15619 Project Phase 2 Report

Performance Data and Configurations

	Front end	Web service with HBase	Web service with MySQL
Query	q1	q2/q3/q4/ mixed	q2/q3/q4/mixed
Scoreboard request ID	26549		26620/26698/26816/27017/27019/ 27023/27027
Submission timestamp	2014-11-07 00:36:48 (-0500)		2014-11-07 01:06:51 (-0500) 2014-11-07 01:36:54 (-0500) 2014-11-07 02:07:04 (-0500) 2014-11-07 02:07:04 (-0500) 2014-11-07 02:37:06 (-0500) 2014-11-07 02:37:06 (-0500) 2014-11-07 02:37:06 (-0500)
Instance type	m3.large		m3.large
Number of instances	5		5
Queries Per Second (QPS)	13059.1		3846.6/10533/9261.1/3038.7/1268. 1/1514.4/1527.0
Error rate	0.00		0.00/0.00/0.00/0.01/0.01/0.02/0.01
Correctness	100		100/100/100/99/99/100/98/99
Cost per hour	$0.7		$0.7

Do you want to be graded on MySQL or HBase? MySQL

Task 1: Front end

Questions

1. Which front end system solution did you use? Explain why did you decide to use this solution.

We used JBoss' Undertow IO as our frond end system's framework. According to Techempower's test on frameworks, Undertow is among the best web frameworks with regards to performance in the area of JSON serialization, database queries and plaintext HTTP responds. The reason why Undertow framework is fast is mainly because it is an unblock framework which takes the advantage of java NIO whose intent is to facilitate an implementation that can directly use the most efficient operations of the underlying platform. Besides, unlike some frameworks which need tomcat to support, Undertow is very lightweight and don't need many setups. Thus, we choose Undertow IO as our front-end system's framework, which is proved to be a right choice.

2. Explain your choice of instance type and numbers for your front end system.

We chose m3.large instance as our instance type because it had the best computing performance and the largest network bandwidth among the instance types we could choose. At first, we implemented Undertow IO on one instance and we got the throughput up to 1759.2 rps (please check the submission with submission ID 2585). This is far away from the bottom line which required 15000 rps. Then we decided to use Elastic Load Balancer to help us to distribute the network traffic to different instances. We also found out that even we used 5-6 m3.large instances to handle the requests, the best throughput we could get is 12500.2 rps which is not far away to 15000 rps. We believe that ELB has the bandwidth limit which cannot help us get the 15000 rps throughput. However, we need to use ELB to distribute requests to different instances to improve the throughput of q2,q3 and q4. Thus, even though we cannot achieve 15000 rps with ELB, we still use it. And that's a tradeoff.

3. Did you do any special configurations on your front end system? Explain your design decisions and provide details here.

Yes, we add a cache to each web service to improve the throughput of requests. However, the cache doesn't improve the throughput too much because of the ELB bandwidth. When we test the q1 throughput using one instance, we can get the best q1 throughput about 18000 rps. We have discussed that we need ELB to improve the throughput of q2, q3 and q4. We develop the whole web service based on Undertow include q2, q3 and q4.

4. What is the cost to develop the front end system?

In the development, I just use one m3.large instance to test q1, q2, q3 and q4.

5. What did you change from Phase 1 and why?
    $0.14*10 = $1.4

Task 2: Back end (database)

Questions
1. Describe your table design for both HBase and MySQL. Explain your design decisions.

In phase 2, we only used and designed MySQL as the backend of the whole system to finish the test.

In the design part of MySQL of query 2, firstly, we were planning to design the table as three columns (time, userid, tweet) and then used the time and userid as index for querying, but for some technical reason, the MySQL server in the instance kept connection timeout. Therefore, we decided to use the table structure of phase 1, in which column id is concatenated by user_id and create_at(time) and column text is concatenated by tweet_id, censored score and tweet_text. Since it is not necessary to guarantee the uniqueness of any field, we create the table with adding any primary key. We increase the speed of querying by creating index of user_id, which is contained in the first 8-11 characters of the id field.

In the design for query 3, there are only two columns in the whole table(id, retweetId), and the data size is small— 1.3G. Thus, we didn't make many modifications on the design of the database and the table. We created a primary key on the id column, because it can be sorted with its value's uniqueness property.

For query 4, after the ETL process, we got data with size of 1.1G. We designed a table with five columns(time, location, hashtag, id, rank). At the first of beginning, to speed up the query, we created three indexes on the table—time index, location index, rank index, because the input are the three columns. But the throughput was not ideal enough, then we changed our strategy to create a index on multiple columns, and did research on the sequence of columns in the process of creating index. Column time has the least uniqueness, we placed it as the first index argument. The second least unique column is column location, we used it as the the second argument. And we took column rank which has the highest cardinality as the third index argument. This improvement incased our throughput from 1400 TPS to 4100 TPS.

2. What is the cost to develop your back end system?

$7.6

Task 3: ETL
Since ETL was performed for both HBase and MySQL, you will be required to submit information for each type of database.

MySQL:

1. The code for the ETL job

We have submitted it. Please refer to the code and comment.

2. The programming model used for the ETL job and justification

Step 1). ET (Extract and Transform) => MapReduce

According to our database schema, we use MapReduce for extract and transform steps, because the data set is very large (1TB). In the Mapper stage, one master node takes the twitter input records, divides it into smaller subtasks, and distributes them to slave nodes. The slave nodes only split the input record and return a processed record in format. In the Reducer stage, the nodes aggregate some items and calculate the necessary information to print out.
[Note]
We use -Dfile.encoding=UTF-8 option in MapReduce to solve Unicode problem.
To compare the performance for various data format in databases, we create some processing tools using Java.

Step 2). L (Load)

We use the following command to load data to MySQL database:
LOAD DATA LOCAL INFILE "file path" INTO TABLE tablename FIELDS TERMI NATED BY "delimiter" LINES TERMINATED BY "delimiter";

3. The type of instances used and justification

m3.large has been used.

Because we tradeoff the budget and the deadline time. Actually the key point is the correctness of Map-reduce program to succeed by one time. To make full use of AWS resources and improve data transform flexibility, we have not made all the work done in

Map-Reduce, because we can launch an instance to do post-process in a short time. [Note] The process complexity of Map-Reduce code is also very important.

4. The number of instances used and justification

10 instances have been used. Because

5. The spot cost for all instances use

0 instance. We tried to use spot instances however, requests not succeeded with a low spot price.

6. The execution time for the entire ETL process

Nearly 2 hour for Q3, and nearly 3 hours for Q4.

7. The overall cost of the ETL process

$ 10

8. The number of incomplete ETL runs before your final run

2 runs, which include configuration related issues and the interruption because of the sudden jump in small instance.

9. Discuss difficulties encountered

Difficulties come from two aspects, the correctness of ETL and the design of the database schema. During ETL, especially in Q4, the correctness of ETL is very important. Considered that the budget and performance, we reviewed the code with team member and tested on small sample data. We have found a bug that Unicode cannot be distinguished by Linux sort command and Java code (e.g. Istanbul). Apart from that, In what format, what type the data is stored in databases decides the throughput of our system.

10. The size of the resulting database and reasoning

62GB. The original data size is 26.5GB(Q2) + 1.3GB(Q3) + 1.2GB(Q4) = 29GB. In Q2, we extract user_id, date_time and text, which have longer data. In Q3, we extract retweet_id, user_id. In Q4, we extract date_time, location, hashtag, id, rank. Because the Unicode text is very space-consuming, and when data are loaded into database, more space is needed for additional information. So the total size is 62GB.

11. The time required to backup the database on S3

The backup of database costs about 15 minutes.

12. The size of S3 backup

AMI size is the backup size.

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

Questions

1. What are the advantages and disadvantages of MySQL for each of the queries you've encountered so far? Which queries are better suited for MySQL (not HBase)?

For Query 2, it requires the database to process big data, and to return the target result with a lightning speed. To increase the speed, we have to warm up the database for a while so that we can utilize the cache of the mysql server. Therefore, for the big data test query 2, the MySQL has the advantage that it presents clear relation between different columns, but it consumes a lot of time in querying.

For Query 3, there is a column with unique value in every row, adding primary key significantly incased the query speed, but after several time of test, the throughput dropped, the reason could be the query connection occupy server's resource which cause the performance of following query.

For Query 4, there are three related columns and they have different cardinalities among the whole table, after creating the multiple column index, the query speed

can be increased. The disadvantage is that the sequence of the index argument do matter the query efficiency.

Query 3 and query 4 are better suited for MySQL, because they have smaller data size, columns are related to each other.

2. What are the advantages and disadvantages of HBase for each of the queries you've encountered so far? Which queries are better suited for MySQL (not HBase)?

3. For your backend design, what did you change from Phase 1 and why?

On the design of phase 2 backend, we were supposed to split the two column table into three column one and create a multiple column index on the table, but for some technical reason, we could not load the data into database. The reason why we want to change the indexing method is that the multiple column index can search the target result by the different cardinality according to the sequence.The optimizer will notice that the columns being used are indexed and will then construct its query plan to take advantage of that fact to arrive at a more efficiently executed query.