Apache Cassandra:
Core Concepts, Skills, and Tools

Understanding Compaction
Exercise Workbook

Joe Chu
Leo Schuman
October, 2014

Demo 1: Open and show SSTable with TTLs and Deletes

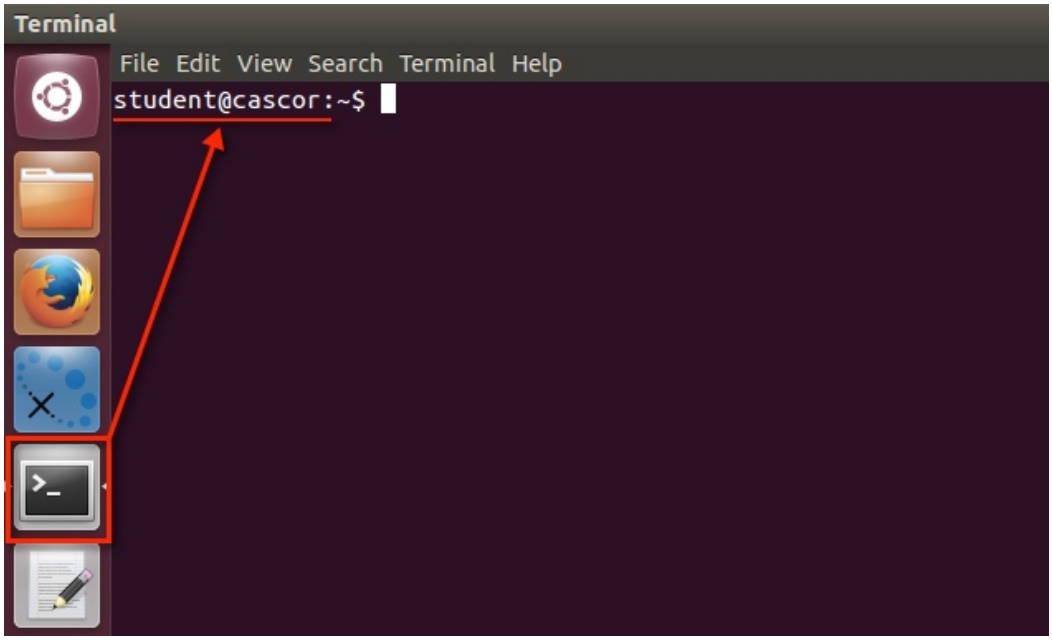
In this demo, your instructor will:

- Examine TTL columns and tombstone markers in a SSTable

Steps

Examine TTL columns and tombstone markers in a SSTable

1. In the virtual machine, open a Terminal window or switch to an existing Terminal running the Linux shell.



2. From the Linux shell, run *ccm flush*.

```
ccm flush
```

3. From the Linux shell, start up *cqlsh*.

```
ccm node1 cqlsh
```

4. In *cqlsh*, set the default keyspace to *musicdb*.

```
USE musicdb;
```

5. From the *musicdb* keyspace, execute a query to view the first 20 distinct performers in the *albums_by_performer*.

```
SELECT DISTINCT performer FROM albums_by_performer LIMIT
20;
```

6. Choose a performer and execute a DELETE operation on the *albums_by_performer* table to delete the entire partition for that performer.

```
DELETE FROM albums_by_performer WHERE performer = 'F5';
```

7. Select another performer and query the first 10 rows from the *albums_by_performer* table with this performer.

```
SELECT * FROM albums_by_performer WHERE performer =
'Rancid' LIMIT 10;
```

8. Run a DELETE operation for an entire row, and a DELETE operation on a column for a row.

```
DELETE FROM albums_by_performer WHERE performer = 'Rancid'
AND year = 2003 AND title = 'Indestructible';

DELETE genre FROM albums_by_performer WHERE performer =
'Rancid'AND year = 1993 AND title = 'Rancid';
```

9. Execute an UPDATE operation on another row in the *albums_by_performer* table, and set the *genre* column to a different value with a TTL of 120 seconds.

```
UPDATE albums_by_performer USING TTL 120 SET genre =
'Rock' WHERE performer = 'Rancid' AND year = 1995 AND
title = 'Time Bomb';
```

10. Exit out of *cqlsh* and return to the command line.

```
EXIT
```

11. On the command line, run *ccm flush*.

```
ccm flush
```

12. Navigate to the directory for *musicdb* keyspace and *albums_by_performer* table.

```
cd ~/node1/data/musicdb/albums_by_performer-[table id]
```

13. In the *albums_by_performer* directory, list the SSTables and find the newest one.

```
ls -l musicdb-albums_by_performer-*Data.db
```

```
student@cascor:~/.ccm/cascor/node1/data/musicdb/albums_by_performer-ac843b806b8f11e4a851d16f287d9d64$ ls -l
musicdb-albums_by_performer-*Data.db
-rw-rw-r-- 1 student student 148924 Nov 13 15:50 musicdb-albums_by_performer-ka-1-Data.db
-rw-rw-r-- 1 student student    160 Nov 13 15:52 musicdb-albums_by_performer-ka-2-Data.db
```

14. Run *sstable2json* against the newest SSTable file and view how the tombstones and TTL row are written.

Modify the command below to use the newest generation number which exists in your current files.

```
sstable2json musicdb-albums_by_performer-ka-[generation-number]-Data.db
```

```
student@cascor:~/.ccm/cascor/node1/data/musicdb/albums_by_performer-ac843b806b8f11e4a851d16f287d9d64$
sstable2json musicdb-albums_by_performer-ka-2-Data.db
[
{"key": "Rancid",
 "cells": [["2003:Indestructible:_","2003:Indestructible:!",1415923113363507,"t",1415923113],
           ["1995:Time Bomb:genre","Rock",1415923134131318,"e",120,1415923254],
           ["1993:Rancid:genre",1415923113,1415923113383539,"d"]]},
{"key": "F5",
 "metadata": {"deletionInfo": {"markedForDeleteAt":1415923106448845,"localDeletionTime":1415923106}},
 "cells": []}
]
```

END OF DEMO

Demo 2: Show size-tiered compaction in use

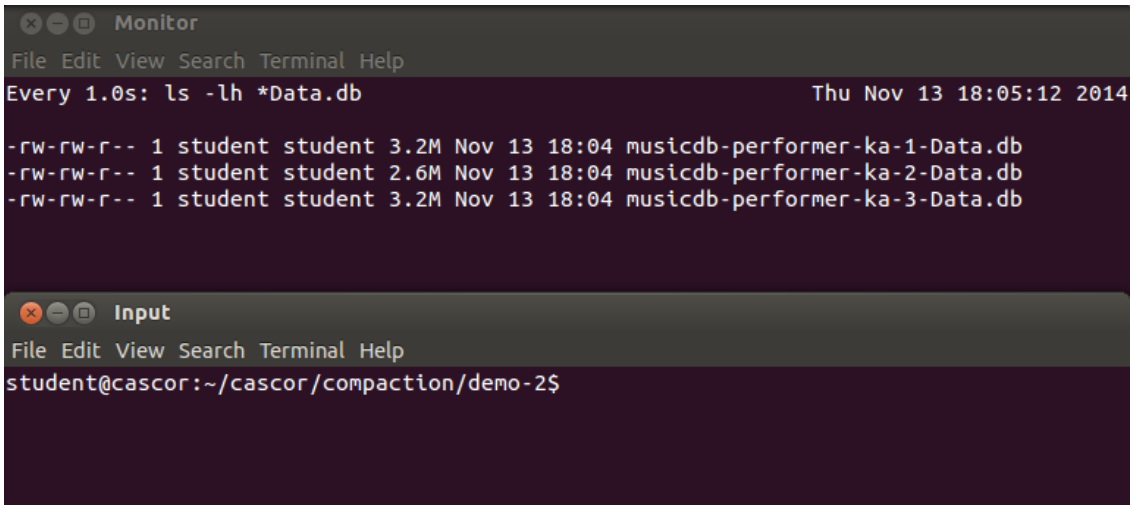
In this exercise, your instructor will:

- Watch how SSTables are created and compacted while inserting data

Steps

Watch how SSTables are created and compacted while inserting data

1. In the virtual machine, open two Terminal windows. One is to be used for input, the other will monitor the SSTables as they are created.



2. In the input window, navigate to the directory for *demo-2*.

```
cd ~/cascor/compaction/demo-2
```

3. Start *cqlsh*, and truncate the *musicdb.performer* table.

```
ccm node1 cqlsh

TRUNCATE musicdb.performer;
```

4. Exit out of *cqlsh* to return to the command line.

```
EXIT
```

5. In the monitor window, navigate to the *performer* table *data* directory in *musicdb*.

```
cd ~/node1/data/musicdb/performer-[table id]
```

6. Start a *watch* to view how the files in the *user* directory change.

```
watch -n 1 -d "ls -lh *Data.db"
```

7. Switch to the input window, and run *cassandra-stress* using the cqlstress.yaml profile to performer 250,000 operations with 25 client threads and no warmup.

```
cassandra-stress user profile=cqlstress.yaml
ops\(insert=1\) no-warmup n=250000 -rate threads=25
```

cassandra-stress should write enough partitions to trigger memtable flushing, creating approximately four SSTables. This should then trigger a compaction, with the resulting SSTable being about four times the size of the compacted SSTables.

You may see a new SSTable being created by compaction that is writing slowly, and may actually see other SSTables written afterwards than finishes much quicker? Why is that?

Running cassandra-stress once should be enough to show a compaction taking place. You can run cassandra-stress as many times as you'd like to demonstrate multiple compactions.

You may also want to open a window running a watch on nodetool compactionstats *to follow the progress of a compaction as well.*

8. From the input window, run *ccm compact* and watch the major compaction.

```
ccm compact
```

END OF DEMO

Exercise 3: Configure and run other compaction strategies

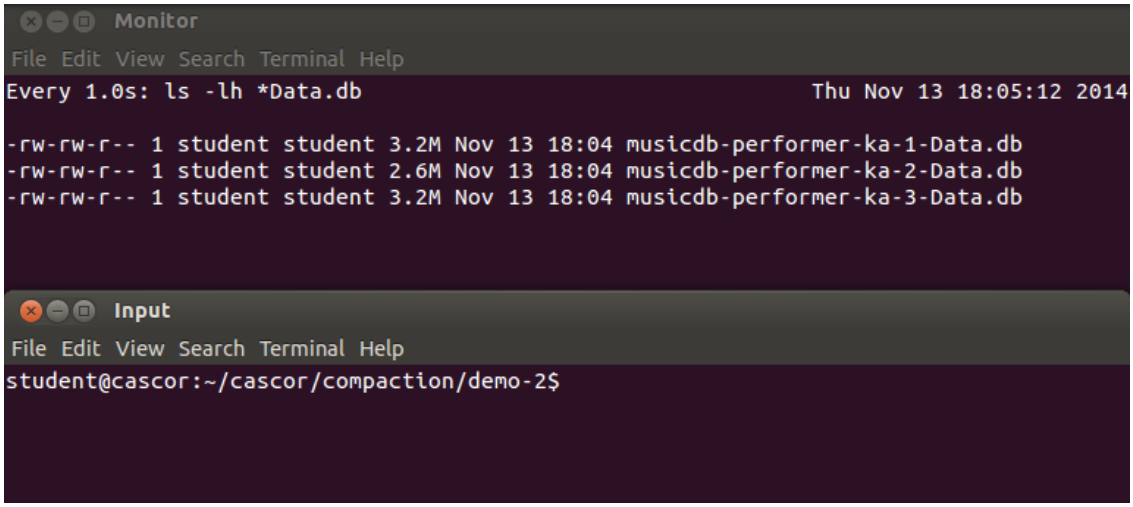
In this exercise, you will:

- Configure a table to use the Leveled Compaction Strategy
- Create a new table to use the Date-tiered Compaction Strategy

Steps

Configure a table to use the Leveled Compaction Strategy

1. In the virtual machine, open two Terminal windows. One is to be used for input, the other will monitor the SSTables as they are created.



2. In the monitor window, navigate to the data directory for the *musicdb.performer* table on node1.

```
cd ~/node1/data/musicdb/performer-[table id]
```

3. In the performer directory, run a watch to monitor the SSTable files.

```
watch -n 1 -d "ls -lh *-Data.db"
```

4. Switch to the input window and navigate to the *exercise-3* directory for the compaction module.

```
cd ~/cascor/compaction/exercise-3
```

5. In the *exercise-3* directory, start up *cqlsh*.

```
ccm node1 cqlsh
```

6. Change the default keyspace to *musicdb*.

```
USE musicdb;
```

7. From the *musicdb* keyspace, alter the table *performer* to use the Leveled Compaction Strategy and set the SSTable size to 1MB.

```
ALTER TABLE performer WITH compaction = {'class':
'LeveledCompactionStrategy', 'sstable_size_in_mb': 1};
```

If you are monitoring the window showing the SSTables, you may see that the SSTables are compacted into new SSTables using the Leveled Compaction Strategy.

8. Use the EXIT or QUIT command to close *cqlsh* and return to the command line.

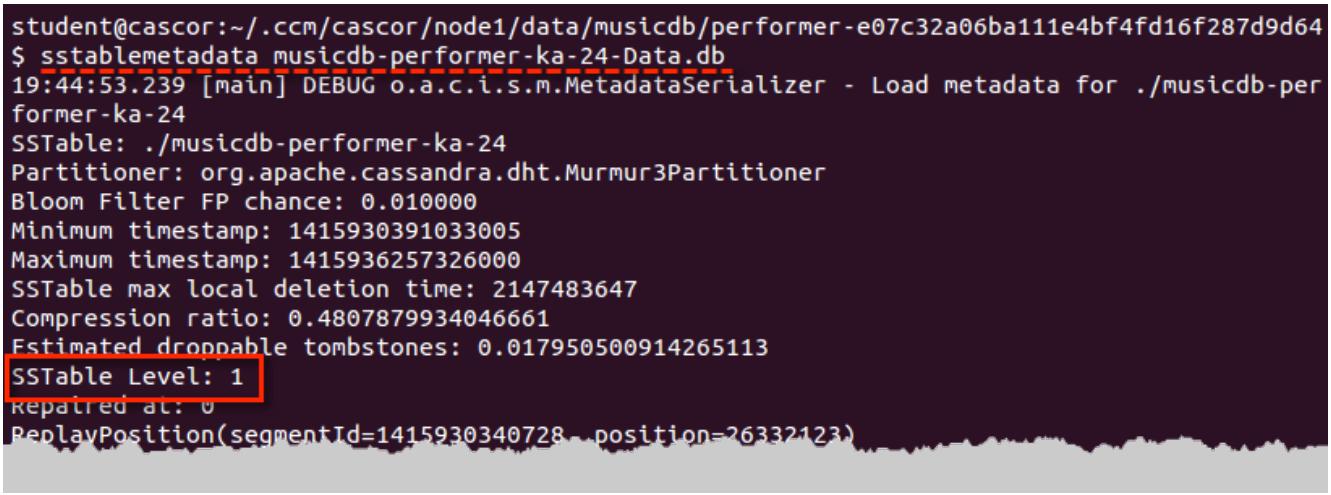
```
EXIT
```

9. In the input window, run *cassandra-stress* with the *cqlstress.yaml* profile to run 20,000 more insert operations in the *performer* table, using 1 client thread and no warmup.

```
cassandra-stress user profile=cqlstress.yaml
ops\(insert=1\) no-warmup n=20000 -rate threads=1
```

10. From the monitor window, interrupt the *watch* program with ctrl-c, and then run *sstablemetadata* on any one of the SSTable files to find its level.

```
sstablemetadata musicdb-performer-ka-[generation]-Data.db
```



11. Switch back to the input window, and run the script *sstablelevel.sh* to display the levels for all of the SSTables in the *performer* table on node1.

```
./sstablelevel.sh node1 musicdb performer
```

This script uses sstablemetadata to find the SSTable level, and just makes it a bit easier to visualize the levels for each of the SSTables in the performer table.

Configure a table to use the Date-Tiered Compaction Strategy

12. In the input window, run *cqlsh* to drop the *track_ratings_by_user* table in the *musicdb keyspace*.

```
ccm node1 cqlsh -x "USE musicdb; DROP TABLE IF EXISTS
track_ratings_by_user"
```

13. Open the file timeseriesdata.yaml with a text editor and review the profile. For the table definition, make note of the compaction strategy and sub-compaction properties.

14. In the input window, run *cassandra-stress* using the *timeseriesdata.yaml* profile to perform inserts for 3 minutes using 1 client thread with no warmup. Continue with the next step while *cassandra-stress* is running.

```
cassandra-stress user profile=timeseriesdata.yaml
ops\(insert=1\) no-warmup duration=3m -rate threads=1
```

As SSTables are created, if there are multiple SSTables created within the first 60 seconds, they may be compacted since they are in the same time window.

15. In the monitor window, navigate to the data directory for the *musicdb* keyspace and *track_ratings_by_user* table on *node1*.

```
cd ~/node1/data/musicdb/track_ratings_by_user-[table id]
```

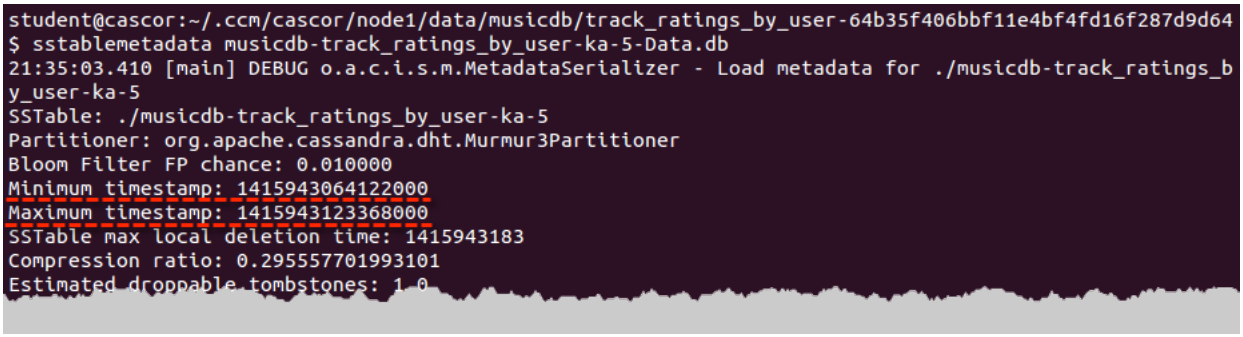
16. In the *track_ratings_by_user* data directory, run *watch* to monitor the SSTables being created.

```
watch -n 1 -d ls -lh "*Data.db"
```

17. After *cassandra-stress* has completed and all pending compactions have completed, switch to the monitor window and press ctrl-c to exit out of the *watch* program.

18. Use *sstablemetadata* to take a look at the minimum and maximum timestamp in one of the SSTables.

```
sstablemetadata musicdb-track_ratings_by_user-ka-5-Data.db
```



The DateTieredCompactionStrategy makes use of this metadata when placing SSTables in time windows to select candidates for compaction.

END OF EXERCISE