Lab 10 – Cassandra

CC5212-1

August 6, 2015

So today we finally get to play around with a NoSQL store! In particular, we will play around with Apache Cassandra, which is a column family/tabular store. We will have a look at adding and removing keyspaces, adding and removing tables, managing schema and types, how data are inserted and queried, etc. We are going to use the Cassandra Query Language (CQL) interface, which is a bit like a lightweight version of SQL. You may find the documentation here helpful: `https://cassandra.apache.org/doc/cql3/CQL.html`.

You can use Linux or Windows, as you prefer (you just need SSH today). To start with, log into the cluster (username `uhadoop`, password on the board).

- You can use the following commands to get some info on the Cassandra nodes:

```
/data/hadoop/cassandra/bin/nodetool -h 10.10.10.1 info
/data/hadoop/cassandra/bin/nodetool -h 10.10.10.11 info
/data/hadoop/cassandra/bin/nodetool -h 10.10.10.12 info
/data/hadoop/cassandra/bin/nodetool -h 10.10.10.13 info
```

- Next we are going to launch the CQL (Cassandra Query Language) client. Run the following command (the first argument is the IP of the server, the second is the port):

```
/data/hadoop/cassandra/bin/cqlsh 10.10.10.1 9160
```

- Now we first create a keyspace and some tables.
  - First we will create a keyspace with a simple replication strategy (hashing on key) and 3 copies:

```
CREATE keyspace [yourusername] with replication =
{'class' :  'SimpleStrategy', 'replication_factor' :  3 };
```

    We can see the list of keyspaces:
    `DESCRIBE keyspaces;`
  - Next we want to move to that keyspace: `USE [yourusername]`
  - Now we can *try* create a table in that keyspace with two columns, one called username of type text, one called age of type int:
    `CREATE TABLE [someuniquename] ( username text, age int);`
    But wait, that didn't work! Remember Cassandra tables are really maps and every map needs keys. We need to specify a primary key, which gives a unique identifier for each row:
    `CREATE TABLE [someuniquename] ( username text PRIMARY KEY, age int);`
    That's better. Note that if you are not in a given namespace, you can still reference a table using `namespace.table`. If you are inside the right namespace, you can just use the table name.
  - Now we can list the tables in the keyspace we are in:
    `DESCRIBE tables;`
    On second thought, let's add an alive column to the table:
    `ALTER TABLE [someuniquename] ADD alive boolean;`

  - Okay, now you have a taste for how to create keyspaces and tables. After all your hard work above, you're going to delete everything you just made. Please only drop your stuff! We will have more interesting data if we work from now on in one common keyspace/table. `DROP TABLE [someuniquename];`
    `DROP KEYSPACE [yourusername];`

- Let's move to a common keyspace I made earlier where we can see how to update and query data.
  - First load the keyspace: `USE students;`
  - Next, see what tables there are: `DESCRIBE tables;`
  - What's inside the table? `SELECT * FROM students.cc5212;`

- Now you will add your own data. In the following, I use my own data as an example. *Please replace my details with yours.* Please don't overwrite my row. If you do, I will be sad.
  - First we will insert a name with initials:
    `INSERT INTO students.cc5212( username, name ) VALUES ( 'ahogan', 'A. Hogan');`
    Query the table again to make sure it's in there: `SELECT * FROM students.cc5212;`
  - What happens if we try to insert a bad type:
    `INSERT INTO students.cc5212( username, name ) VALUES ( 'ahogan', true);`
    (Doesn't work! Type of `name` is text, not boolean.)
  - Now let's try give our full name instead:
    `INSERT INTO students.cc5212( username, name ) VALUES ( 'ahogan', 'Aidan Hogan');`
    If you query the table again, you will see that the old value is overwritten. Unlike BigTable/Dynamo, etc., values are not versioned in Cassandra.
  - Let's try add more information, like our age:
    `INSERT INTO students.cc5121( age ) VALUES ( 30 );`
    Doesn't work? Why not? (Whose age should we update? We need to give the key!)
    `INSERT INTO students.cc5212( username, age ) VALUES ( 'ahogan', 30 );`
    Try query the table again. We see that age is added alongside name.
  - Fill in the rest of the details in the table for yourself following the above idea.

- Okay, so let's try some queries over the table now; hopefully it's filling up.
  - Let's get the number of folks in the table:
    `SELECT COUNT(*) AS student_count FROM cc5212;`
  - The name of a given user:
    `SELECT name FROM cc5212 WHERE username = 'ahogan';`
  - People who are a given age: `SELECT name FROM cc5212 WHERE age = 30;`
    Oops, didn't work.

- Actually we are very limited in what we can do by default in Cassandra. Like a map, we can only really do look ups by keys: in this case, the primary key is username. To do more, we need to create indexes.
  - As a first step, you should make a personal copy of the table by saving it to a file and reloading it (all one command here):
    `COPY students.cc5212 (username, age, fav_colour, fav_movie, female, message, name) TO '[myname].csv';`
  - Now create a new table with your username that fits the data, and load the data from the file:
    `COPY students.[myuername] (username, age, fav_colour, fav_movie, female, message, name) FROM '[myname].csv';`
    This is your own copy of the table to play with.

- Let's say we're building an application that needs to lookup users with a given age. In Cassandra, we need to build an index for that.
  `CREATE INDEX ageIndex ON students.[myuername] (age);`
- Now we can try find users by age:
  `SELECT name FROM [myuername] WHERE age = 30;`
  Okay great, now it works!
- Now try to figure out how to get the following query to work (check out the documentation linked at the top):
  `SELECT name FROM [myuername] WHERE age > 25 AND female = false;`

As we see, the default query power of Cassandra is very limited. This can be improved by adding default indexes, up to a point. The scalability of Cassandra trades off against its flexibility.