Apache Cassandra

Download the latest Cassandra binaries from apache site.

```
$ cd ~/Downloads
$ wget http://redrockdigimark.com/apachemirror/cassandra/3.9/apache-cassandra-3.9-bin.tar.gz
$ tar xf apache-cassandra-3.9-bin.tar.gz
```

Single Cassandra Node Setup

Launch cassandra server

```
$ cd apache-cassandra-3.9-bin
$ bin/cassandra -f
```

Launch cqlsh. Create a table for stocks data and load the stocks data. You can get the stocks data from this link.

```
$ bin/cqlsh
cqlsh:lab> create keyspace lab with replication = {'class': 'SimpleStrategy',
'replication_factor': 1} ;
cqlsh:lab> create table stocks(date timestamp,open double,high double,low
double,close double,volume double,adjclose double,symbol text, primary
key((symbol), date)) with clustering order by (date DESC);

cqlsh:lab> copy stocks(date,open,high,low,close,volume,adjclose,symbol) from
'~/Downloads/stocks.csv' with header = true and DATETIMEFORMAT = '%Y-%m-%d';

cqlsh:lab> select * from stocks where symbol = 'GE' and date < '2016-01-01Z' limit
10;
```

Specifying timeout in cqlsh

```
 --connect-timeout=CONNECT_TIMEOUT
                       Specify the connection timeout in seconds (default: 5
                       seconds).
 --request-timeout=REQUEST_TIMEOUT
                       Specify the default request timeout in seconds
                       (default: 10 seconds).
```

Export Schema

```
$ bin/cqlsh -e "DESC KEYSPACE user" > user_schema.cql
$ bin/cqlsh -e "DESC SCHEMA" > db_schema.cql
```

Compaction Behaviour

Major Compaction

Flush memtables save the data in memtable to SSTables. After flushing the memtable and commit logs are reset.

```
$ bin/nodetool flush
```

View

```
$ ls -lh data/data/lab/stocks-<hex coded id>/
```

Run compaction

```
$ bin/nodetool compact
```

View SSTables again. You should less number of files with larger size ... compared to the total of old files of same category such as data

```
$ ls -lh data/data/lab/stocks-<hex coded id>/
```

Minor Compaction

Minor compaction is triggered automatically by cassandra when it finds SSTables count > min_compaction_threshold (default - 4). To observe the behaviour, create a 4 insert operation in cqlsh followed by a nodetool flush and monitor the SSTables.

GC_Grace Behavior

When you mark a column on a row for deletion, Cassandra put a tombstone marker marked_deleted to the record or column. During compaction process the tombstones are removed. So, that the same record does not propagate from other nodes, Cassandra keep the tombstones for a grace period (default GCGraceSeconds 10 days). After this period, the tombstone actually expires and following compaction will actually remove data.

To see the tombstone in SSTable,
1. delete a row using cqlsh
2. bin/nodetool flush
3. bin/nodetool conpact
4. tools/sstabledump to view the content of the SSTable. You should the deleted the record in the SSTable dump.

Reduce the grace period to 30 secs and restart the node. Run a compact so see the records is gone.
5. bin/nodetool compact

Determine the total number of SSTables for each table.

$ nodetool tablestats.

Get the number of SSTables consulted for each read.
$ nodetool tablehistograms.

A median value over 2 or 3 is likely causing problems.

SSTables

View sstables
```
$ bin/sstableutil lab customer -t all
```

Dump the content of a SSTables in json format. Below the path is just a sample.
```
$ tools/bin/sstabledump data/data/lab/customer-dcd89670c04f11e6a2ee2f10c3a11575/mc-3-big-Data.db
```

Timestamp

Each time you write data into Cassandra, a timestamp is generated for each column value that is updated. Internally, Cassandra uses these timestamps for resolving any conflicting changes that are made to the same value. Generally, the last timestamp wins.

Find timestamp for a column. Timestamp is not available for primary key columns since those are not mutable.
```
cqlsh:lab> select pan, name, writetime(name) from customer;
```

You can send a write request with timestamp.
```
cqlsh:lab> update customer using timestamp 1481536019673255 set name = 'namo' where pan = 'xyz123'
```

Commit Log Behavior

An approach for improving commitlog performance is to pre-allocate the full 32MB segment files and reuse them once all the mutations have been flushed. The amount of data can potentially include commit logs from multiple table. Pre-allocation allows writes to be performed without modifying the file size metadata, and should (in theory) allow the file system to allocate a contiguous block of space for the file. Recycling the segment files prevents the overhead of pre-allocation from impacting overall performance. To see the behaviour, using cqlsh insert a row in a table, and keep monitoring the updated timestamp to seconds level. Remember, the size will remain same, only the update time stamp will change.

```
$ cd $CASSANDRA_HOME
$ ls -lh --time-style=full-iso data/commitlog/
```

Columns with TTL

Use case: you want to automatically purge data after a duration (specified in seconds).

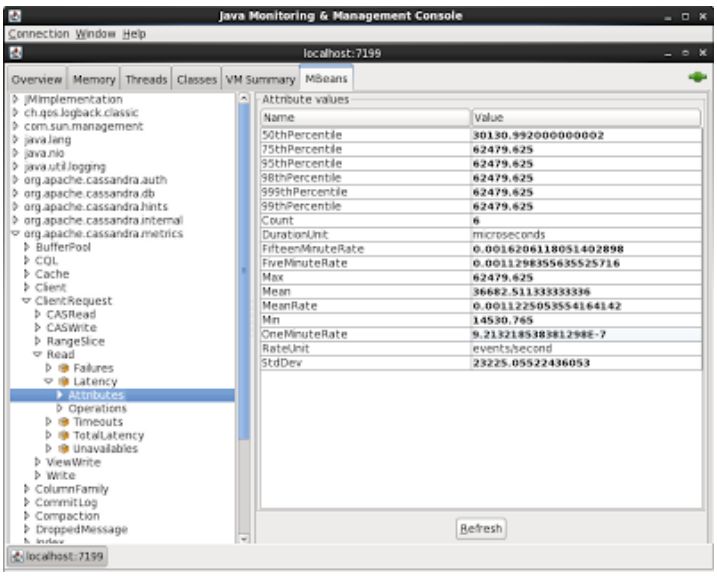
```
cqlsh:lab> CREATE TABLE latest_temperatures ( weatherstation_id text, event_time
timestamp, temperature text, PRIMARY KEY (weatherstation_id,event_time), ) WITH
CLUSTERING ORDER BY (event_time DESC);

cqlsh:lab> INSERT INTO
latest_temperatures(weatherstation_id,event_time,temperature) VALUES
('1234ABCD','2013-04-03 07:03:00','72F') USING TTL 20;

cqlsh:lab> select * from latest_temperatures;
```

Explore JMX

```
$ jconsole localhost:7199
```

- org.apache.cassandra.db: This includes caching, table metrics, and compaction
- org.apache.cassandra.internal: These are internal server operations such as gossip and hinted handoff
- org.apache.cassandra.metrics: These are various metrics of the Cassandra instance such as cache and compaction
- org.apache.cassandra.net: This has Inter-node communication including FailureDetector, MessagingService and StreamingService
- org.apache.cassandra.request: These include tasks related to read, write, and replication operations

Example: Find avg read latency from across Cassandra cluster.



Explore Logs

Cassandra maintains node level logs at each node with name system.log. You can change the logging level by setting log4j.properties. You can permanently set logging level or alter the log level during a session itself using bin/nodetool command.

Understanding the directory structure for tarball installation.

Configuration and sample files	Locations	Description
`cassandra.yaml`	*`install_location`*`/conf`	Main configuration file.
`cassandra-env.sh`	*`install_location`*`/conf`	Linux settings for Java, some JVM, and JMX.
`jvm.options`	*`install_location`*`/conf`	Static JVM settings for heap, garbage collection, and Cassandra startup parameters.
`cassandra.in.sh`	*`install_location`*`/bin`	Sets environment variables.
`cassandra-rackdc.properties`	*`install_location`*`/conf`	Defines the default datacenter and rack used by the GossipingPropertyFileSnitch, Ec2Snitch, Ec2MultiRegionSnitch, and GoogleCloudSnitch.
`cassandra-topology.properties`	*`install_location`*`/conf`	Defines the default datacenter and rack used by the PropertyFileSnitch.
`commit_archiving.properties`	*`install_location`*`/conf`	Configures commitlog archiving.
cqlshrc.sample	*`install_location`*`/conf`	Example file for using cqlsh with SSL encryption.
`metrics-reporter-config-sample.yaml`	*`install_location`*`/conf`	Example file for configuring metrics in Cassandra.
`logback.xml`cat	*`install_location`*`/conf`	Configuration file for logback.
`triggers`	*`install_location`*`/conf`	The default location for the trigger JARs.

Source: https://docs.datastax.com/en/cassandra/3.x/cassandra/install/referenceInstallLocateTar.html

SSTables structures
https://wiki.apache.org/cassandra/ArchitectureSSTable

Threadpool Stats

Use nodetool tpstats to collect thread pool information

Run tpstats to collect thread pool statistics. The subcommand tpstats displays the number active, pending, and completed tasks for each of the thread pools that Cassandra uses for stages of operations. A high number of pending tasks for any pool can indicate performance problems.

In tpstats, pending numbers that don't back down indicate high utilization. Blocked or All Time blocked numbers indicate saturation. Dropped messages indicate errors, probably resulting from maximum saturation
`~/Downloads/cassandra11/bin/nodetool tpstats`

View Latency

See the read latency for user table in demo keyspace

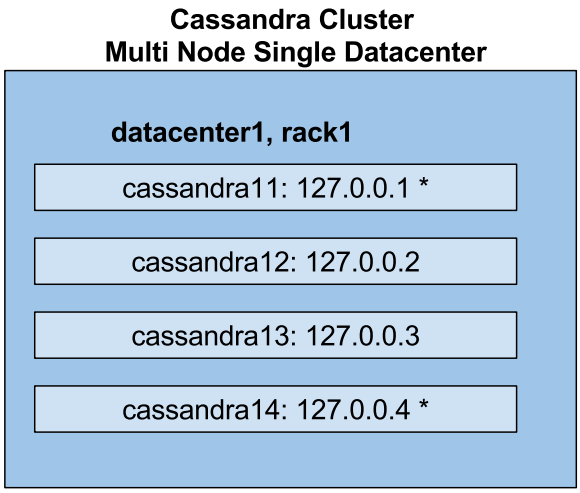
```
$ bin/nodetool cfstats demo.user
```

Learning objectives from the following exercise

1. Create a multi node Cassandra cluster - single data center and multiple data center
2. Add / remove / replace nodes from a cluster
3. How fault tolerance is achieved in cases of server failures
4. How to use consistency levels for transactions
5. Read repair
6. Hints
7. Add/remove seed nodes
8. How to add a new cluster

Multi Node Single Data Center

Cassandra Multi Node Single Data Center



cassandra.yaml

```
num_tokens: 1
seeds: "127.0.0.1,127.0.0.4"
listen_address: 127.0.0.x
rpc_address: 127.0.0.x
```

cassandra-env.sh

```
JMX_PORT=7x99
```

Copy the binaries of Cassandra to run different instances of Cassandra on the sample machine. [Demo purpose]

```
$ ~/Downloads
$ rm -rf apache-cassandra-3.9-bin
$ tar xf apache-cassandra-3.9-bin.tar.gz
$ cp -r apache-cassandra-3.9-bin cassandra11
$ cp -r apache-cassandra-3.9-bin cassandra12
$ cp -r apache-cassandra-3.9-bin cassandra13
$ cp -r apache-cassandra-3.9-bin cassandra14
```

Update the following in conf/cassandra.yaml. Replace x with server serial number.

```
num_tokens: 1
seeds: "127.0.0.1,127.0.0.4"
listen_address: 127.0.0.x
rpc_address: 127.0.0.x
```

In cassandra-env.sh, update JMX_PORT - replace x with serial number of the server to dedicate port for JMX.

```
JMX_PORT=7x99
```

On different terminal launch cassandra server processes.

```
$ ~/Downloads/cassandra11/bin/cassandra -f
$ ~/Downloads/cassandra12/bin/cassandra -f
$ ~/Downloads/cassandra13/bin/cassandra -f
$ ~/Downloads/cassandra21/bin/cassandra -f
$ ~/Downloads/cassandra22/bin/cassandra -f
$ ~/Downloads/cassandra23/bin/cassandra -f
```

View gossip info to verify all nodes that are part of cluster and the listen addresses for each node.

```
~/Downloads/cassandra11/bin/nodetool gossipinfo
```

View the ring. Examine the token ranges.

```
~/Downloads/cassandra11/bin/nodetool ring
```

View the ring summary and load distribution.

```
~/Downloads/cassandra11/bin/nodetool status
```

View state and memory usage of a node. Observe that the output reports some useful information such as the amount of data used, the used and available heap.

```
~/Downloads/cassandra11/bin/nodetool info
```

Some useful linux commands:

Find processes that occupies certain ports

```
$ sudo netstat -tulpn | grep 7199
tcp 0 0 0.0.0.0:7199 0.0.0.0:* LISTEN 11446/java
```

Kill a process using a process ID

```
$ sudo kill -9 <pid like 11446>
```

```
$ sudo ps -ef | grep cassandra

training 11446 8123 1 13:09 pts/1 00:01:17 /usr/java/jdk1.8.0_131/bin/java -
Xloggc:bin/../logs/gc.log -ea -XX:+UseThreadPriorities -XX:ThreadPriorityPolicy=42
```

```
-XX:+HeapDumpOnOutOfMemoryError -Xss256k -XX:StringTableSize=1000003 -
XX:+AlwaysPreTouch -XX:-UseBiasedLocking -XX:+UseTLAB -XX:+ResizeTLAB -XX:+UseNUMA
-XX:+PerfDisableSharedMem -Djava.net.preferIPv4Stack=true -XX:+UseParNewGC -
.... truncated output
```

Decommission a Node

Connect nodetool to the node that you want to decommission

```
$ bin/nodetool -h 127.0.0.4 -p 7499 decommission
```

Verify

```
$ bin/nodetool -h 127.0.0.1 -p 7199 status
```

If you want to the decommissioned node to rejoin the cluster, stop the server, delete the data dir, start the node and run repair.

Open another terminal with cqlsh

```
$ ~/Downloads/cassandra11/bin/cqlsh
cqlsh> CREATE KEYSPACE lab WITH replication = {'class': 'SimpleStrategy',
'replication_factor': 2} ;
cqlsh> use lab;
cqlsh:lab> create table user (id uuid primary key, name text);
cqlsh:lab> insert into user (id, name) values(uuid(), 'test 1'); -- to insert into
a timeuuid, use now()
cqlsh:lab> insert into user (id, name) values(uuid(), 'test 2');
cqlsh:lab> select * from user;
```

Find the token for each record keys

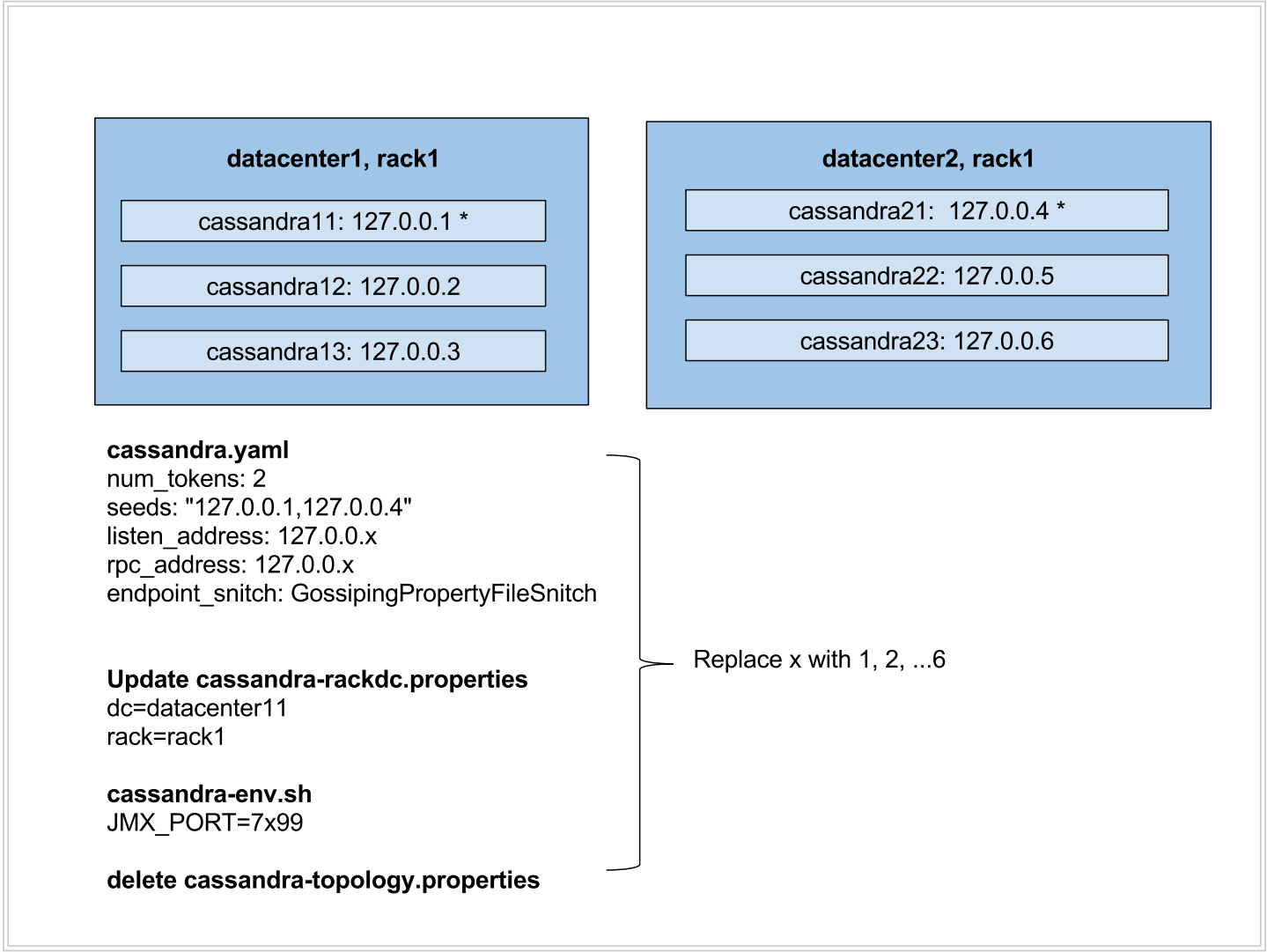
```
cqlsh:lab> select token(id) from user;
```

From the ring tokens, you can find out which node the first replica is stored. You can verify the result by the following command as well.

See target nodes for a given key.

```
$ bin/nodetool getendpoints lab user <user.id e.g. 8cca2490-d74d-4d67-8f82-
e21c34b016b8>
```

Multi Node Multi Data Center



1. Stop all running Cassandra service.

2. Add endpoint_snitch configuration to cassandra.yaml

3. Configure cassandra-rackdc.properties to reflect the nodes belong to multiple data center

Group 1	Nodes
dc=datacenter1 rack = rack1	cassandra11 cassandra12 cassandra13
dc=datacenter2 rack=rack2	cassandra21 cassandra22 cassandra23

4. Delete existing cassandra-topology.properties.

```
rm -rf ~/Downloads/cassandra11/conf/cassandra-topology.properties
rm -rf ~/Downloads/cassandra12/conf/cassandra-topology.properties
rm -rf ~/Downloads/cassandra13/conf/cassandra-topology.properties
rm -rf ~/Downloads/cassandra21/conf/cassandra-topology.properties
rm -rf ~/Downloads/cassandra22/conf/cassandra-topology.properties
rm -rf ~/Downloads/cassandra23/conf/cassandra-topology.properties
```

5. [Optional] Delete existing data from cassandra to start from scratch

```
rm -rf ~/Downloads/cassandra11/data
rm -rf ~/Downloads/cassandra12/data
```

```
rm -rf ~/Downloads/cassandra13/data
rm -rf ~/Downloads/cassandra21/data
rm -rf ~/Downloads/cassandra22/data
rm -rf ~/Downloads/cassandra23/data
```

6. Star the cassandra processes

Create keyspace

```
cqlsh> CREATE KEYSPACE lab WITH REPLICATION = { 'class' :
'NetworkTopologyStrategy', 'datacenter1' : 2, 'datacenter2': 3};
```

References:

- Setup multi node on a single host
- Data Types in Cassandra
- Bulk data loader
- Data with TTL
- Cassandra Bulk Loader - Java application
- Cassandra metrics
- Cassandra Benchmarking Using YCSB tool
- MySQL to Cassandra with Case Studies
- Migration Strategy to Cassandra
- Cassandra Key Performance Metrics
- Cassandra Exception Codes
- Bulk-loading-into-cassandra
- Write Path
- Cassandra on AWS cloud
- JNA for Cassandra, to avoid swap
- Create SSTables from external data sources