CASSANDRA OPTIONAL BIG DATA PROJECT
USER GUIDE
CMPUT 391 LEC H1

Prepared by Steven Myers, Michael Kmicik, & Divyank

Installation Guide

Step 1: Download version 2.1.3 of apache-cassandra.
http://archive.apache.org/dist/cassandra/2.1.3/apache-cassandra-2.1.3-bin.tar.gz

Step 2: extract the apache-cassandra-2.1.3 folder.
tar -xvzf apache-cassandra-2.1.3-bin.tar.gz

Step 3: make the required directories.
cd apache-cassandra-2.1.3
sudo mkdir data
cd data
sudo mkdir data
sudo mkdir commitlog
sudo mkdir saved_caches
cd ..

Step 4: edit the cassandra.yaml file found in the conf folder.

4a) set seed node (line 226):
- seeds: “<ip of seed node>”

4b) set listen address (line 372):
listen_address: “<ip of current node>”

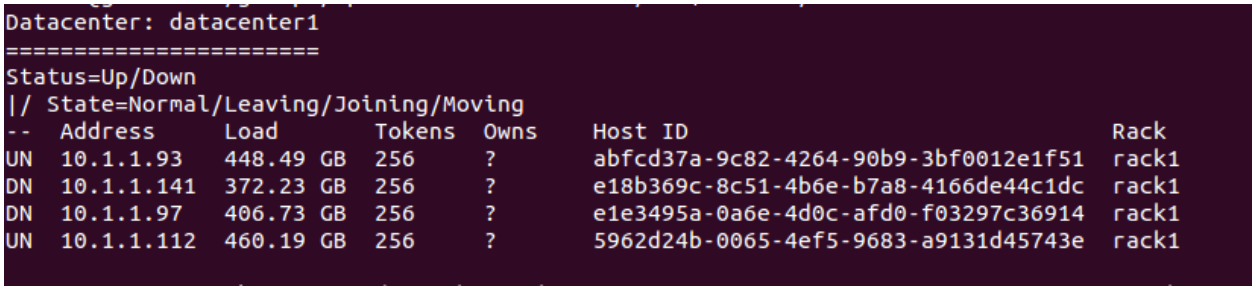
4c) set rpc_address (line 414):
rpc_address: 0.0.0.0

Step 5: repeat the previous steps for as many nodes as needed.

Step 6: start each Cassandra node
cd bin
sudo ./cassandra

Step 7: verify cluster
cd bin
sudo ./nodetool status

At this point you should see something similar the following screenshot, but with arbitrary load values:



```
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address      Load       Tokens  Owns    Host ID                               Rack
UN  10.1.1.93    448.49 GB  256     ?       abfcd37a-9c82-4264-90b9-3bf0012e1f51  rack1
DN  10.1.1.141   372.23 GB  256     ?       e18b369c-8c51-4b6e-b7a8-4166de44c1dc  rack1
DN  10.1.1.97    406.73 GB  256     ?       e1e3495a-0a6e-4d0c-afd0-f03297c36914  rack1
UN  10.1.1.112   460.19 GB  256     ?       5962d24b-0065-4ef5-9683-a9131d45743e  rack1
```

Create Database

Step 0: If the codebase is not downloaded, then clone the following git repo:
https://github.com/SMD-Cassandra-391/project.git

Step 1:
sudo ./<path to cassandra folder>/bin/cqlsh -f /<path to big-data project>/CQL/project_tables.cql

Populate Database

Building the project is most easily done in an Eclipse environment. One should **import the project as an existing Maven project**, and then simply **run as Maven install**. Two Jar files will then be generated in the target folder. The Jar file with the suffix **one-jar.jar** is a Jar file that runs the data generation application and also includes the required dependencies.

DATA GENERATION APPLICATION USAGE

The data generation program takes two arguments:

1. The keyspace for the SSTables to be written. This argument must be **'demo'**, **'project'**, **'modified_demo' or 'modified_project'**.
2. The number of rows per SSTable. The value **'1000000'** was used in experimentation to populate the **'project'** tables and **'100000'** was used to populate the **'modified_project'** tables. These values also determine the sequence of randomly generated values.

Furthermore, various commands must be utilized in order to run the application independent of **ssh** connection and to feed the stdout text into the null space of the machine. A sample invocation is as follows:
nohup java -jar DatabaseGenerator-1.0-SNAPSHOT.one-jar.jar project 1000000 >/dev/null 2>&1 &

At this time one should record the pid of the process, because currently there is no termination to SSTable generation and loading. One must kill the process manually when there is enough data loaded into the cluster to test queries.

Query Execution

The queries are evaluated using the database-test.one-jar.jar file. One must indicate which set of queries to run at command line invocation.

sudo java -jar database-tester-0.0.1-SNAPSHOT.one-jar.jar <query set>

query set must be one of: **'demo', 'project', 'modified_demo'** or **'modified_project'**.

< Include UI usages >