S15 15619 Project Phase 1 Report

Performance Data and Configurations

Best Configuration and Results	
Number and type of instances	Q1: M3.Large and 3 instances Q2H: M3.Large and Q2M: M3.Large and 1 instance
Cost per hour	
Queries Per Second (QPS)	INSERT HERE: (Q1,Q2H,Q2M) score[20,0,0] tput [16791.2,0,0] latcy [5,0,0] corr [100,0,0] error [0,0,0]
Rank on the scoreboard:	Q1: 37 Q2H: Did you submit a snail? Q2M: Did you submit a snail?

Team : MadHatters
Members : Sahibdeep Singh, Swati Chowdhury

Rubric:

Each unanswered question = -5%
Each unsatisfactory answer = -2%

[Please provide an insightful, data-driven, colorful, chart/table-filled, humorous 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]

Servlet.

Documentation	Very Good
JSON Serialization	Very good
Latency	0.4ms (Very little lag)
Tomcat Integration	Very easy and seamless

Since we were pretty new to java web service development we wanted good documentation.

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

   Initially we started we 6 m1.large with the impression that more instances will be able to handle more requests and give the desired throughput. Scaled to 10 instances but the throughput was stuck on 12000ish. Finally used 3 m3.large instances and warmed the ELB by sending continuous requests to server and received the throughput.

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

   We used tomcat server and used Elastic Load Balancer to manage the load. On the ELB we did not scale the instances. Instead started with 3 instances flat for load management.

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. One of the major reasons was for load redistribution. Load balancing matters a lot as the server sends requests; it needs instances to manage those requests. And only one instance was not sufficient to give the throughput in our case. ELB plays a crucial role in cloud as it balances load by popping new instances and terminating unused instances. A very intuitive technique of actual instance scaling on the cloud.

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 tried with single instances and that did not work. Even the throughput maximum we could achieve was 8499.3.

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

   None.

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

   Cloud Metrics. We do not have the graphs since the ELB was closed after running.

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

   A lot.

9. What are the best references URLs (or books) that you found for your front-end? Provide at least 3.
   http://www.tutorialspoint.com/servlets/servlets-first-example.htm
   http://www.javatpoint.com/creating-servlet-in-eclipse-ide
   http://stackoverflow.com/questions/1007346/how-can-i-create-a-war-file-of-my-project-in-netbeans (took the project to netbeans for this)
   http://www.excelsior-usa.com/articles/tomcat-amazon-ec2-basic.html
   https://www.youtube.com/watch?v=siKFOI8PNKM
   https://glassonionblog.wordpress.com/2011/04/08/tomcat-redirecting-traffic-from-port-8080-to-80-using-iptables/

[Please submit the code for the frontend in your 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

   Our schema consists of a single table with one column that includes the userid and timestamp and one with the tweetid, tweet text(encoded) and sentiment score.
   3 iterations.
   Initially we kept all of these as separate columns. But then realized that each used_id can have multiple tweets at the same time.
   Updated the dB schema and kept used and timestamp as separate and combined the remaining columns into one column all delimited.
   Again updated the schema and combined the timestamp and userid.
   Also the user_id_timestamp column served as the primary key.

   HBase

   Row Key: User_Id_Timestamp (User Id + Time Stamp)
   Column Family: Tweet_Text_Score (Tweet id, text and sentiment score)

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.

   Most expensive operation was loading the data. We decided on using sharding for loading the data. 5 database nodes and the database loading was very slow. We ran the operation for 10 hours each on individual runs but could load only 3-4 files received from the EMR.

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).

   For MySQL:
   We used indexing as that would lead to faster searching.
   Also since the process was slow we made changes to my.cnf file to update the buffer size.

   For HBase
   Row Key is the primary index and data loading was fast. Therefore no other optimization was required.

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).

5. Would your design work if your web service also implemented insert/update (PUT) requests? Why or why not?
   Yes it will work with a put operation also. Since the schema handles the data request any changes will update the row.

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

   We used JDBC driver to connect to MySQL database. Since it works very will with backend service we decided to move forward with it.

   For HBase we tried HBaseConfiguration to connect to database. We are still exploring more options on this and further exploring the Apache Hadoop. Documentation.

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.) 25%
   b. Load Balancer to Web Service 40%
   c. Parsing request 10%
   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.

We monitored the MySQL process during sharding but sinces sharding did not work we did not use that approach for data loading.

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).

   Amazon e-ommerce website amazon.com should be using NoSQL since it generates a lot of data everyday that needs to be scaled in and out based on the traffic on the website.
   Amazon should be using RDBMS for storing the users that create account for AWS as that is a transactional data with manageable volume and data integrity is very crucial in

these kind of data.

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

10. What were the best resources (online or otherwise) that you found. Answer for both HBase and MySQL.
    http://samhuri.net/posts/2007/04/quickly-inserting-millions-of-rows-with-mysql-innodb/
    http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_Java.rds.html
    http://stackoverflow.com/questions/5490789/json-parsing-using-gson-for-java

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

Task 3: ETL

1. For each query, write about:
   a. The programming model used for the ETL job and justification
      For q1 since it was creating a Restful web service no ETL processing was required.
      For q2 we used Map Reduce programming model for data loading into the databases.
   b. The number and type of instances used and justification
      For q1 3 M3.Large
      For q2 20 M3.Large for EMR. 7 for HBase database setup. 1 instance for MySQL database setup with 100GB Volume.
   c. The spot cost for all instances used
      M3.Large 0.04. M1.Large 0.03.

   d. The execution time for the entire ETL process
      50 Minutes for Map Reduce job.
      3 hours HBase database loading.
      2 hours MySQL database loading.

   e. The overall cost of the ETL process
      Approximately 1.8 dollars.
   f. The number of incomplete ETL runs before your final run
      7
   g. Discuss difficulties encountered
      Major difficulties encountered during Extraction phase were preserving the tweet text since it involved a lot of special and wild characters.
      Another issue was loading the data in MySQL as database loading was taking a lot of time using database sharding technique. It was constantly giving table size exceeded error even with 100Gb of volumes attached to each database node.
   h. The size of the resulting database and reasoning
      Size of the database on MySQL was approximately 50 GB.
   i. The size of the backup

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

   The most effective way of speeding up the ETL process was using Bulk Import SQL command for MySQL.

   Also using a single instance with 100 GB volume attached really helped sped up the process for data loading in MySQL although we were a little too late when we implemented that.

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

   Yes. We used streaming process. We used two Map Reduce jobs for the extraction and transformation process where the first MapReduce output was a JSON object with the required tweet objects to be shown on front end. Another Map Reduce job was to then pick these JSON objects from files and process them into database schema format

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

   No we did not use any external tools for loading data.

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

   HBase was easier to load since it was very fast and it did not involve load distribution like we were trying initially with MySQL with database sharding.

[Please submit the code for the ETL job in your 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?
   We are still working on data scalability issues.

2. Did you attempt to generate load on your own? If yes, how? And why?
   No we did not try load on our own.
3. Describe an alternative design to your system that you wish you had time to try.

   We are still working on our existing system design.

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

   The toughest roadblock for Phase 1 was loading the data and connecting to the front end to the database especially in the case of HBase.
5. Did you do something unique (any cool optimization/trick/hack) that you would like to share with the class?

   We tweaked the My.cnf and voila! Database loading was so fast on MySQL with only one instance and 100 GB volume. We don't know if that was cool.