S16 15619 Project Phase 1 Report

Team Name: elder

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

Lei Jin (ljin1)

Jialing Liu (jialingl)

Hao Wang (haow2)

Performance Data and Configurations

Best Configuration and	Results
Number and type of instances	Q1: m3.large Q2H: master 1 m4.large, core 2 m4.large Q2M: elb: 1 m4.large and 4 m3.large
Cost per hour	Q1:5* m3.large + ELB + 32 * 5 GB volume = 0.69 Q2H: 3 * m4.large +3 * emr + 300GB volume = 0.571 Q2M: 5 * m3.large + ELB + 750GB volume = 0.70
Queries Per Second (QPS)	INSERT HERE: (Q1,Q2H,Q2M) Score 100, 25.76, 100 tput 26461.9, 2711.7, 15478.4 Latcy 3, 16, 2 corr 100, 95, 100 Error 0, 0, 0
Rank on the scoreboard:	25

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]

We used Undertow and Vertx. Undertow is very good at handling HTTP requests and have great performance for it can create threads automatically, have very small package and easy to configurate. Vertx also has a very great performance and it has its own thread pool to handle concurrent requests so we needn't to handle them by ourselves. It can pass distributed messages in a distributed way. It supports multiple programming languages and have a very small package and easy to do configuration.

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

   m3.large and m4.large makes a great balance of computing, RAM and network resources. So we used those two kinds of instances where m3.large has higher speed of CPU and m4.large has slightly more memory space.

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

   We used a very big front-end cache and changed the request buffer size to 4966 bytes we also change the java heap space to 2G for cache and ram for java programmes.

4. 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, we used ELB as front-end to automatically distribute requests to 5 back-end MySQL servers with data replication. ELB is great to balance load, which performs much better than the load-balancer implemented by ourselves. Load balancer can better make use of servers. Besides, it can also check the health of servers so as to launch a new server when necessary. If use Round-robin, it means simply allocate the requests to a server, and it can also consider other features such as CPU utilization etc.

5. 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.

   We have also tried a load balancer implemented by ourselves, but it performs much worse than ELB, for ELB has many internal optimizations that our load balancer does not have.

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

   Yes. We first construct all of configuration and codes into one instance, then we build an ami image of that instance. By creating several new instance using that ami, we can easily get the exact copies of the instances and use ELB to do load balancer.

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

   When debugging the system. We first printed all requests and the time consumption of handling the request to the shell. This can let us better know the feature of requests and the decide our design of database and whether using a front-end cache or not.

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

   We spot or on demand a m3.large to write and debug our code and use about $2.50 for whole Q1 session.

9. What are the best reference URLs (or books) that you found for your front-end? Provide at least 3.

http://vertx.io/vertx2/docs.html

http://undertow.io/undertow-docs/undertow-docs-1.3.0/index.html

http://tutorials.jenkov.com/vert.x/http-server.html

10. 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

Vertx has larger CPU/Memory utilization compared with undertow since undertow is more light weighted and vertx has many other function. In this project, we only need to handle simple http request through URL, so undertow have better CPU/Memory utilization. Because of undertow's simplicity, it has less programming difficulty compared with vertx. In addition, undertow use totally jvm environment to run, we bundle it using gradle and easily to tuning the parameters in it. On the other handle, vertx is hard to tuning the parameters if we use verticle class. As a result, our choice is undertow.

RPS:

undertow: 26461.9 vertx: 23483

[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.

MySQL schema:

PRIMARY KEY: userid_hashtag: varchar(50) allText: TEXT

HBase schema:

Row key	Column family
Userid: BINARY	allText: BINARY

Iterations:

MySQL: 54974923

HBase: 17156403

Reason: user PRIMARY KEY and Row key to search text in database is fast, since primary and row key is unique.

2. 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.

   Concurrency. We first used single thread to do query, but it is extremely slow. Then we created multi-thread by ourselves, but it is still very slow. Finally we used connection pool and it is much faster than before.

3. 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).

   MySQL
   1) Concurrency. We used connection pool to do this part.
   2) Search primary key.

   HBase has already finished currency part for us. It used key value to search result, which is just like what we do in MySQL. We changed key from String to Integer to optimize the third part.

4. 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).

   MySQL:

Submission id	Without optimization	With optimization
Connection pool	Id 68116 with 0.3 score	Id 68560 with 75.58 score
Primary key	Id 68278 with 1.19 score	Id 68359 with 60.19 score

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

   No. Our system cannot support PUT request now, since the test query does not contain any PUT up to now. Adding a put route to uri is useless in q1 and q2.

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

   We use JDBC to connect to Mysql server and Apache HBase library to connect to HBase. JDBC is part of the Java Standard Edition platform, from Oracle Corporation. It is a industry standard for database-independent connectivity between the Java programming language. Also Apache HBase library is the standard library connect with Java and HBase. We did not use any other alternatives so far, however, we may try Cloudera Hadoop API in phase

7. 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:
    a. Load Generator to Load Balancer (if any, else merge with b.)
    b. Load Balancer to Web Service
    c. Parsing request
    d. Web Service to DB
    e. At DB (execution)
    f. DB to Web Service
    g. Parsing DB response
    h. Web Service to LB
    i. LB to LG

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

8. 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).

   Use RDBMS to finished money exchange transactions, use NoSQL to store user information such as tweets, comments, etc.

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

   We use a m3.large to develop MySQL back end and two m1.large for HBase development. The total cost is about $1.

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

    For both HBase and MySQL: official documentation, Stackoverflow

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

Task 3: ETL

1. For each query, write about:
    a. The programming model used for the ETL job and justification
    b. The number and type of instances used and justification
    c. The spot cost for all instances used
    d. The execution time for the entire ETL process
    e. The overall cost of the ETL process
    f. The number of incomplete ETL runs before your final run
    g. Discuss difficulties encountered
    h. The size of the resulting database and reasoning
    i. The size of the backup

We use EMR MapReduce model for ETL job. We use 1 master m3.xlarge instance and 9 core m3.xlarge for MapReduce. Since in that time our team can not spot 20 instances, we use on demand for 20 m3.xlarge plus emr for two hour is $13.54. For load time, we use m3.4xlarge spot $0.1 per hour to load sql and 3 m4.large spot $0.1 per hour to load data into HBase. The execution for entire Mysql ETL time is 4-5 hours and HBase about 3 hours. The overall cost is about $20 for ETL process. Since we change the database schema twice, we have 2 incomplete ETL runs before my final run. When we try to load data into mysql, we used 50GB disk and encountered the the problem no free space. Finally, we added a volume of 150GB and work fine. The size of resulting data base is around 90million row nearly 25GB, since we use userid_hashtag as primary key and the exacted sorted answer as column. We use 150GB volume as the backup.

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

   MapReduce. MapReduce support process and generate large data sets with a parallel, distributed algorithm on a cluster which contains multiple machines. We use 20 m3.xlarge machines for the extract and transform step, and 3 m4.large for load the data into Hbase. Since Mysql load does not support MapReduce, it take much longer time the load the data into database.

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

   Streaming. Streaming is faster, for mapper and reducer can work nearly at the same time. If using non-streaming, then all reducer need to wait till all mapper finished their work.

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

   We did not use any external tool such as pig to load the data. Although traditional tools provided by Mysql and HBase may take long time and effort to load data, we build several AMIs to prevent multiple loading when we create new instance.

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

   MySQL. In HBase, we need to put the into HDFS to prepare for HBase loading and Prepare the HFiles for the HBase table. For MySQL, data could be loaded directly.

[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 think there should not be much different, for we used large-scale database systems to store those data, instead of saving them in plat files or RAM. So this design of front-end and back-end server is enough and it does not make much difference of performance.

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

Yes. We used a browser to send HTTP GET request, just like the load generator does, in order to see whether the result is bad or not. And observe the time consumption of handling single query. But it is hard for us to generate a large amount of concurrent queries, so it is mainly used at the beginning.

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

   We would like to try HBase on Cloudera, but we do not have enough time this time. Or use two small clusters and a ELB to handle HBase requests.

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

   I think it is very hard to improve the performance of HBase, we tried several methods, but none of them has a satisfying result.

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

   We first considered save more than 10 million files on disk, so that we can just used the name of each file to query and it is very quick. But Linux file system does not allow us to create so many files. If the data is smaller, it might be a very quick and cool solution.