Twitter Analytics Web Service
Project Competition
Phase 1

(Due on 11:59PM EST, Mar 25)

Introduction

In this phase you will build a web service from scratch. The web service you provide should be able to provide responses to two types of queries. This phase is essential for the development of later phases. It is designed to prepare you with the required skills to cope with more complicated queries, so please take this phase very seriously and allocate enough time to finish the requirements by the deadline. There will be a checkpoint at the end of this phase, at that time you should have a web service up and running to give the correct response to two queries and pass a very basic performance baseline. Specifically, you will design and develop your front end system along with two back end systems (Hbase and MySQL). In this phase you will be expected to learn about the basic advantages and disadvantages of each type of back end database so that you can articulate them in the writeup of this phase.

Dataset

The dataset for this phase is a file containing about 0.8 million tweets, which is a relatively small dataset. The tweets are in JSON format. For more detail explanation please refer to the overall project writeup.

The dataset is stored on S3.

s3://15619project-dataset-s14/phase1/part-00000

You may use the folder as input if you are using MapReduce.

Query types:

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
   - Minimum throughput requirement: 4000 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\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
   - Minimum throughput requirement: 1000 qps

Tasks

★ *Make sure you tag all the instances you use in this phase with the key "**15619project**" and value "**phase_1**". Remember the tags are **case-sensitive**.*

Task 1: Front End

This task requires you to build the front end system of the web service. The front end should accept RESTful requests and send back responses. As the first step, the front end system is only required to handle q1 and q2. As you may have already noticed, q1 queries do not require connectivity to back end databases, while q2 needs to query the back end database in order to provide the proper response to the query.

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 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)
For your system to provide responses for q2, you need to store the required data in a back end database. Your front end system connects to the back end and queries it in order to get the response and send it to the requester.

The databases you are going to use in this task are both **HBase** and **MySQL**. We will provide you with some references that will accelerate your learning process. You are expected to read and learn about these database systems in order to finish this task. You also need to know enough about the advantages and disadvantages of each database system in order to answer the questions in the Phase 1 report template.

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 prices when you test your back end system. 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 price 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 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 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: Each back end system should not cost more than $1/hour (on demand prices). If you are using HBase on EMR cluster, the cost of EMR can be excluded from the cost, because in real world 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
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 2.3 GB Twitter dataset of JSON file stored on **S3,** containing about 0.8 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 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 small dataset (2.3GB), however, in later phases, you will be required to run ETL for a much larger dataset. 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. 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.

Budget

You will have a budget **$20/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_1***". You decide how to spend the budget. You will be penalized on grade if you overrun 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 ETL too many times.

Deliverables

You need to make sure you have at least **1 submission record for q1 and 2 submission**

records for q2 (one for MySQL and another for HBase) which meet the minimum throughput requirement on testing system. This phase is graded based on your performance of those records. The submission time of the record on the testing system should be before the deadline of this phase. Besides, please submit

1. Phase 1 checkpoint report in PDF format (Phase1 checkpoint report template)
2. All your code written in this phase (front end, ETL, etc...)

To submit your work:

1. Package your completed Phase 1 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 10% of the total points of the entire project. You need to finish all the tasks in order to move on to the next phase.

Hints and References

Front End Suggestions:

Although we do not have any constraints on the front end, performance of different web frameworks do vary a lot. Choosing a bad system 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 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 interactions 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 massive dataset (we will release an even larger dataset in the later phases) and poor design of your ETL process. Due to many reasons that lead to 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, map-reduce 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