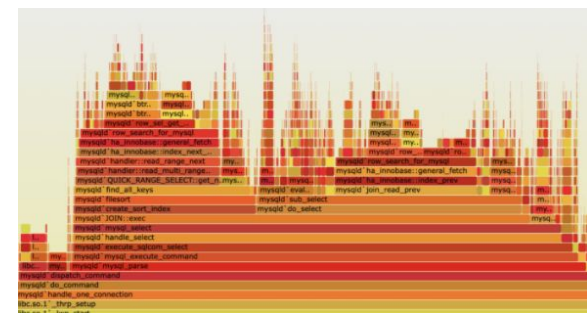
{
  "chain": [
    {
      "all_tx": [
        {
          "recv": 895456882897,
          "amt": 50000000,
          "time": "1582520400000000000",
          "hash": "4b277860"
        }
      ],
      "pow": "0",
      "id": 0,
      "hash": "07c98747",
      "target": "1"
    },
    {
      "all_tx": [
        {
          "sig": 1523500375459,
          "recv": 831361201829,
          "fee": 2408,
          "amt": 126848946,
          "time": "1582520454597521976",
          "send": 895456882897,
          "hash": "c0473abd"
        },
        {
          "recv": 621452032379,
          "amt": 50000000,
          "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": "1582521645744862688"
    }
  ]
}
  
```

Team Project - Q1 CKPT1

- 25 teams attempted a 600s submission for Query 1
- 10 teams reached 32,000 RPS and scored Early Bird Bonus



Q1 Tips



Flamegraph

If the throughput is lower than expected

- Try [m6g](#) instances
- Identify the bottleneck with a profiler
 - TPZ Primer: [Profiling a Cloud Service](#)
 - You may also find it convenient to profile with your IDE locally
- Change to another package, framework or even language

If the correctness is lower than expected

- Read the writeup more carefully

Read Query 2 Now. Start ETL Now.

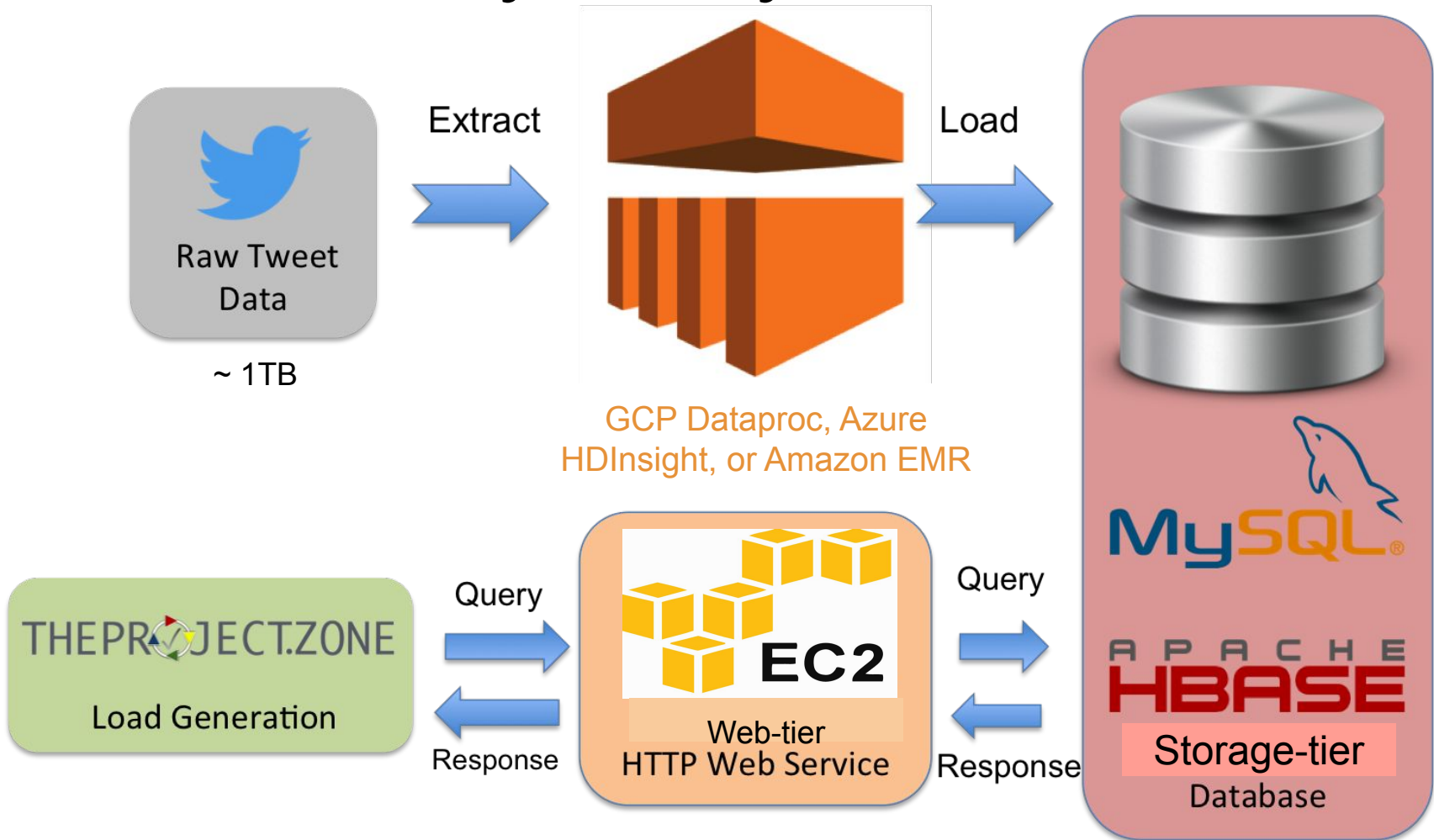
Value	Target RPS	Weight	Due date (at 11:59PM EST)
Query 1 Checkpoint	-	5%	Sunday, March 14th
Checkpoint Report	-	5%	Sunday, March 14th
Query 1 Early Bird Bonus	32,000	5%	Sunday, March 14th
Query 1 Final	32,000	10%	Sunday, March 21st
Query 2 Checkpoint	-	10%	Sunday, March 21st
Query 2 Correctness Bonus	-	Waive one most significant penalty for each team member	Sunday, March 21st
Query 2 Early Bird Bonus	10,000	5%	Sunday, March 21st
Query 2 Final	10,000	50%	Sunday, March 28th
Query 2 Performance Bonus	15,000	5%	Sunday, March 28th
Final Report + Code	-	20%	Tuesday, March 30th

You have only one week to meet the Query 2 checkpoint

Question: Is one(1) weekend enough time for Q2?

Hint: No. Start now.

Twitter Analytics System Architecture



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



Query 2 - User Recommendation System

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

Use Case: Recommend users you may also be interested in when you follow someone on twitter

Three Scores when making recommendation:

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

Query 2 - Filtering

Each line from the provided files is a tweet object

- Malformed JSON Object
- Malformed Tweets
 - Make sure each tweet you keep has valid tweet id, sender's user id, timestamp, content and at least 1 hashtag
- Tweets not in the required languages
- Duplicate Tweets

Query 2 - Contact Tweets

Given a valid tweet JSON object `t`.

A **contact tweet** is a tweet that is either a reply tweet or a retweet.

- A tweet is a **reply** tweet if `t.in_reply_to_user_id` is not null.
- A tweet is a **retweet** if `t.retweeted_status` is not null.

tweet_id	user_id	content	reply_to_id	retweet_to_id
01	15618	cloud	15213	null
02	15640	computing	null	15319
03	15513	is	15213	null
04	15513	fun	null	null

Then we have the followings:

user_id	contact_tweet_id	contacted_user
15213	01, 03	15618, 15513
15513	03	15213
15319	02	15640
15618	01	15213
15640	02	15319

Query 2 - User Information

Given a valid tweet JSON object `t`.

User information can appear in `t` and `t.retweeted_status` objects.

- For any tweet `t`, we can find the sender information in `t.user`
- If the tweet `t` happens to be a **retweet**, we can additionally find the original poster's information in `t.retweeted_status.user`

For each user appeared, we can get the timestamp from `t.created_at`.

After processing all the valid tweets, we can get the latest information of all users.

Note: For user information with the same timestamp, break the tie by tweet ID in **descending numerical order**.

Query 2 - Interaction Score

There are two types of interactions: reply and retweet. The more replies and retweets between two users, the higher their interaction score. **Interaction score** is calculated as $\log(1 + 2 * \text{reply_count} + \text{retweet_count})$

Some examples are below:

1. A replied B 4 times; B retweeted A 3 times $\log(1 + 2*4 + 1*3) = 2.485$
2. A replied B twice; B replied A once $\log(1 + 2*(2+1) + 1*0) = 1.946$
3. A retweeted B once $\log(1 + 2*0 + 1*1) = 0.693$
4. no replies/retweets between A and B $\log(1 + 2*0 + 1*0) = 0$

Query 2 - Hashtag Score

Same hashtag count is calculated by counting hashtags among all the tweets two users **posted**, excluding popular hashtags from the list provided by us.

The final `hashtag_score` is calculated as follows.

- If `same_tag_count > 10`, then `hashtag_score = 1 + log(1 + same_tag_count - 10)`.
- Else, `hashtag_score = 1`

Here are a few examples. Assume hashtag `zipcode` is a very popular hashtag that we exclude.

Note: For the cases of self-reply or self-retweet (the reply or retweet is to the same user of the original tweet), the **hashtag score will always be 1**.

Note: hashtags are case-insensitive

```
| sender_uid | hashtags |
| ----- | ----- |
| 15619      | Aws, azure, ZIPCODE |
| 15619      | Cloud, Azure |
| 15619      | Cloud, GCP |
| 15619      | cloud, aws |
| 15319      | cmu, us |
| 15319      | AZure |
| 15319      | Cloud, GCP |
| 15319      | aWs, zipcode, CLOUD |
| 15513      | cmu, us |
| 15513      | haha, ZIPcode |
| 15513      | zipcode |
```

Given all the tweets above, the hashtag score of the user pairs below are:

```
| uid_1 | uid_2 | same_tag_count | explanation |
| ---- | ---- | - | - |
| 15619 | 15319 | 13 | aws=3, cloud=5, azure=3, GCP=2 |
| 15619 | 15513 | 0 | no match |
| 15319 | 15513 | 4 | cmu=2, us=2 |
```

Query 2 - Keyword Score

Keywords score is calculated by counting the total number of matches of phrase and also hashtag (both provided in the query) across the **contact tweets** of a specific *type*. The *type* is given in the query, and valid values are `[reply|retweet|both]`.

Matching rule for the phrase: case sensitive match.

Another example, for phrase `haha`

- `hahaha` has 2 matches (overlapping matches are possible)
- `haHaha` has no matches
- `Haha bahaha` has 1 match

Matching rule for the hashtag: case insensitive exact match.

For example, if `hashtag` in the request is `cloud`, and a tweet has hashtags `#Cloud #CLOUD #CLOUD #cmu` (note that duplicate tags are allowed), then this tweet will add 3 to `number_of_matches`.

Between two users, if there are no contact tweets of the type specified in the query, then `keywords_score = 0`.

Otherwise, `keywords_score = 1 + log(number_of_matches + 1)`.

Query 2 - Final Score and Ordering

Final Score

The final ranking score between two users is calculated as

```
final_score = interaction_score * hashtag_score * keywords_score
```

- keep 5 decimal points of precision for the final score rounding half up **before** ranking.
- ignore user pairs with a final score of 0.

Ordering

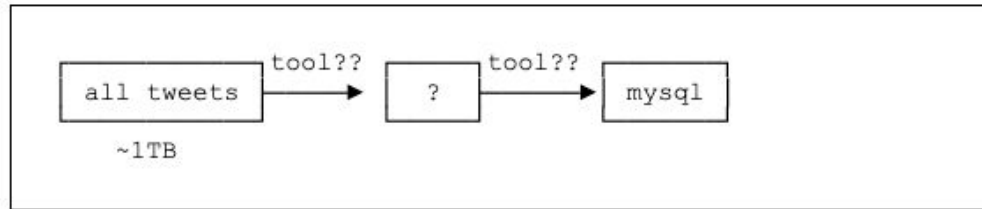
Your web service should return the most up-to-date information of users and their latest contact tweet with the user.

- **Rank by the score in descending order.**
 - Break ties by user ID in **descending numerical order.**

For the latest contact tweets between two users, break the tie by tweet ID in **descending numerical order** if they have the same timestamp.

Q2 Roadmap

- Use a flowchart as an ETL mind map and code design



- Do the filtering on the first part (part-00000) of the dataset and make sure the result is **exactly the same** as the reference answer
- Start ETL on the mini dataset locally or in GCP/Azure with sufficient unit tests
 - Think about what information is necessary for one query
 - Start with a tentative schema and adjust it accordingly
 - Pick some test queries that can help you partially verify for ETL process
 - While loading to MySQL may be more familiar, start loading to HBase in parallel to get familiar with it
 - Make use of the mini-ref server

Continued

Q2 Roadmap (continued)

- Start ETL on the entire dataset in GCP/Azure and compare your result against the reference server.
 - E - T - L - Verify with reference server
 - Spot the bug, correct the code and rerun the ETL
 - Can you just rerun the ETL partially?
 - Make multiple 600s submissions if you think it's good enough
- Optimize your implementation to reach the target throughput.
 - Identify the bottleneck with profiling
 - Web framework? Query Processing Logic? DB Schema?
 - If you decide to change the schema (you're very likely **need** to)
 - what part of ETL you need to rerun?
 - If loading all the data into MySQL takes X hours, HBase takes Y hours, how many interaction can you do?
 - Can you accelerate your design iteration?

How to help TA to help you

- Hint: hint won't come from nowhere.
 - The more context you provide, the more we can understand your situation, the more accurate help we can offer
- Generate context: see piazza @6 and @9
- Context for asking correctness improvement:
 - A checklist, flowchart, mind map describing your understanding
 - Measurements you've taken to check each item in the checklist/flowchart/mindmap
 - For example, "I've wrote unit tests for everything in my checklist"

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
 - e.g **m5d is not allowed** since it uses NVMe storage
- AWS endpoints only (EC2/ELB).
- **\$0.70/hour (MySQL)** and **\$0.85/hour (HBase)** applies to every submission

Phase 1 Budget

- Your web service should not cost more than **\$0.70/hour (Q1 and Q2 MySQL)** and **\$0.85/hour (Q2 HBase)**, which includes:
 - EC2 cost (Even if you use spot instances, we will calculate your cost using the **on-demand** instance price)
 - **EBS cost**
 - **ELB cost (excludes LCU)**
 - We will not consider the cost of data transfer and EMR
 - See writeup for details
- AWS total budget of \$55 for Phase 1

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
- Spark stores data in **memory**, allowing it to run an order of magnitude **faster** than Hadoop
- Spark is **more expressive** for some operations (e.g. join with two keys)
- You can use Spark or Hadoop - it is your choice since you have total freedom in ETL frameworks



Good luck!