MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

Cluster Setup & Sharding

Student Name: Laiba Asif

Student ID: R00201303

Module: NoSQL Data Architectures

Date: 10/05/23

To set up the distributed environment using MongoDB scripts

1. Change the name of the machine in the scripts to the name of the machine you are working on.
   - Use a text editor to view each script file ('create_nodes.bat','setup_cluster.bat', 'insert_collection.bat','shard_collection.bat', and'remove_cluster.bat').
   - Search for the computer's name and replace all instances with the name of my own machine.
   - Save the script files after making changes.

2. Reboot the machine before starting the process

Before beginning the setup procedure, restart the computer to check that it is in a clean state.

3. C:

- On your computer, launch the command prompt.

4. check a single mongod server will start

- Enter the character "C:" to switch to the C: drive.

5. Find logical lab machine name - this PC −→ properties then copy computer name

- Run the command to launch a "mongod" server, and then verify that it launches successfully.
- This step makes that MongoDB is correctly installed and running on your computer.

6. Change the directory paths in mkdir commands to a directory where you have write permissions on the C drive in the lab machines.

- To find the name of computer, open "This PC" and go to "Properties."
- copy it.

7. CD to the directory on the C: drive where you store the scripts

- Find the'mkdir' commands that create folders in each script file.
- On computer's C: drive,I modify the directory paths to point to a directory in which I wrote permissions.
- This procedure makes sure that the required directories are created in the proper location.

8. Run remove_cluster.bat script first to ensure clean environment

- Launch a fresh command window.
- Switch the directory to the location where the script files are kept.

9. Run create_nodes.bat from cmd terminal

- Run the "remove_cluster.bat" script before configuring the cluster to ensure a clean environment.

- Any existing MongoDB cluster configurations and data will be deleted by this script.

10. Verify processes up

- Run the 'create_nodes.bat' script to add the cluster's required nodes.
- The configuration servers and replica sets will be created using this script.

11. Run the remaining scripts from a different cmd terminal

- Verify the script's output and see if the MongoDB processes are active.

- Verify that all of the configuration servers and replica sets have started up correctly.

12. make sure output from scripts indicate shards created

- Run each of the other scripts ('setup_cluster.bat', 'insert_collection.bat','shard_collection.bat') one at a time.
- The cluster will be configured, data will be inserted into the MongoDB collection, and the collection will be sharded across the replica sets using these scripts.

13. After all scripts have been executed start a Mongo client in a new cmd terminal and query some of the data

- Launch a new command prompt.
- Launch a MongoDB client by typing the proper command (such as "mongo"), then connect to the cluster.
- Run queries to retrieve data and respond to inquiries about the preferred foods of New Yorkers, available market prospects, and rivals.

14. After your cluster is functioning, add block comments in the script.

- Once the cluster is operating properly, I add block comments to the script files to record the procedures followed and any observations or adjustments made during the setup process.

15. Use the cluster to answer the following queries.

(a) Which particular cuisines do New Yorkers prefer?

```
db.restaurants.aggregate([
 { $group: { _id: "$cuisine", count: { $sum: 1 } } },
 { $sort: { count: -1 } }
])
```

This query counts the number of each cuisine and organises the documents according to the "cuisine" column. The result is then sorted based on count in descending order. The top cuisines with the largest count reveal New Yorkers' preferences.

(b) Which areas represents the biggest market opportunities for opening new restaurant for particular cuisines?

```
db.restaurants.aggregate([
 { $group: { _id: { cuisine: "$cuisine", borough: "$borough" }, count: { $sum: 1 } } },
 { $sort: { count: -1 } }
])
```

This query counts each combination of the "cuisine" and "borough" attributes, groups the documents by those counts, and then sorts the results in descending order. The top combinations correspond to the regions where there are the most potential markets for particular cuisines.

(c) Who would be the biggest competitors in these area?

```
db.restaurants.aggregate([
 { $group: { _id: { cuisine: "$cuisine", name: "$name" }, count: { $sum: 1 } } },
 { $sort: { count: -1 } }
])
```

This query groups the documents by the intersection of the "cuisine" and "name" fields, counts each intersection, and then sorts the output in descending order. The most competitive combinations in each market are indicated by the top combos.

16. This script only works with MongoDB version 3.0 in windows machines. Modify the script so that it works with the newest version of MongoDB in windows machines. What changes we need to make so that it works in Linux as well

- Examine the script files to find any commands or configurations that might have been modified for the more recent MongoDB version.
- Modify the scripts as necessary to maintain compatibility

Explain what is happening when the scripts are executing.

I set up a simulated distributed MongoDB cluster on my machine by running the scripts; I then import data, partition the collection, establish replica sets, and query the data to analyse information about New York restaurants.

Prior to making any edits or executing the scripts, its important to routinely save the progress and backup any vital data.

1. Replica Set Names:

Each replica set name should have "R00201303(my student ID)" appended to the end of it. For instance:

- cf g0 -> cf g0_R00201303
- cf g1 -> cf g1_R00201303
- cf g2 -> cf g2_R00201303
- dublin0 -> dublin0_R00201303
- dublin1 -> dublin1_R00201303
- dublin2 -> dublin2_R00201303
- cork0 -> cork0_R00201303
- cork1 -> cork1_R00201303
- cork2 -> cork2_R00201303
- limerick0 -> limerick0_R00201303
- limerick1 -> limerick1_R00201303
- limerick2 -> limerick2_R00201303
- galway0 -> galway0_R00201303
- galway1 -> galway1_R00201303
- galway2 -> galway2_R00201303

2. Port Numbers:

For each node in the replica sets, use the following ports:

- cf g0: 26050
- cf g1: 26051
- cf g2: 26052
- dublin0: 27000
- dublin1: 27001
- dublin2: 27002
- cork0: 27100
- cork1: 27101
- cork2: 27102
- limerick0: 27200

- limerick1: 27201

- limerick2: 27202

- galway0: 27300

- galway1: 27301

- galway2: 27302

3. Shard Key:

- After configuring the cluster, you must run the necessary commands to shard the "restaurants" collection using the shard key "cuisine", "borough".
- To enable sharding on the "nobeldb" database, connect to the MongoDB instance and issue the command once the cluster has started.

  "nobeldb"); ''' sh.enableSharding; '''

- After enabling sharding, use the following command to shard the "restaurants" collection with the supplied shard key:

  sh.shardCollection("nobeldb.restaurants", "cuisine": 1, "borough": 1) '''

  Prior to executing any queries, make sure to run this command after inserting the collection.

update the script files and configurations with the revised replica set names and port numbers mentioned above.