CSCI 5408, Winter 2017

Assignment 2: Cassandra Database and Load Balancing on Cloud

(Issue: Jan 24, Due: Feb 07, 11:59PM)

- **TA:** Abhinav Kalra (Abhinav.Kalra@Dal.Ca)
- **Ass2 Tutorial:** Jan 25, 1-2:30 PM, Room 1020, Kenneth C Rowe Mgmt. Building
- **Ass2 Help Hours:** Wed 1:00-2:30PM, CS 134; Fri, 1:00-2:30PM, Room: CS 233

---

1. Objectives:

1) To learn concepts of Distributed Database Systems running on Clouds
2) To learn load balancing and scalability on mission critical applications
3) To learn fault tolerance and performance tuning using Clouds
4) To learn Cassandra database system and firewall concepts on Clouds

Note: You are allowed to work in teams of two.

2. Tasks:

1) You need to use Cassandra Database to store a data set with at least one million tuples and should be at least 30 Megabytes in size.
2) Create an account on a Cloud service providing hosting services for Cassandra. For example, you can use one of below managed Cassandra cluster services:
    a) https://seastar.io/
    b) https://www.digitalocean.com/products/one-click-apps/cassandra/
    c) https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-linux-classic-cassandra-nodejs/
    d) https://www.instaclustr.com/managed/cassandra/
3) Select a data set to work with:
    a) Use the given Dataset with the Assignment. (metadata: https://www.opendataphilly.org/dataset/crime-incidents)
    b) Understand the dataset and preprocess, if required, before uploading to Cassandra Database
    c) Write up an application scenario based on which you will design and run your search queries in Cassandra(no insert/update or delete queries)
    d) Write four queries that will fetch results from Cassandra based upon your scenario.
4) Create a UI application that will query data from Cassandra database. The UI should be simple and intuitive. You can use a web page application, desktop or console application for this purpose (use Java/.Net or Python). UI needs to be as simple as possible. The function of UI would be to input an open text field and would search data from database. The results returned from the query need to be displayed/saved to a file along with response times.

5) Create a RESTful web service hosted on a Cloud System that acts as a broker between a UI and Cassandra database. The web service needs to query Cassandra database and make sure the cluster is up and running in addition to fetching data.
6) Processing logic for handling HTTP response codes are mentioned below:
    a) HTTP 200 OK: Should be returned when web service execution ends in success.
    b) HTTP 400 Bad Request: If any of the input parameter is either out of range or invalid.
    c) HTTP 500 Internal Server Error: If any error occurs while processing the request. Please return a generalized error message in this case.
7) In addition to querying for cluster status, the web service should also execute and return results for application scenarios you created in step 3 part d.
8) Test your web service by calling it from UI (Postman/ Your own UI application/ browser, etc.) and observe the response time from Cloud server. Observe performance (response time etc. from end to end) and comment in the report.
9) Add another instance of your REST service created in Step 5. The new REST service should follow "Resource Replication" scheme and should be an exact replica of the first service. Please test both of the server nodes as you did in Step 8.
10) **Elastic Load Balancer**: Now use load balancer and distribute load between both nodes/service instances on the cloud. Both servers needs to have same REST service deployed. By using "Multiple Threads" from UI application (JMeter/your application or any other tool), send simultaneous 10 REST service calls to the load balancer and observe response times. Increase number of threads to 20, 30, 40 and 50 and observe response time for all five sets of requests. You can use any one of the REST web service calls of your web service for this task. Create a graph with number of request on x-axis and response time on y-axis. Comment on performance and include graphs in the report.
11) Scale down the number of cloud/service instances on load balancer to one and repeat step 10. This time load will be distributed on one cloud instance only. Create a graph with number of request on x-axis and response time on y-axis. Send 10, 20, 30, 40, 50 requests in sets and observe response times.

3. Write a report including the following sections:

a) Task Description: Present the application scenario (i.e., the application and the requirements), and the DB (provide an instance of the database, and the reference source, etc.).
b) Cassandra Database Design: Provide an overview of what configuration steps are taken to setup Cassandra database on a cloud. Please use screenshots to verify implementation. Also describe the dataset to the best of your capacity.
c) Application Queries: Provide description of the search queries, web method calls, CQL syntax, sample output and response time (milliseconds) that it takes to fetch data.

d) Test Results: Describe your test results and comparisons with the help of charts and graphs
e) Summary: Provide a summary of your work & observations on the application (i.e. the DB), the developed web service and the experience of using the software tools (i.e. your comments & recommendations, etc.).

4. Submit your Ass2 report electronically:

1. Please use Bright Space to submit your assignment
2. In addition to the report, submit code and/or output files for web service and UI application in a zip file that contains a README with installation instructions.
3. Mention any additional / third party dependencies required for compilation and execution in README file.
4. Also provide scripts that you used for implementation of Cassandra database.

*** Plagiarism and Intellectual Honesty:** (http://plagiarism.dal.ca)

Dalhousie University defines "plagiarism as the presentation of the work of another author in such a way as to give one's reader reason to think it to be one's own." Plagiarism is considered a serious academic offense which may lead to loss of credit, suspension or expulsion from the University, or even the revocation of a degree.