**S16 15619 Project Phase 1 Report**

**Team Name:**

MyLittlePony

**Members (First Name, Last Name, Andrew ID):**

Jiexin, Guo,jiexing

Wei-Yu, Yen, wyen

Yingxue, Mei, yingxuem

**Performance Data and Configurations**

|  |  |
| --- | --- |
| Best Configuration and | Results |
| Number and type of instances | Q1:4 m4.large  Q2H:4 m4.large  Q2M:5 m3.large |
| Cost per hour | Q1:0.48,Q2H:0.48,Q2M:0.665 |
| Queries Per Second (QPS) | INSERT HERE: (Q1,Q2H,Q2M)  score[100,89,100 ]  tput [33405.3, 10157.1, 14017.06 ]  latcy [2, 3, 3 ]  corr [ 100, 100, 95 ]  error [ 0.00, 0.00, 0.00 ] |
| Rank on the scoreboard: | Q1:1  Q2H:4  Q2M:4 |

**Rubric:**

**Each unanswered question = -5%**

**Each unsatisfactory answer = -2%**

**[Please provide an insightful, data-driven, colorful, chart/table-filled, and interesting final report. This is worth a quarter of the grade for Phase 1. Use the report as a record of your progress, and then condense it before sharing it with us. Questions ending with “Why?” need evidence (not just logic)]**

**Task 1: Front end**

**Questions**

1. Which front end framework did you use? Explain why you used this solution. [Provide a small table of special properties that this framework/platform provides]

Undertow httphandler and tomcat servlet. I use servlet first because it is widely used in java web framework and the information about it is sufficient.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Reliability | Loading | Performance |
| Undertow httphandler | Easy to shutdown after test for a while | Lightweight, just peek the part you needed | Better |
| Tomcat servlet | More reliable | Heavyweight compare to Undertow httphandler |  |

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

First I use the spot m4.large because it is the best machine that we can use(from the performance view), and the on-demand price of m4.large is less than m3.large. But then I found those spot price of m4.large is not stable(my machine got killed because of the price once). So I choose m3.large at the end.

1. Explain any special configurations of your front end system.

In tomcat, we set maxThread number in connector property in server.xml

In undertow, we set IO threads to 256.

1. Did you use an ELB for the front-end? Why, or why not? Condense your experience with ELB in the next few sentences. Talk about load-balancing in general and why it matters in the cloud.

Yes. Because I don't need to design and implement the load balancer by myself. What I need is just some server that has the same MySQL database and request handler.

1. Did you explore any alternatives to ELB? List a few of these alternatives. What did you finally decide to use? (if possible) Provide some graphs comparing performance between different types of systems.

I could use only one frontend as the load balancer, but use multiple MySQL connections to different backend database instances.

1. Did you automate your front-end instance? If yes, how? If no, why not?

No, because in one submission, we at most have 4 front end servers at the same time. We choose to manually connect to each of these instance and start the front end server. And we also have to spend about one hour warming up loading balancer so that the latency would reduced. Therefore, consider time and budget, we decided to finish q1 test in three to four hours. If we got the acceptable rps, we close all instance and not to do this test again. Writing automated script for instance is not cost effect and could not reuse for use.

1. Did you use any form of monitoring on your front-end? Why or why not? If you did, show us the results.

Yes, I use the top command to view the cpu and memory usage of processes.

1. What was the cost to develop the front end system?

After the end of test of these two framework, the total cost is about $5.1.

1. What are the best reference URLs (or books) that you found for your front-end? Provide at least 3.
2. <http://undertow.io/>
3. http://tomcat.apache.org
4. http://blog.csdn.net/agods/article/details/7801853
5. Please provide comparison between at least two front end frameworks. Give explanation on at least the following dimensions.

(1). CPU/Memory utilization

(2). Programming difficulty

(3). Why your current choice outstand the other one

(4). What is your RPS on these frameworks

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **CPU/Memory utilization** | **Programming difficulty** | **Why your current choice outstand the other one** | **What is your RPS on these frameworks** |
| undertow httphandler | 50/25 | medium | the performance is better | Q1:34000 |
| tomcat servlet | 30/5 | low |  | Q1:25000 |

[Please submit the code for the frontend in your code ZIP file]

**Task 2: Back end (database)**

**Questions**

1. Describe your schema. Explain your schema design decisions. Would your design be different if you were not using this database? How many iterations did your schema design require? Also mention any other design ideas you had, and why you chose this one? Answers backed by evidence (actual test results and bar charts) will be valued highly.

CREATE TABLE `tweetdata` (`useridhash` varchar(50) NOT NULL,`rawtext` text NOT NULL,INDEX index\_useridhash (useridhash) )ENGINE=myisam DEFAULT CHARSET=utf8mb4;

We connect the userid and hashtag in one column, and the rawtext column is the response for a specific request. Since one tweet will have multiple hashtag, one tweet may have more than one rows in the database. The mysql schema design changes for 3 times. We have used to store the data in columns like tweetid, date, hashtags, userid, rawtext. These designs have too many columns which makes the query very slow. So we change to a less colmuns one.

1. What was the most expensive operation / biggest problem with your DB that you had to resolve for Q2? Why does this problem exist in this DB? How did you resolve it? Plot a chart showing the improvements with time.

The query time is too long, so that our db can not get high throughput. I think this is because the data in the database is too large. So we change the database table design and add cache on our frontend. And after our cache is large enough and warmup entirely, the performance improved.

1. Explain (briefly) **the theory** behind (at least) 3 performance optimization techniques for databases. How are each of these implemented in MySQL? How are each of these implemented in HBase? Which optimizations only exist in one type of DB? How can you simulate that optimization in the other (or if you cannot, why not)? Use your own words (paraphrase).

1. cache: put all those data from disk into memory since the speed of memory is much higher than disk. This is implemented in MySQL by turning on the query\_cache.

2. index: Indexes are used to find rows with specific column values quickly by storing extra information about the data. It is implemented in MySQL using B+ trees.

3. dividing the table: we can divide the whole data into different table, so that the query's range is limited and could be faster.

The dividing table method can not be directly support by hbase, because the hbase is sharding automatically. And in HBase, we use userid as the key (and hashtags1 + raw\_texts1 + hashtags2 +raw\_texts2 + … concat as the value), so we don’t use something like secondary index on hbase.

1. Plot a graph showing results with/without each individual optimization that you used. Extremely impressive will be a timeline of rps v/s submission id (mentioning which optimization was in use at that time).

For the index part, because the database without index is too slow, so we don't have the performance of without index, and all our results are based on database with index.

We didn't have the time to do the dividing table optimization.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| optimization name | rps without this optimization | submission id | rps with this optimization | submission id |
| cache | 3239.2 | 54360 | 14754.8 | 61686 |
| index |  |  |  |  |
| dividing table |  |  |  |  |

1. Would your design work if your web service also implemented insert/update (PUT) requests? Why or why not?

Yes, because if the query's response is cached, we can change it in our cache, but write to the database later, which can maintain our performance. It is also called as write-back method, but it should be applied to the only load balancer otherwise since our database is replication, we have to do some synchronization.

1. Which API/driver did you use to connect to the backend? Why? What were the other alternatives that you tried?

JDBC driver. Because it is widely used in online guides. As I know there is hibernate that can also be used to connect to the backend database.

1. How did you profile the backend? If not, why not? Given a typical request-response for each query (q1-q2) what percentage of the overall latency is due to:
   1. Load Generator to Load Balancer (if any, else merge with b.)
   2. Load Balancer to Web Service
   3. Parsing request
   4. Web Service to DB
   5. At DB (execution)
   6. DB to Web Service
   7. Parsing DB response
   8. Web Service to LB
   9. LB to LG

How did you measure this? A 9x2 table is one possible representation.

No, because I don't know which kind of measurement can be used to measure these latencies.

1. Say you are at any big tech company (Google/Facebook/Twitter/Amazon etc.). List one concrete example of an application/query where they should be using NoSQL versus one where they should be using an RDBMS. Both examples should be based on the same company (you choose).

The company we choose to elaborate on this question is Amazon. The application that  should use NoSQL is the log of the error requests/responses or other operations not related to transaction. As the message volume of each log can vary tremendously, to store the log data in an RDBMS will lead to storage cost that is unnecessary, because an RDBMS requires each column has same data size.

The application should use an RDBMS is the transaction. As the transactions must have the feature as atomicity, which pose the strong-consistency requirement, thus the transaction process should be using RDBMS, which ensures each transaction will complete as a whole or will roll back if any step of the procedure fails, and will ensure read and write query always gets consistent results.

1. What was the cost to develop your back end system?

We spend more 20 dollars to test and improve our backend system.

1. What were the best resources (online or otherwise) that you found. Answer for both HBase and MySQL.

https://dev.mysql.com/doc/refman/5.7/en/optimization.html

<http://www.tutorialspoint.com/hbase/hbase_read_data.htm>

[Please submit the code for the backend in your code ZIP file]

**Task 3: ETL**

1. For each query, write about:
   1. The programming model used for the ETL job and justification

Mapreduce model.

* 1. The number and type of instances used and justification

1+4 m3.xlarge

* 1. The spot cost for all instances used

0.15 each, but the actual price is like 0.03 each

* 1. The execution time for the entire ETL process

10 hours and 9 minutes

* 1. The overall cost of the ETL process

25 dollars

* 1. The number of incomplete ETL runs before your final run

16

* 1. Discuss difficulties encountered

We put heavy workload in the mapper which makes it fail with memory limit exceed. And the cache-file parameter of emr job is not working, we use java network reading method to come through this problem.

* 1. The size of the resulting database and reasoning

The result of ETL is 9.6G, we put away those tweets without hashtag, and also those tweet whose tweetid appear more than once.

* 1. The size of the backup

About 24 G for each backup.

1. What are the most effective ways to speed up ETL?

Add more machines, do less thing in ETL.

1. Did you use EMR? Streaming or non-streaming? Which approach would be faster and why?

Yes, we use streaming. I think streaming is faster because it is easier to develop.

1. Did you use an external tool to load the data? Which one? Why?

No, we use the load data clause in MySQL to load the data, because our data format is standard and it can be directly load into the database.

1. Which database was easier to load (MySQL or HBase)? Why?

MySQL, because we can directly import the data csv file, and MySQL is also easier to deploy.

[Please submit the code for the ETL job in your code ZIP file]

**General Questions**

1. Would your design work as well if the quantity of data would double? What if it was 10 times larger? Why or why not?

I don't think so. We are now using replication method to maintain our database. If the data becomes 2 or more times, the scan of the whole database will cost unacceptable time. And in this situation we should use shading method rather than replication.

1. Did you attempt to generate load on your own? If yes, how? And why?

No, because we don't have enough time to implement one. But I know that we can use JMeter as the load generator.

1. Describe an alternative design to your system that you wish you had time to try.

I can store all the tweets for one userid and one hashtag in one row, and it should reduce the query time because we got exactly one answer each time.

1. Which was/were the toughest roadblock(s) faced in Phase 1?

First the format of the correct response is hard to get (a lot of small stuff needs to be taken care). Next is that we can not do much thing in the mapper procedure.

The third is to warmup and tune the backend database.

1. Did you do something unique (any cool optimization/trick/hack) that you would like to share with the class?

We make up a large scale of cache on our frontend.

We use connection pool to manage our database connections.