Exercise 15 – Read Repair

In this exercise, you will:

- Understand how Apache Cassandra™ performs read-repairs on inconsistent data.

Consistency is the tricky challenge for distributed systems. As distributed systems trade-off consistency for performance, some of the nodes in a cluster may become inconsistent. When Apache Cassandra™ notices these inconsistencies, Apache Cassandra™ takes steps to resolve the consistencies. This resolution is the role of Read-Repair.

Steps

1) Ensure that all three of your nodes are up and running.

2) We are going to bring down one of the nodes responsible for the `cassandra` replica. As a review, the following command will tell you what nodes these are:

   ```
   /home/ubuntu/node1/resources/cassandra/bin/nodetool getendpoints
   killrvideo videos_by_tag 'cassandra'
   ```

   Be sure you have these written down.

3) Choose one of the nodes to bring down. Before bringing the node down, flush its data by executing the following command:

   ```
   /home/ubuntu/nodeX/resources/cassandra/bin/nodetool drain
   ```

4) Now bring down your chosen node responsible for the `cassandra` replica. Wait for the node to terminate before continuing.

   Keep track of which node you brought down.

5) In the `/home/ubuntu/nodeX/data/data` folder of the downed node, find the directory that contains the table data for `videos_by_tag`. Delete the entire directory.

   ```
   ubuntu@ds201-node1:~/node2/data/data/killrvideo$ ls -l
   total 4
   drwxrwxr-x 3 ubuntu ubuntu 4096 Apr 11 15:59 videos_by_tag-
   fb88cf911e0011e7aa5bb3ed7c3998b1
   ```

```
ubuntu@ds201-node1:~/node2/data/data/killrvideo$ rm -rf
videos_by_tag-fb88cf911e0011e7aa5bb3ed7c3998b1/
```

6) Start `cqlsh` on one of your up and running nodes. Switch to the killrvideo keyspace. Ensure that your consistency level is set to ONE.

```
CONSISTENCY ONE;
```

7) Execute the following query. What results do you expect back?

```
SELECT *
FROM killrvideo.videos_by_tag
WHERE tag = 'cassandra';
```

We get all the data back because the replica node for the `cassandra` partition is up and still has the data. The query did not fail because our consistency level is ONE.

8) Take down the other node that is responsible for the `cassandra` partition in the `videos_by_tag` table. Wait for it to terminate before continuing.

9) Now bring up the downed node (the one you deleted the data files for). Wait for the node to come up before continuing.

10) In `cqlsh`, execute the query that follows. What do you think the results will be?

```
SELECT *
FROM killrvideo.videos_by_tag
WHERE tag = 'cassandra';
```

The result is empty because we deleted the data files on that node for `videos_by_tag`.

11) Bring up the other downed node. Wait for it to come online. Keep track of which node this is as we will take it down again shortly after triggering a read repair. Remember that this node you are bringing up still has all our data for the `cassandra` partition.

12) In `cqlsh`, set your consistency level to two.

```
CONSISTENCY TWO;
```

A consistency level of two will cause Cassandra to read both replicas, perform the checksum diff, and notice the data is not in sync between the two nodes. Apache Cassandra™ will then invoke a read repair to repair the node from which we deleted the data.

13) Execute the following query:

```
SELECT *
FROM killrvideo.videos_by_tag
WHERE tag = 'cassandra';
```

Our data is back.

14) Now bring down the node you just barely brought up--the node which you did NOT delete the data files for `videos_by_tag`.

15) In `cqlsh`, ensure your consistency is ONE.

```
CONSISTENCY ONE;
```

16) Execute the following query.

```
SELECT *
FROM killrvideo.videos_by_tag
WHERE tag = 'cassandra';
```

Note that, this time, we get our data from this node because the previous invocation of the query caused a read repair.