Twitter Analytics Web Service
Project Competition
Phase 2
(Due at 11:59PM EDT, Apr 8)

Introduction
In phase 1 of the project, you have already built a web service consisting of a front end system and two different databases. In this phase, you are going to use the knowledge and skills you obtained from phase 1 to process and store a new larger dataset and one new query. Unlike Phase 1, where you mostly focused on the system functionality, this time you will learn process a larger dataset along with the necessary database schema design and further system performance tuning. By the end of this phase, you should have a clear idea of the performance difference between HBase and MySQL, and in phase 3 you have to make a decision on which database to use for phase 3 and the final live test.

Dataset
The dataset for this phase is about 200 GBs containing about 70 million tweets. You will be using this dataset for phases 2 and 3.The tweets are in JSON format. For a more detailed explanation, please refer to the overall project writeup.

The dataset is stored in the following folder on S3:
s3://15619project-dataset-s14/phase2

Query types:
- ★ In Phase 2 and Phase 3, we do not have a minimum throughput requirement for each query anymore. Instead, we will provide you a full score line for each query when we release Phase 2 testing system later. Please refer to the detail score calculation in the overall project writeup.

Your web service's front end will have to handle the following query types through HTTP GET requests **on port 80**:

1. **Heartbeat (q1)**
   The query asks about the state of the web service. The front end server responds with the project team id, AWS account id and the current timestamp. It is generally used as a heartbeat mechanism, but it could be abused here to test whether your front end system can handle varying loads.
   - REST format: GET /q1
     - Example: http://webservice_public_dns/q1
   - response format: TEAMID,AWS_ACCOUNT_ID

yyyy-MM-dd HH:mm:ss

- Example: WeCloud,1234-5678-9012
  2013-10-02 00:00:00

- Full score throughput: **16000 qps.**
- Full score throughput for mix-q1: **7000 qps**

2. **Find a particular tweet using userid and timestamp (q2)**
   The query asks for the Tweet ID posted by a given user at a specific time. This query can basically test whether your backend database is functioning properly.
   - REST format: GET /q2?userid=uid&tweet_time=timestamp
     - Example: http://webservice_public_dns/q2?userid=123456789&tweet_time=2013-10-02+00:00:00
   - response format: TEAMID,AWS_ACCOUNT_ID
     Tweet ID 1
     Tweet ID 2
     …
     Tweet ID n
     (all Tweet IDs need to be sorted numerically. There should be a line break '\n' at the end of each response.)
     - Example: WeCloud,1234-5678-9012
       12345678987654321
       98765432123456789
   - Full score throughput: **4500 qps.**
   - Full score throughput for mix-q2: **2000 qps**
   - Correctness file for q2 on small dataset is available at: s3://15619project-dataset-s14/correctness_files/q2s_answers.txt

3. **Who retweeted a tweet (q3)**
   The query asks for the set of userids who have retweeted any tweet posted by a given userid.
   - request format: GET /q3?userid=133710000
     - Example: http://webservice_public_dns/q3?userid=133710000
   - response format: TEAMID,AWS_ACCOUNT_ID
     userid1
     userid2
     …
     useridn
     (all userids need to be sorted numerically. There should be a line break '\n' at the end of each response.)
     - Example: WeCloud,1234-5678-9012
       1000205192
       1117007780

- Full score throughput: **7000 qps.**
- Full score throughput for mix-q3: **3000 qps**
- Correctness file for q3 on small dataset is available at: s3://15619project-dataset-s14/correctness_files/q3s_answers.txt

Tasks

- ★ *Make sure you tag all the instances you use in this phase with the key "**15619project**" and value "**phase_2**". Remember the tags are **case-sensitive**.*
- ★ *Our test system is built in availability zone **us-east-1a**. We recommend that you provision your instance in the same availability zone to avoid having the network becoming a potential bottleneck.*

Task 1: Front End

As you may already find, in this phase, a new query q3 is added. So your first task is to update your front end system to handle q3. Also, from now on, the scoreboard will rank your performance based on the throughput of your system. Since front end performance affects all the queries, you should make sure that your front end system does not limit your overall system performance. Hint: spend some time to optimize your front end system.

Design constraints:

- Execution model: you can use any web framework you want.
- Cost budget: The front end configuration should not cost more than $0.3/hour at light load and no higher than $1/hour at maximum load.
- Spot instances are highly recommended during development period, otherwise it is very likely that you will exceed the overall budget (see below) for this step and fail the project.

Front end submission requirements:

1. The design of the front end system
2. Provide all code developed for the front end
3. The type of instances used and justification
4. The number of instances used and justification
5. Other configuration parameters used
6. The cost per hour for the front end system
7. The total development cost of the front end system
8. The throughput of your front end for the given workload of q1 queries

Recommendation:

You might want to consider using auto-scaling because there could be fluctuations in the load over the test period. To test your front end system, use the Heartbeat query (q1). It will be wise to ensure that your system can satisfy the minimum throughput requirement of heartbeat requests before you move forward. However, as you design the front end, make sure to account for the cost. Write an automatic script, or make a new AMI to configure the whole front end instance. It can help to rebuild the front end quickly, which may happen several times in the building process. Please terminate your instances when not needed. Save time or your cost will

increase higher than the budget and lead to failure.

Task 2: Back end (database)
In this phase, one of the most important tasks is to explore the performance difference of MySQL and HBase on the large dataset.

Similar to phase 1, the databases you are going to use in this task are both **HBase** and **MySQL**. However, now you need to work on a much larger dataset.

This task requires you to build the back end system. It should be able to store whatever data you need to satisfy the query requests. You **MUST** use spot instances for the back end system development. But you can use on-demand instances for the live test. Normally, the dataset storage system is the most crucial part of a web service. People cannot take the risk of losing any bit of data, so please use on-demand prices to launch spot instances to reduce the probability of instances termination when testing your system. Do this after you are confident that you have completed system development.

For the database design, you should take q2 and q3 into consideration, which includes the requirement of throughput and cost. We require you to build two back ends, one with ***HBase*** and the other with ***MySQL*** as your back end databases. You need to consider the design of the table structure for the database. The design of the table (schema) significantly affects the performance of a database.

In this task you should also test and make sure that your front end system connects to your back end database and can get responses for queries.

Design constraints:

- Storage model and justification: you should use both HBase or MySQL and explain the differences between them.
- Cost budget: The total cost (front end system + back end system) should not cost more than **$1.6/hour**. If you are using HBase on EMR cluster, the cost of EMR can be excluded from the cost, because in a real world setting you can configure your Hadoop cluster without additional cost.
- Spot instances are highly recommended during the development period, otherwise it is very likely that you will exceed this phase's overall budget and fail the project.

Back end submission requirements:

1. The design of the back end system
2. The table structure of the database, justify your design decisions
3. The type of instances (m1.small, m1.medium or m1.large) used and justification
4. The number of instances used and justification
5. The cost per hour for the back end system
6. The spot cost for all instances used

7. The total development cost of the back end system

Recommendations:
Test the functionality of the database with a few dummy entries before the Twitter data is loaded into it. The test ensures that your database can correctly produce the response to the q2 query.

Task 3: ETL

- ★ *The ETL process can last from hours to tens of hours depending on your ETL solution, code efficiency and the database you are using. Testing will be crucial to limit your expenditure and improve your debugging process.*
- ★ *ETL is probably the most costly task in the project and you need to seriously consider the approach you do the query. We strongly recommend you to consider:*
  1. generating some intermediate results that can be used for both HBase and MySQL
  2. performing ETL on multiple queries in the same run to improve efficiency.

This task requires you to load the Twitter dataset into the database using the **extract**, **transform** and **load** (ETL) process in data warehousing. In the extract step, you will extract data from an outside source. For this phase, the outside source is a 200 GB Twitter dataset of JSON file stored on **S3,** containing about 70 million tweets. The transform step applies a series of functions on the extracted data to realize the data needed in the target database. The extract step relies on the schema design of the target database. The load phase loads the data into the target database.

You will have to carefully design the ETL process using AWS resources. Considerations include the programming model used for the ETL job, the type and number of instances needed to execute the job. Given the above, you should be able to come up with an expected time to complete the job and hence an expected overall cost of the ETL job.

Once this step is completed, you **MUST** backup your database to save cost. If you use EMR, you can do the Hbase backup on S3 using the command:

```
./elastic-mapreduce --jobflow j-EMR_Job_Flow --hbase-backup
--backup-dir s3://myawsbucket/backups/j-EMR_Job_Flow
```

The backup will run as a Hadoop MapReduce job and you can monitor it from both Amazon EMR debug tool or by accessing the http://jobtracker-ip:9100. To learn more about Hbase S3 backup, please refer to the following link
http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-hbase-backup-restore.html

Design constraints:

- Programming model: you can use any programming model you see fit for this job.
- AWS resources: You **MUST** use SPOT instances for this step otherwise it is very likely that you will exceed the budget and fail the project.

- ETL code verification: Before you start the ETL process on the large dataset, you **MUST** verify the correctness of your ETL code using the small dataset and corresponding correctness files we provided for each query.

ETL submission requirements for both MySQL and HBase:

1. The code for the ETL job
2. The programming model used for the ETL job and justification
3. The type of instances used and justification
4. The number of instances used and justification
5. The spot cost for all instances used
6. The execution time for the entire ETL process
7. The overall cost of the ETL process
8. The number of incomplete ETL runs before your final run
9. Discuss difficulties encountered
10. The size of the resulting database and reasoning
11. The time required to backup the database on S3
12. The size of S3 backup

Recommendations: For this phase, the ETL jobs will utilize a large dataset (200 GB). Always test the correctness of your system using a small dataset. If your ETL job fails or produces wrong results, you will be burning through your budget and wasting time. You only have 2 weeks for this phase. The ETL job depends on you completing Step 2 (especially for the T and L in ETL). After the database is populated with data, it will be wise to test the throughput of your back end system for different types of queries. Always test that your system produces correct query responses for all query types.

Task 4: Performance Optimization

By this stage, your system is functionally operational and you have already tested its performance and understand its limitations. You might want to consider certain optimizations to performance while accounting for cost. Remember, you are competing against other companies (teams) to win the contract.

Performance Optimization submission requirements:

1. The performance comparison between large dataset and small dataset for q2.
2. The optimizations utilized based on insights gain. Always provide justifications.
3. Changes to the overall system design for optimization.
4. Changes to the overall system implementation for optimization.
5. For the given workload
    a. The maximum throughput of the optimized system for q1, q2 & q3
    b. The latency of your optimized system for q1, q2 & q3
6. The cost per hour of your optimized system

Budget
You will have a budget **$40/team** for all the tasks in this phase. Again, make sure you tag all of your instances in this phase with key "***15619project***" and value "***phase_2***". You decide how to spend your budget.

Recommendations:
1. Use smaller instances when you do function development and verification of the front end (you may even finish a lot of the development needed for this task on your own laptop).
2. Think carefully about your database schema to avoid running long ETL jobs too many times.

Deliverables
You need to submit your system for **a live test** at the end of this phase. **You need to and are only able to submit 2 requests for live test, one for System with HBase, the other one for System with MySQL.** The submission time of live test request on the testing system must be before the deadline of this phase. Besides, please submit

1. Phase 2 checkpoint report in PDF format(Phase2 checkpoint report template)
2. All your code written in this phase (front end, ETL, etc...)

To submit your work:

1. Package your completed Phase 2 checkpoint report and all the source code files you wrote in the phase into a single TEAM_NAME.zip file.
2. Upload the TEAM_NAME.zip file to S3.
3. Submit your s3 link using this Google form

Grading
This phase accounts for 15% of the total grades of the entire project. You need to finish all the tasks in order to move on to the next phase.
If you finally choose MySQL as your database, your Phase 2 grade will be based on the performance of your HBase solution. Otherwise, if your final choice is HBase, grading of Phase 2 will be based on the performance of your MySQL solution. This simply means that you should make an effort to explore the performance benefits of both databases.

Live Test Schedule
The live test of this phase lasts **2.5 hours consisting of five parts**:
1. 30 minutes warm up test for q1 (not graded)
2. 30 minutes test for q1
3. 30 minutes test for q2
4. 30 minutes test for q3
5. 30 minutes mixed queries test

Hints and References
Front End Suggestions:
Although we do not have any constraints on the front end, the performance of different web

frameworks varies a lot. Choosing a bad web framework may have a negative impact on the throughput of every query. Therefore, we strongly recommend that you **do some investigation about this topic** before you start. Techempower provides a very complete benchmark for mainstream web frameworks, you may find it helpful.

You can also compare the performance of different frameworks under our testing environment by testing q1 since it is just a heartbeat message and has no interaction with the back end. Please also think about whether the front end framework you choose has API support for MySQL and HBase.

General Guideline for ETL:

How to do ETL correctly and efficiently will be a critical part for your success in this project. Notice that ETL on a large dataset could take 10 - 30 hours for just a single run, so it will be very painful to do it more than once, although this may be inevitable since you will be refining your schema throughout your development.

You may find your ETL job extremely time consuming because of the large dataset and poor design of your ETL process. Due to the many reasons that could lead to the failure of your ETL step, please start thinking about your database schema and doing your ETL as EARLY as possible.

Also try to utilize parallelism as much as possible, loading the data with single process/thread is a waste of time and computing resources, MapReduce may be your friend, though other methods work so long as it can accelerate your ETL process.

Reference documents for MySQL and HBase

MySQL

- http://dev.mysql.com/doc/
- OLI Unit 4 and Project 3

HBase:

- https://hbase.apache.org/
- OLI Unit 4, Module 14
- OLI Project 3 and Project 4