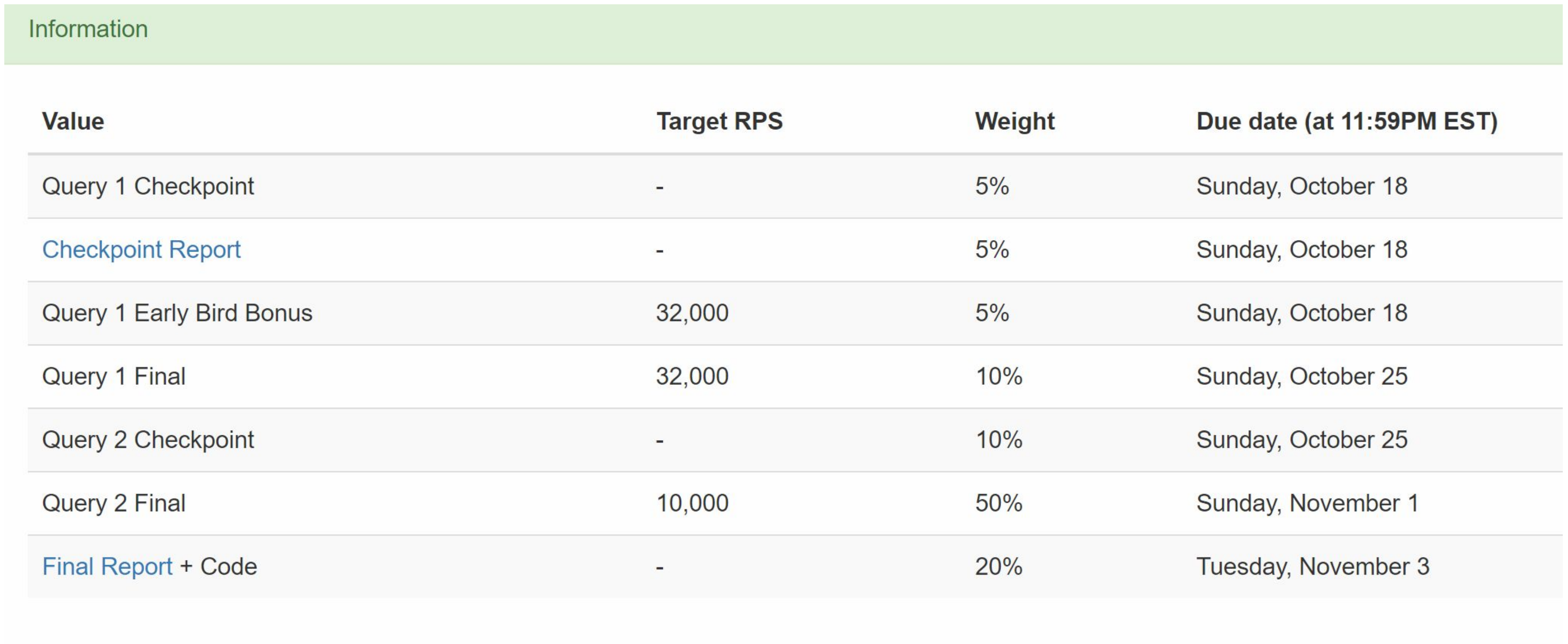
TEAM PROJECT
Twitter Data Analytics

Read about Query 2 Now.
Start ETL Now.



Information

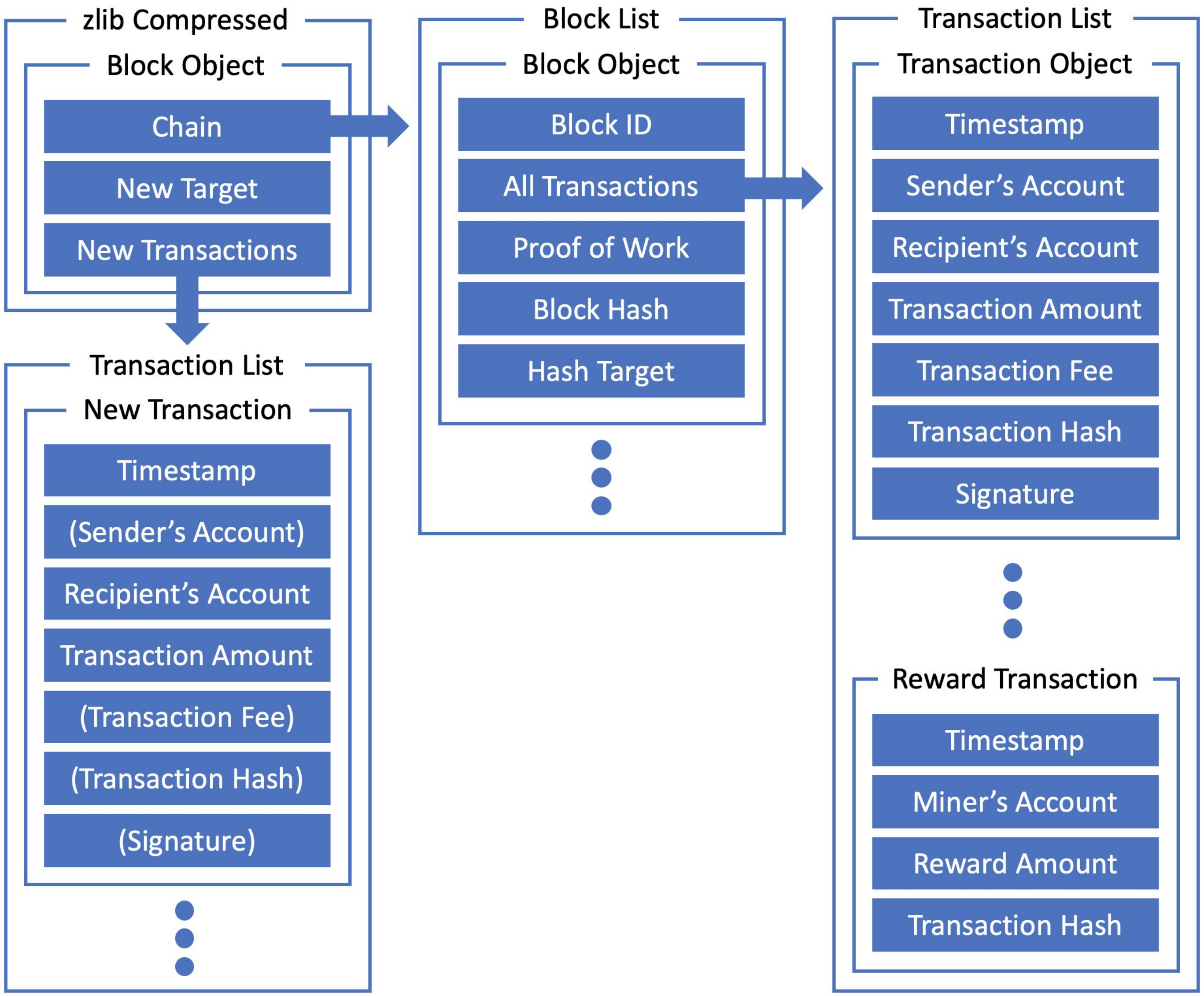
Value	Target RPS	Weight	Due date (at 11:59PM EST)
Query 1 Checkpoint	-	5%	Sunday, October 18
Checkpoint Report	-	5%	Sunday, October 18
Query 1 Early Bird Bonus	32,000	5%	Sunday, October 18
Query 1 Final	32,000	10%	Sunday, October 25
Query 2 Checkpoint	-	10%	Sunday, October 25
Query 2 Final	10,000	50%	Sunday, November 1
Final Report + Code	-	20%	Tuesday, November 3

You have only one week
to meet the Query 2 checkpoint and another week for
Query 2 Final

Question: Is 1 week enough time for that?

Hint: No. Start now.

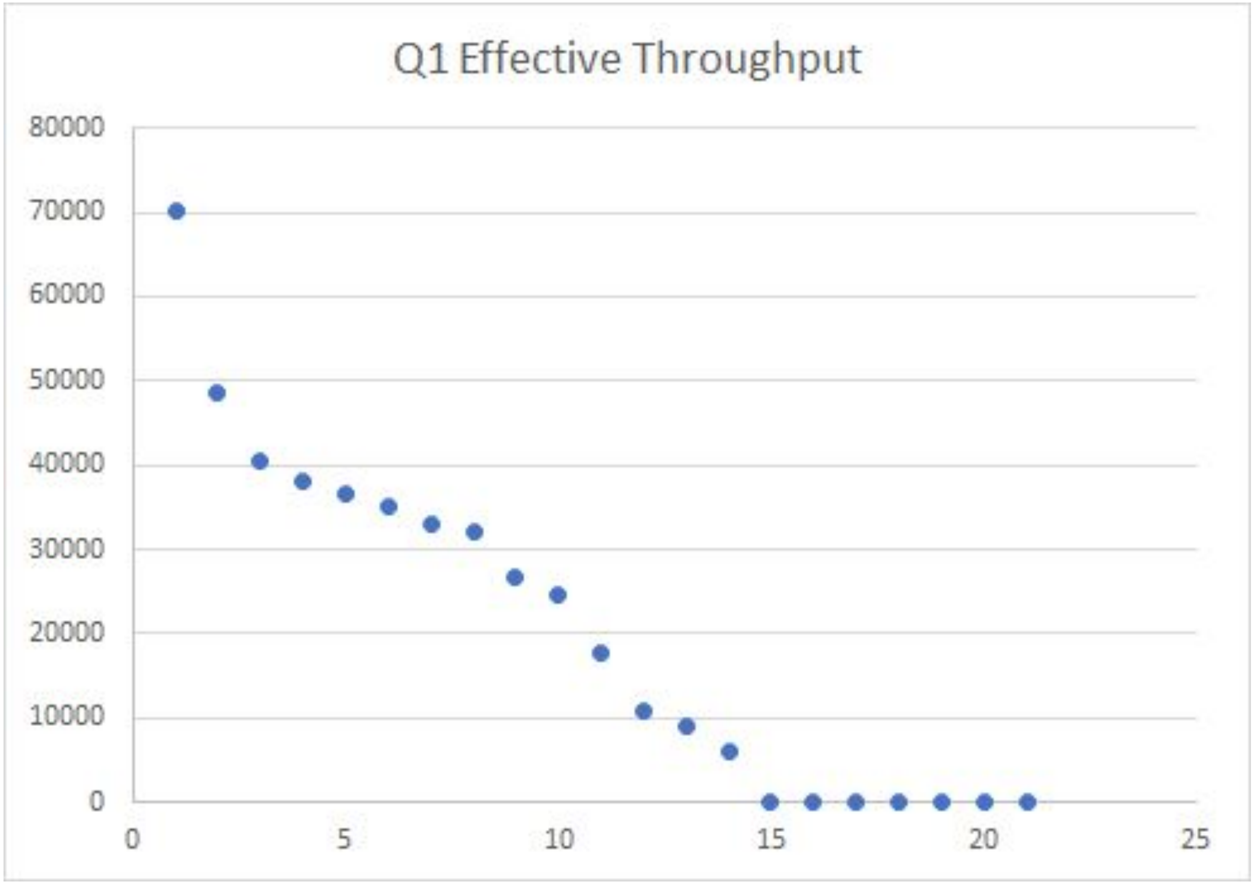
Query 1 Recap



```
{
  "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

- All 21 teams attempted Query 1 10-minute submission
- 8 teams reached 32,000 RPS



Q1 Tips

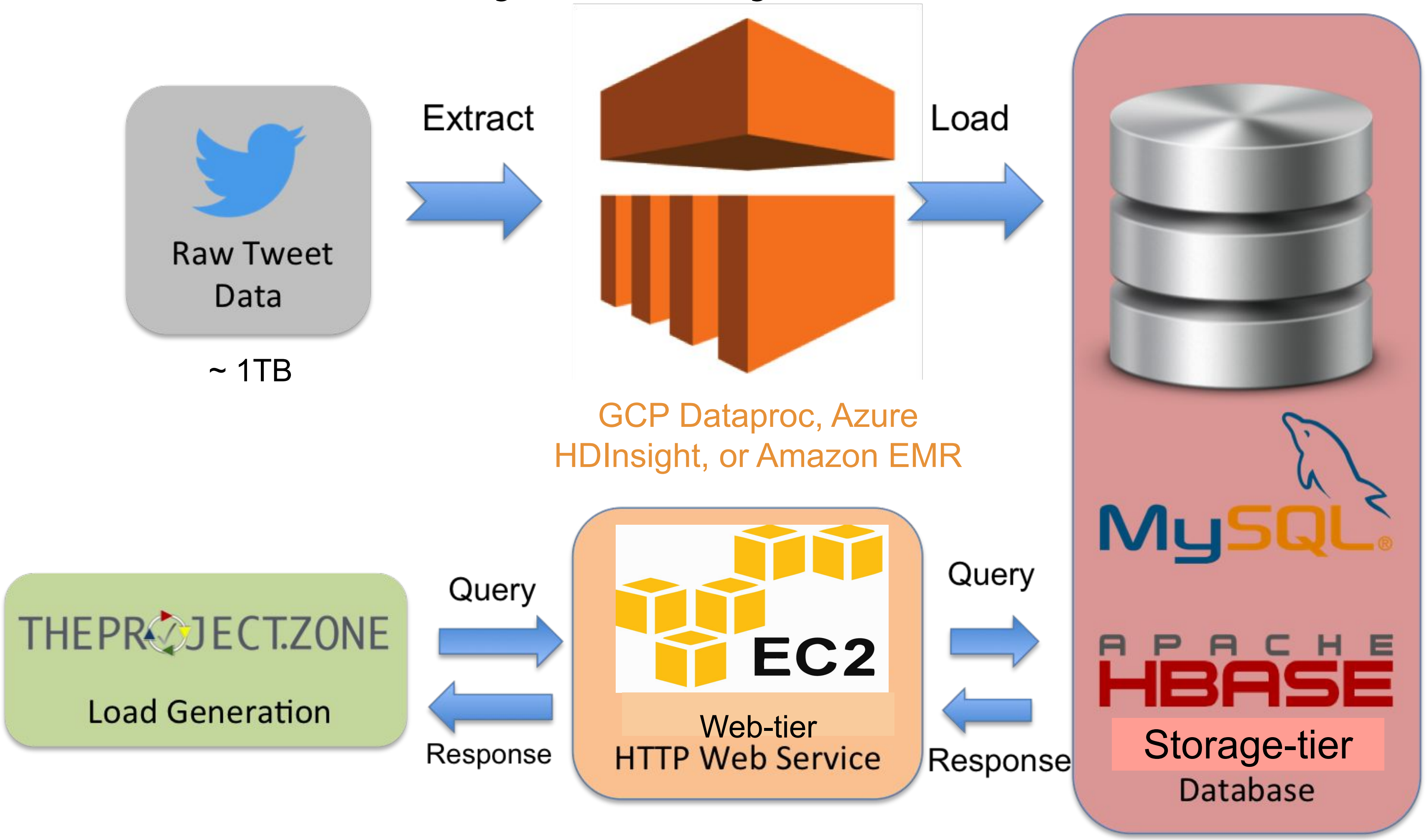
If the throughput is lower than expected

- Try m6g instances
  - If you build your own ami, you might need to rebuild it since it's arm-based
- Identify the bottleneck
  - Use a profiler
  - Logging timestamps before and after each function
- Change to another framework or even language

If the correctness is lower than expected

- Read the writeup more carefully

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: When you follow someone on twitter, recommend users you may also be interested in

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

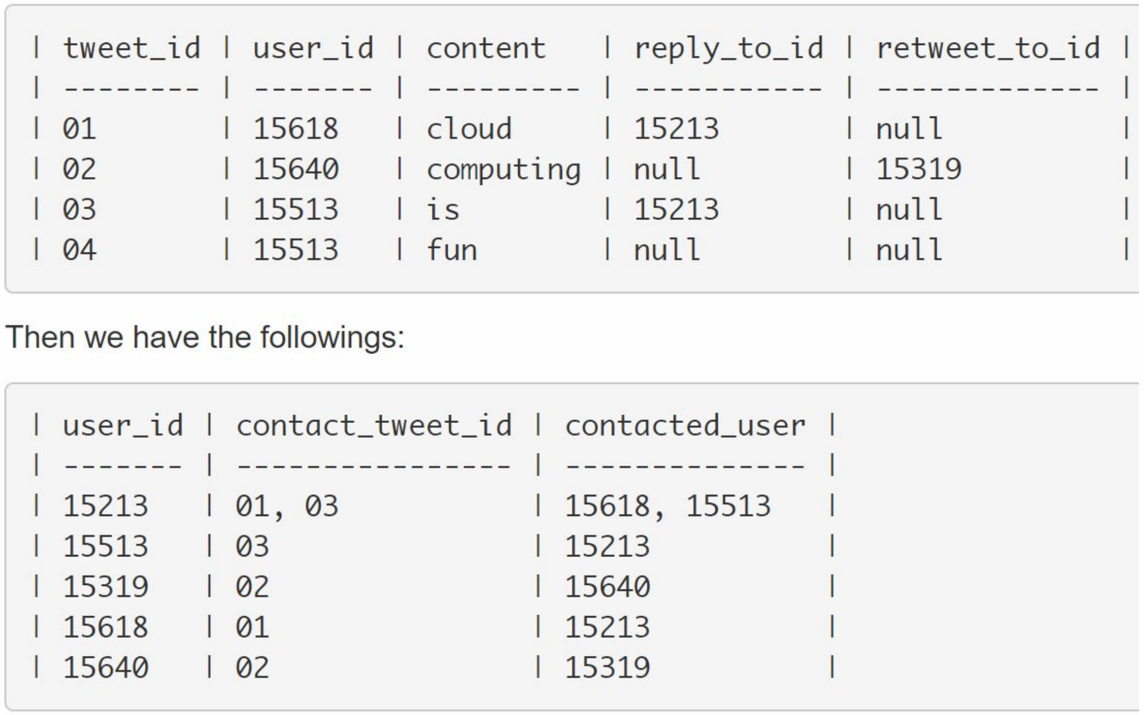
Final Score: Interaction Score * Hashtag Score * Keywords Score

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 - Filtering

Each line from the dataset is supposed to be a valid 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 - 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 * reply_count + 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`

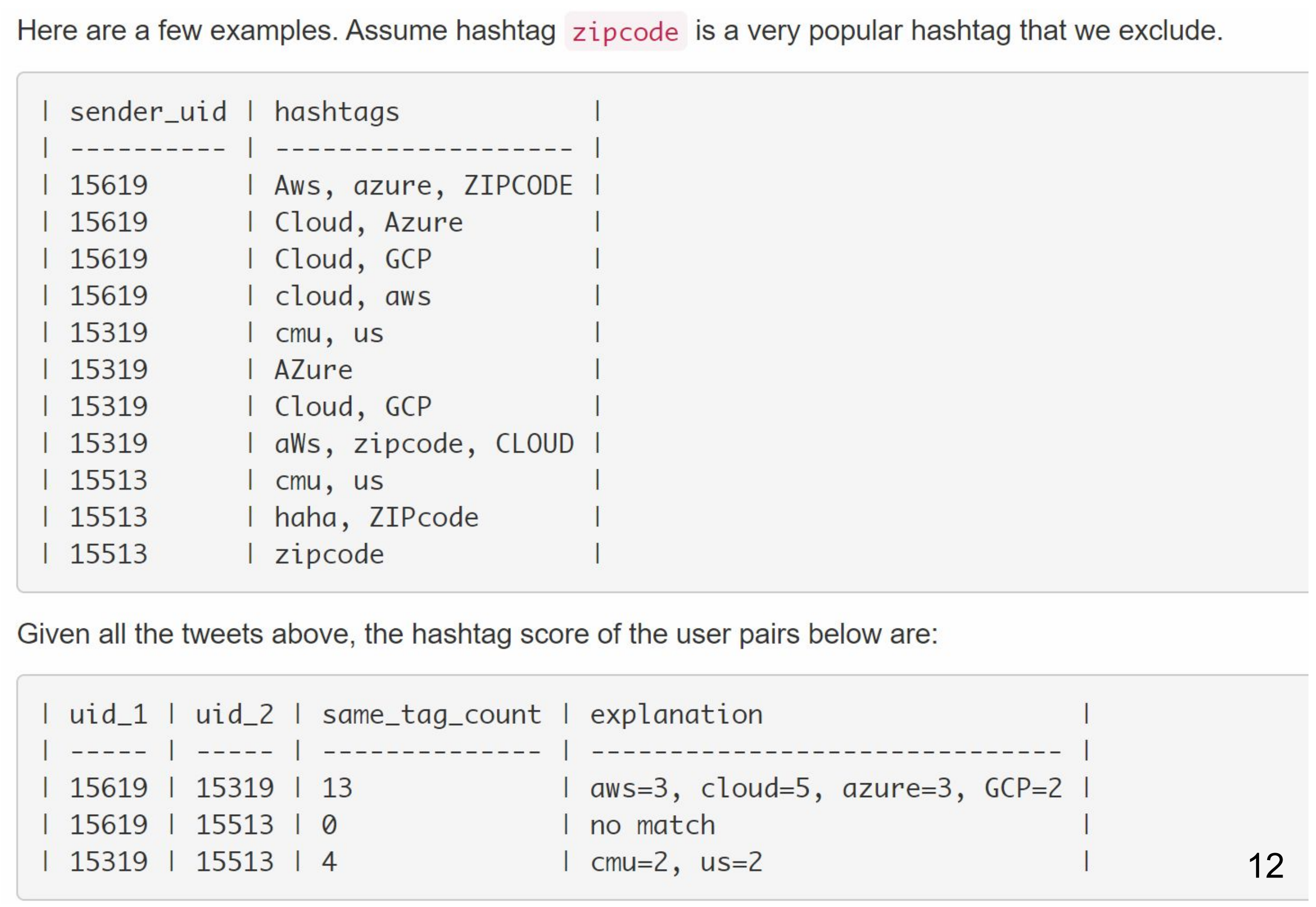
Query 2 - Hashtag Score

Same hashtags 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`

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



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

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: <u>case sensitive</u> match.

Another example, for phrase `haha`

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

Matching rule for the hashtag: <u>case insensitive</u> 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`

- log base is **e**,
- 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

- 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)
    - Before loading the result of mini dataset to MySQL and HBase, read the write-up (**Overview of Tasks - ETL**) carefully
    - You can load into MySQL first if you are more familiar with it. It’s highly recommended to test loading into HBase (AWS EMR) as well to get familiar with HBase. This will save you so much time and budget when you try to load the entire dataset into HBase in the next step.

Continued ….

Q2 Roadmap (continued)

- Start ETL on the entire dataset in GCP/Azure and compare your result against the reference server. Once you think the result good enough, load them to AWS.
- Start submit your endpoint to TPZ to test your implementation.
- Improve your implementation to achieve at least 80% correctness.
    - If you initial test on the mini dataset is not good enough, you may need to redo the entire ETL process multiple times!
- Optimize your implementation to reach the target throughput.

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)** 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

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
- You can use Spark or Hadoop - it is your choice since you have total freedom in ETL frameworks

Suggested Tasks for Phase 1

Phase 1 weeks	**Tasks**	**Deadline**
Week 1 ● 10/12 - 10/18	● Team meeting ● Read Writeup & Report ● Complete Q1 code & achieve correctness ● Start ETL on mini dataset and design q2 schema	● Q1 Checkpoint due on 10/18 ● Checkpoint Report due on 10/18
Week 2 ● **10/19 - 10/25**	● **Q1 target reached** ● **Q2 ETL & Initial schema design completed** ● **Achieve Q2 basic correctness and submit to TPZ**	● **Q1 final target due on 10/25** ● **Q2 MySQL Checkpoint due on 10/25** ● **Q2 HBase Checkpoint due on 10/25**
Week 3 ● 10/26- 11/1	● Achieved correctness for both Q2 MySQL, Q2 HBase & basic throughput ● Optimizations to achieve target throughputs for Q2 MySQL and Q2 HBase	● Q2 MySQL final target due on 11/1 ● Q2 HBase final target due on 11/1 ● Final Report due on 11/3