CISC 432*/CMPE 432*

Assignment #1 – NoSQL Databases

Due Friday Oct 5, 2018 – 4pm

Overview:

NoSQL (Not Only SQL) data models encompass a range of technologies and architectures different from the relational model. They seek to solve the scalability and big data performance issues by adopting different approaches. Most of these systems sacrifice consistency for improved scalability and availability. As a result of these different technologies they present different data models and query languages. The set of current NoSQL database products include column stores, key-value stores, document stores and graph stores.

The goal of this assignment is to get familiar with some of the available open-source NoSQL database products. You may use the Microsoft Azure service or download and install the open-source NoSQL database systems (see list of systems below) on your own machine. You can also use VirtualBox software to create virtual nodes on your local machine. You are required to create databases for the data sets provided in the assignment, and use the system's query language/interface to evaluate a set of queries. You may need to use python (or java) if required. Finally, you will write a report on your experiences.

Rules:

- Teams of maximum 2 people [**Very Recommended**].
- One submission per team.
- Your submission **MUST** include:
  - A report as stated in the requirements section below. The report must have a cover letter that states: team members' names and IDs.
  - All code developed (python or java). Add the code inside a folder with the requirement number as the folder name. Add a *readme.txt* file inside each folder that states the steps to run the code.
  - All the code steps for python (or java) must have well written comments that explain what each line of code does.
  - The 60K randomly generated ratings *netIDs.dat* from Dataset#1. The one you used in requirements 1-5.
  - The results for each query per database system (Dataset#1 only).
- Follow the folder structure inside the **SubmissionTemplate** folder. Create a folder for your submission and compress it into one archive (.zip). Upload your archive to OnQ. Name the archive with your student number (**actual** number). For example:
  - studentNumber1_studentNumber2.zip (group of 2 students), eg:1432412_3412313.zip
  - studentNumber1.zip (1 student)
- Assignment archive should be uploaded by 4:00 pm on the due date. Late assignments are subject to a 10% per day late penalty, with weekends

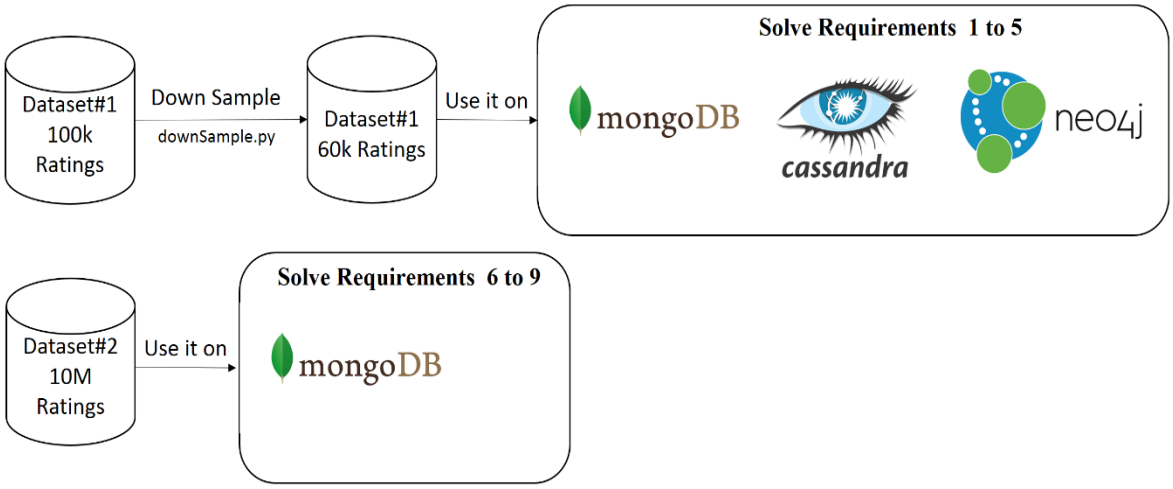

Figure 1: Assignment#1: Datasets and NOSQL Overview.

counted as one day. Late assignments will not be accepted beyond 5 days past the date due.

Data Set and Queries:

Datasets:

This assignment offers two datasets, denoted by Dataset#1 and Dataset#2. Dataset#1 includes 100K ratings of movies while Dataset#2 includes 10M ratings of movies. View the readme.txt file to understand the structure of these datasets. Each dataset has two files: movies.dat and ratings.dat.

You are required to design an appropriate database for each dataset using a NoSQL system as follows. Use the system's facilities to load the dataset into each database based on the assignment requirements.

Dataset#1:

- Document stores – **MongoDB**
- Column stores –**Cassandra**
- Graph stores – **Neo4j**

Dataset#2:

- Document stores – **MongoDB**

Queries:

Evaluate the following queries on the database:

- **Query #1:** Find the total number of reviews by each *user_id*. (netIDs.dat is only needed)
- **Query #2:** Find top-10 movies (only *item_ids*) with highest rating average. (netIDs.dat is only needed)
- **Query #3:** Find the top-10 movie titles of all movies with at least one review rating less than 3. (netIDs.dat and movies.dat are both needed).

Notice that you will do the queries above for each NoSQL system (MongoDB, Cassandra and Neo4j).

Requirements:

Write a 6 – 10 pages report (12 pt. font, double-spaced) on your experience using the NoSQL systems mentioned. The report should include the following:

Dataset#1 Requirements: (NoSQLs Systems involved are MongoDB, Casandra and Neo4j)

1. Summary of the key aspects of each database system's architecture and implementation.
2. Description of the data objects defined in each system and a discussion of the steps you performed to create and use the database and how easy/hard you found the process.
3. Discussion of how you query the database in each NoSQL system (**Must include any query language statements or code you used, for each of the three queries**). How easy/hard did you found querying each NoSQL system?
4. Discussion of the differences you found between the NoSQL systems and the relational DBMS (eg. MySql).
5. Discussion of what you liked and did not like about each NoSQL system and what kinds of applications you think the system is best suited for.

Dataset#2 Requirements: (NoSQLs Systems involved: MongoDB)

Pseudo code#1 (Performance Test)

```
Long startTime= getCurrentTime()
ConnectToMongoDB()

Start Loop n:1 to 100
Long startTimeQuery1= getCurrentTime()
Run Query1
Long TimeDiffQuery1=getCurrentTime()-startTime;

Long startTimeQuery2= getCurrentTime()
Run Query 2
Long TimeDiffQuery2=getCurrentTime()-startTime;

Long startTimeQuery3= getCurrentTime()
Run Query 3
Long TimeDiffQuery3=getCurrentTime()-startTime;

End Loop

Long TimeDiff=getCurrentTime()-startTime;
Print (TimeDiff)
```

6. Create a **ReplicaSet** configuration (setup) for MongoDB with at least 2 MongoDB instances on two different machines (or virtual machines). The two machines should have the same Specs (i.e., CPU, RAM, Operating System…, etc) as the one you used for the Single MongoDB instance (this makes the comparison in requirement #8 fair). Discuss the steps you did to apply this requirement. Write all the configurations, machine specs and the commands you used with the accurate step order. Please explain what each MongoDB command and changed configuration you did.

7. Write a python (or java) code to do a performance testing on the two setups: Single MongoDB versus the MongoDB with **ReplicaSet** (Created in requirement 6). Do ten runs of the performance testing on the Single MongoDB and another 10 runs on the MongoDB with ReplicaSet. Get the average time for each test (10 runs) then compare the results of the two in a table. The python code should run the three queries in a for loop 100 times (See the sudo code above). Discuss the time difference to run the python code on Single MongoDB against running it on MongoDB with ReplicaSet. How does the response time differ between the two setups? Explain any reasons for the differences in the performance between the two tests, if any.

8. Create a **Sharding** configuration (setup) for MongoDB with at least 2 shards on two different machines (or virtual machines). The two machines should have the same configuration specs (i.e., CPU, RAM, Operating System, …etc.) as the one you used for the Single MongoDB (this makes the comparison in requirement#9 fair). Discuss the steps you did to apply this requirement. Write all the configurations, machine specs and the commands you used with the accurate steps order. Please explain what each MongoDB command and changed configuration you did.

9. Write a python (or java) code to do a performance testing on the two setups: Single MongoDB vs. MongoDB with **Sharding**. Do ten runs of the performance test for the Single MongoDB and another 10 runs for MongoDB with Sharding. Get the average response time for each test overall and for each query within the test. Then, compare the results of the two setups in a table. The python code should run the three queries in a for loop 100 times (See the pseudo code above). Discuss the time difference to run the python code on Single MongoDB against running it on MongoDB with Sharding. How does the response time differ between the two setups? Explain any reasons for the differences in the performance between the two tests, if any.

Hints:

- Query 1 & 2 have an aggregation. Query 3 is a join.
- The average response time for each query includes 1000 (10X100) data points.
- The average response time for the overall run includes only 10 data points.