15-319 / 15-619
Cloud Computing

Overview 7

October 12th, 2021

Reflection of Last Week

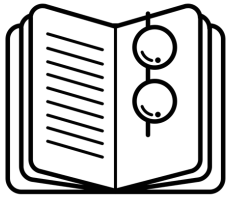
- Conceptual content on OLI
  - Module 10: Resource virtualization (Memory)
  - Module 11: Resource virtualization (I/O)
  - Module 12: Case Study

- Project theme - **Heterogeneous Storage on the Cloud**
  - Introduction to MySQL
  - Introduction to NoSQL (HBase)
  - Scenario: Build Your Own Social Network Website
  - Implement Basic Login with SQL
  - Store Social Relations as Graph using Neo4j
  - Build Homepage using MongoDB
  - Put Everything Together - Social Network Timeline
  - Caching

This Week

- **OPE - Spark Programming**
  - Due on **Sunday**, October 17th, 2021, 11:59PM ET
- **Quiz 6 (OLI Module 13)**
  - Due on **Thursday**, October 14th, 2021, 11:59PM ET
- **Project 3**
  - Due on **Sunday**, October 16rd, 2021, 11:59PM ET
- **Team Project Phase 1 M2 Final + M3 Checkpoint**
  - Due on next **Sunday**, October 24rd, 2021, 11:59PM ET

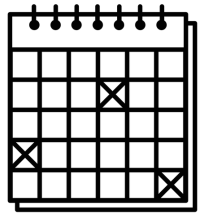
This Week: Conceptual Content



- OLI, Unit 3: Cloud Infrastructure
  - Module 7: Introduction and Motivation
  - Module 8: Virtualization
  - Module 9: Resource Virtualization - CPU
  - Module 10: Resource Virtualization - Memory
  - Module 11: Resource Virtualization – I/O
  - Module 12: Case Study
  - Module 13: Storage and Network Virtualization

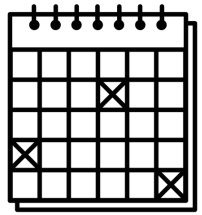
TEAM PROJECT
Twitter Data Analytics

Team Project Time Table



Phase	Deadline (11:59PM ET)
Phase 1 (20%) **- M1** **- M2** **- M3 (ckpt)**	• M1 CKPT (5%): Sun, 10/3 • M1 CKPT Report (5%): Sun, 10/3 • M1 FINAL (10%): Sun, 10/10 • M2 CKPT (5%): Sun, 10/10 • **M2 FINAL (50%): Sun, 10/24** • **M3 CKPT (5%): Sun, 10/24** • **Final Report + Code (20%): Tue, 10/26** **BONUSES:** • M1 Early Bird Bonus (5%): Sun, 10/3 • M2 Early Bird Bonus (5%): Sun, 10/10 • M2 Correctness Penalty Waiver: Sun, 10/10 • **M3 Early Bird Bonus (5%): Sun, 10/24** • **M3 Correctness Penalty Waiver: Sun, 10/24**

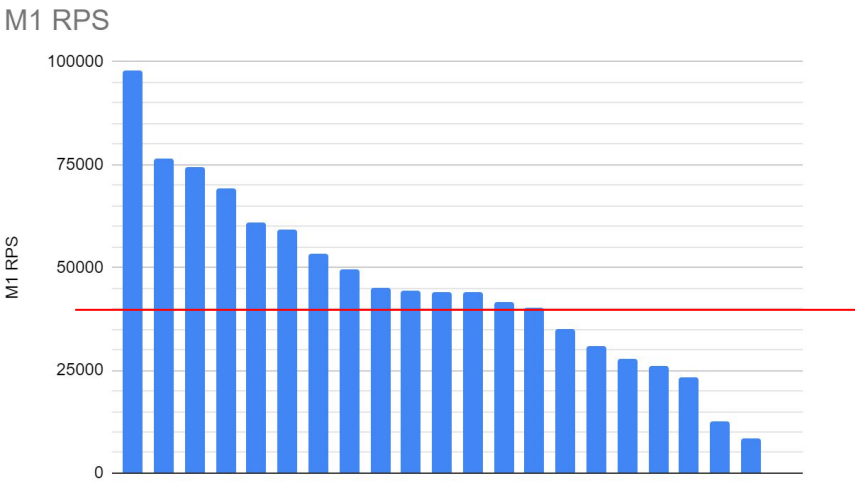
Team Project Time Table

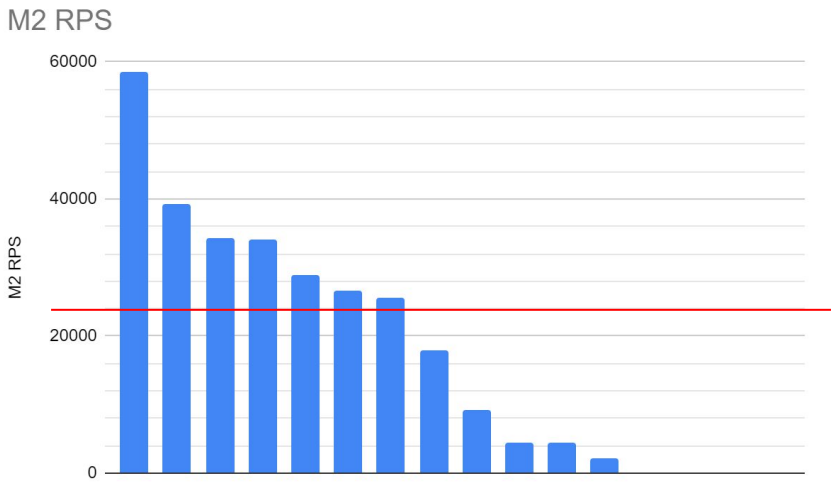


Phase	Deadline (11:59PM EST)
Phase 2 (30%) **- M1** **- M2** **- M3 (full)** **- Live Test!**	• Live Test on Sun, 11/7
Phase 3 (50%) **- Managed Services for Microservice 1-3** **- Live Test!**	• Live Test on Sun, 11/21

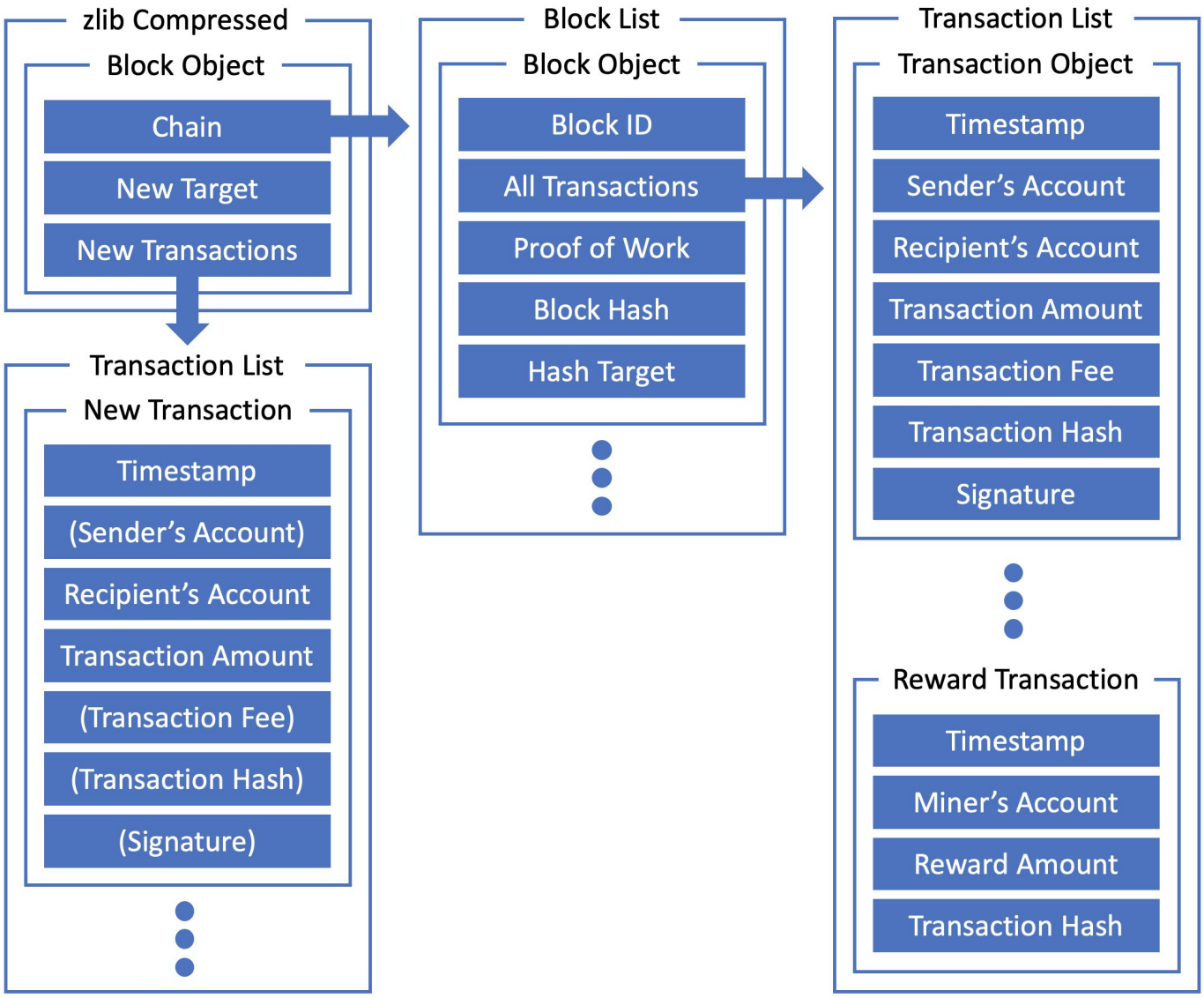
Recap of M1 and M2

- Microservice 1
  - 21/23 teams made a non-zero score 600s submission
  - 14/23 teams achieved 40,000 RPS throughput
- Microservice 2
  - 12/23 teams made a non-zero score 600s submission
  - 7/23 teams achieved the early-bird bonus





Microservice 2 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"
    }
  ]
}
```

Microservice 2 Tips

- Start with a single EC2 instance, work with cluster only when you are confident
- My RPS is low
  - Does your program utilizes all CPU core? Make sure threads / workers are set up properly.
  - Profiling. The Primer can be useful.
  - Try out different instance types. m6g can have better performance/cost ratio.
  - Try different framework or even language.
- After deploying to cluster, make sure the workload is evenly distributed across your worker nodes

Microservice 2 Tips

- My correctness is low
  - Does the sender have enough balance to pay the recipient and the fee?
- How to find PoW?
  - PoW can be any random string as long as it satisfies the hash target
  - for i from 0 to infinity:

    pow = string(i)

    hash = cchash(tx_hash+pow)

    if hash < target: return (pow, hash)

Read M3 Now.
Start ETL Now.

Microservice 3 Checkpoint	-	5%	Sunday, October 24th
Microservice 3 Early Bird Bonus	10,000	5%	Sunday, October 24th
Microservice 3 Correctness Bonus	-	Waive one most significant penalty for each team member	Sunday, October 24th
Final Report + Code	-	20%	Tuesday, October 26th

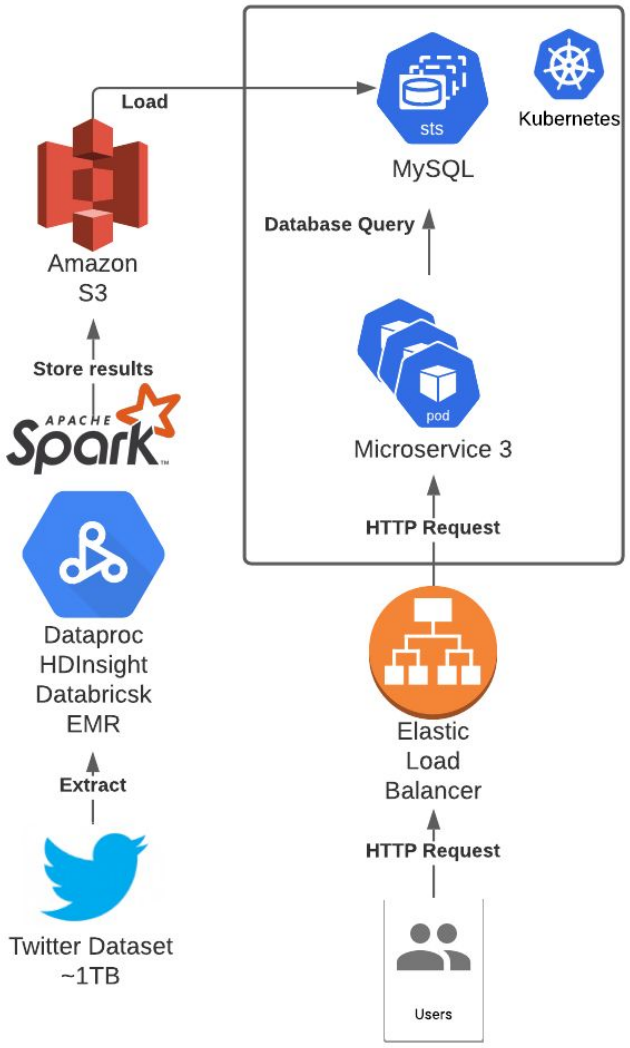
You have two weeks to meet the M3 checkpoint

Question: Is one weekend enough time for M3?

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



M3 - User Recommendation System

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

Use Case: Recommend User B, C and D when you follow User A 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
/twitter?**user_id**=<ID>&**type**=<TYPE>&**phrase**=<PHRASE>&**hashtag**=<HASHTAG>

Response:

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

M3 - Filtering

Each line from the provided files is a tweet object

- Malformed JSON Object
- Malformed Tweets
- Each tweet must contain valid
  - Tweet ID
  - Sender’s user ID
  - Timestamp
  - Content
  - At least 1 hashtag
- Tweets not in the required languages
- Duplicate Tweets

M3 - 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

M3 - 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**.

M3 - Interaction Score

- Two types of interaction: Retweet and Reply

- **Interaction score** =
  `log(1 + 2 * reply_count + retweet_count)`

Examples:

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`

M3 - Hashtag Score

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

For the cases of self-reply or self-retweet , 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

M3 - Keyword Score

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. Example: `haha`

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

Matching rule for the hashtag: case insensitive exact match. Example: `cloud`

Tweet having hashtags **`#Cloud #CLOUD #CLOUD`** `#cmu`
Note that duplicate tags are allowed, thus `number_of_matches += 3`.

If there are no contact tweets of the type specified in the query, `keywords_score = 0`.

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

M3 - Final Score and Ordering

Final Score

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

- Keep 5 decimal points of precision rounding half up **before** ranking
- Ignore user pairs with a final score of 0

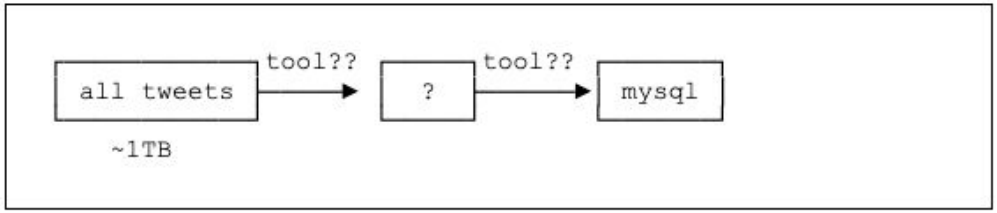
Ordering

- **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.

M3 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
    - Zeppelin can be your good friend
    - 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
    - Store your ETL result as TSV/CSV files
    - Make use of the mini-ref server

Continued ….

M3 Roadmap (continued)

- Start ETL on the entire dataset in GCP/Azure and compare your result against the reference server.
  - Store your ETL results as TSV/CSV and use tools to import into database
  - E - T - L - Verify with reference server
  - Spot the bug, correct the code and rerun the ETL
    - Can you just rerun the ETL partially? Store intermediate results?
  - 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 the computation takes X hours and loading all the data into database takes Y hours, how many iterations can you do?
    - Can you accelerate your design iteration?

Spark, Scala and Zeppelin Primers

- Primers for Apache Spark/Scala/Zeppelin are now available
- Learn more about Spark in OPE, Project 4 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
- You can use Spark or Hadoop - it is your choice since you have total freedom in ETL frameworks

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 guideline post
- 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

- Self-managed Kubernetes cluster + optional EMR, consisting of 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 final 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 $1.10/hour (HBase)** applies to every submission

Phase 1 Budget

- Your web service should not cost more than **$0.70/hour (M1, M2 and M3 MySQL) and $1.10/hour (M3 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 - excluding LCU-hour cost**
  - We will not consider the cost of data transfer and EMR software
  - See writeup for details
- AWS total budget of $80 for Phase 1

Best Wishes!!!