Exercise 04.02: Run Cassandra-Stress

In this exercise, you will:

- Learn to use `cassandra-stress`
- Simulate a workload on your cluster

Background

You are beginning to understand how the KillrVideo schema affects cluster performance. You decide to start to improve performance by analyzing the first table, which is the 'user_by_email' table. Imagine this table has the following schema:

```
CREATE TABLE users_by_email (
   email TEXT,
   password TEXT,
   user_id UUID,
   PRIMARY KEY ((email))
);
```

You realize that the password should not be stored as plain-text, so you make a mental note to talk to the schema designer, but you decide to go ahead and do your analysis as-is for now.

You decide that emails usually consist of a minimum of 8 characters and a maximum of 30 characters uniformly distributed. The password will be a minimum of 8 characters and a maximum of 50 characters (50 for the truly paranoid). However, most of the passwords are skewed to the length of 8. You anticipate that for every additional user that registers, the KillrVideo website will see 10 others login. You also anticipate that, of about a million users, some users will be more active than others, so they will login much more frequently.

You decide to simulate this profile and see what you can learn about this simple schema.

Steps

For this exercise, we will run the `cassandra-stress` command on a separate cloud instance. We do this to keep the impact of the test process separate from the impact of the Cassandra node.

1. In the DSE-node2 window, navigate to the directory */home/ubuntu/labwork*. Using your favorite text editor open the file *TestProfile.yaml*. This file contains a cassandra-stress user profile for a blogpost schema. You can use this profile to guide you, but you will want to modify it to suit your purposes.
2. In the schema section of *TestProfile.yaml* modify the two keyspace lines to name and create the 'killr_video' keyspace. Use 'SimpleStrategy' and a 'replication_factor' of 1.

```
#
# Keyspace Name
#
keyspace: killr_video
keyspace_definition:
CREATE KEYSPACE killr_video WITH replication = {'class': 'SimpleStrategy',
'replication_factor': '1'};
```

3. Below the keyspace within the schema section of *TestProfile.yaml*, modify the table lines to name and create the 'user_by_email' table. Use the table described in the background section above as a reference.

```
#
# Table name and create CQL
#
table: user_by_email
table_definition: |
  CREATE TABLE user_by_email (
    email TEXT,
    password TEXT,
    user_id UUID,
    PRIMARY KEY ((email))
  )
```

4. Find the 'columnspec:' section of the file. Modify this section to describe how to generate the data for each of the three (3) columns in the table. Use the information in the background section above to help you understand how to generate the data.

```
#
# Meta information for generating data
#
columnspec:
  - name: email
    size: gaussian(8..30)
    population: exp(1..1000000)
  - name: password
    size: exp(8..30)
    population: uniform(1..1000000)
  - name: user_id
    size: fixed(4)
    population: uniform(1..1000000)
```

5. Find the 'insert:' section of the file. Modify this section so that each user record has its own partition. You should use an UNLOGGED batch type. Also, the 'select:' setting should be `fixed(1)`, meaning that each insert will create exactly one row per partition.

```
#
# Specs for insert queries
#
```

```
insert:
  partitions: fixed(1)
  batchtype: UNLOGGED       # use unlogged batches
  select: fixed(1)/1
```

6. Find the 'queries:' section of the file. You will create a single query named 'user_by_email:'. The query should select all columns from the table based on the email address.

```
#
# Read queries to run against the schema
#
queries:
   user_by_email:
      cql: select * from user_by_email where email = ?
      fields: samerow
```

7. Save your modifications to the *TestProfile.yaml* file and exit the editor.
8. In the DSE-node1 window, verify your single node cluster is running by executing the following command:

```
nodetool status
```

9. Back in the DSE-node2 window, run the stress test using the following command (you will need to fill in the necessary fields):

```
cassandra-stress user profile=/home/ubuntu/labwork/TestProfile.yaml
ops\(insert=<YOUR_INSERT_COUNT_GOES_HERE>,user_by_email=<YOUR_QUERY_COUNT_
GOES_HERE>\) -node DDAC-node1
```

NOTE: This command is one long line (not multiple lines as shown by the wrapping).

Observe the output of the test as it runs. This test will take several minutes to complete. The test will begin by running the inserts and queries described in *TestProfile.yaml* with four (4) concurrent threads. You will see output about every second or so describing how many of each operation were performed as well as statistics about operation duration.

After the tests runs with four (4) threads, the test will delay for 15 seconds to allow the cluster to quiesce. Then, the test will run again with more threads. The test will increase the number of threads until the throughput of the cluster no longer increases.

At the end of the test, you will see a summary table where each row describes the results of a different number of threads.

NOTE: While running cassandra-stress, you may encounter a message that says, "Failed to connect over JMX; not collecting these stats." This is acceptable; `cassandra-stress` can be configured to gather additional stats using the JMX connection, but we will not do that in this exercise.

END OF EXERCISE