CarbonData Demo

Prepare environment

1. Compile carbondata assembly JAR from https://github.com/apache/carbondata
2. Start spark cluster. Following example assumes spark and hdfs is started in 3 hosts: carbon1, carbon2, carbon3. Carbon1 is running spark master and hdfs namenode
3. Start CarbonThriftServer in carbon1:

```
./bin/spark-submit --master spark://carbon1:7077 --class org.apache.carbondata.spark.thriftserver.CarbonThriftServer ./jars/apache-carbondata-1.5.0-SNAPSHOT-bin-spark2.3.1-hadoop2.7.2.jar hdfs://carbon1:9000/carbon.store
```

4. Start beeline to connect to CarbonThriftServer:

```
. bin/beeline -u jdbc:hive2://carbon1:10000
```

Prepare data

1. Down all script from https://github.com/jackylk/tpch

2. Generate TCPH data, scale is 10

```
./generate_data.sh
```

3. Prepare a lineitem CSV file with 10000 lines (used for batch load and streaming ingest)

```
head -n 10000 lineitem.tbl > lineitem_1w.tbl
```

4. Create TPCH tables and load data

```
./ create_and_load.sh
```

Following database and table will be created

```
0: jdbc:hive2://carbon1:10000> show databases;
+--------------------------+--+
|      databaseName        |
+--------------------------+--+
| default                  |
| tpchcarbon_base          |
| tpchhive                 |
| tpchparquet_partition    |
+--------------------------+--+
```

```
0: jdbc:hive2://carbon1:10000> show tables;
+-------------------+------------+--------------+--+
|     database      | tableName  | isTemporary  |
+-------------------+------------+--------------+--+
| tpchcarbon_base   | customer   | false        |
| tpchcarbon_base   | lineitem   | false        |
| tpchcarbon_base   | nation     | false        |
| tpchcarbon_base   | orders     | false        |
| tpchcarbon_base   | part       | false        |
| tpchcarbon_base   | partsupp   | false        |
| tpchcarbon_base   | region     | false        |
| tpchcarbon_base   | supplier   | false        |
+-------------------+------------+--------------+--+
```

Create CarbonData table

Create table stored ad CarbonData and set “L_SHIPDATE, L_PARTKEY” as index columns.

```
CREATE TABLE LINEITEM (
 L_ORDERKEY BIGINT,
 L_PARTKEY BIGINT,
 L_SUPPKEY BIGINT,
 L_LINENUMBER INTEGER,
 L_QUANTITY DECIMAL,
 L_EXTENDEDPRICE DECIMAL,
 L_DISCOUNT DECIMAL,
 L_TAX DECIMAL,
 L_RETURNFLAG CHAR(1),
 L_LINESTATUS CHAR(1),
 L_SHIPDATE DATE,
 L_COMMITDATE DATE,
 L_RECEIPTDATE DATE,
 L_SHIPINSTRUCT CHAR(25),
```

```
 L_SHIPMODE CHAR(10),
 L_COMMENT VARCHAR(44)
)
STORED AS carbondata
TBLPROPERTIES ('SORT_COLUMNS'='L_SHIPDATE,L_PARTKEY',
'TABLE_BLOCKSIZE'='64');
```

Load data, Update and Delete

Following steps will demonstrate data management feature of carbon including:

1. batch load and show segment
2. delete segment
3. data update/delete
4. query data of specific segment
5. data compaction
6. clean files to release space

Data Management

Demo steps:

1. Load lineitem_1w.tbl to table lineitem

```
LOAD DATA INPATH 'hdfs://carbon1:9000/tpch-data/lineitem_1w' INTO TABLE lineitem OPTIONS ('header'='false','delimiter'='|')
```

2. Show segments

```
show segments for table lineitem
```

It shows

```
0: jdbc:hive2://carbon1:10000> show segments for table lineitem;
+--------------------+----------+--------------------------+--------------------------+-----------+-------------+-----------+------------+--+
| SegmentSequenceId  |  Status  |     Load Start Time      |      Load End Time       | Merged To | File Format | Data Size | Index Size |
+--------------------+----------+--------------------------+--------------------------+-----------+-------------+-----------+------------+--+
| 3                  | Success  | 2018-09-24 23:48:11.789  | 2018-09-24 23:48:12.466  | NA        | COLUMNAR_V3 | 430.94KB  | 2.42KB     |
| 1                  | Success  | 2018-09-24 21:57:51.74   | 2018-09-24 22:01:25.968  | NA        | COLUMNAR_V3 | 2.02GB    | 26.40KB    |
+--------------------+----------+--------------------------+--------------------------+-----------+-------------+-----------+------------+--+
2 rows selected (0.046 seconds)
```

3. Suppose admin told there is some mistake in the data we just loaded, so we use update/delete feature to correct the data

```
delete from lineitem where l_orderkey = 4678;

update lineitem set (l_partkey)=(0) where l_orderkey=5601;
```

4. Or, we just delete the whole segment

```
delete from table lineitem where segment.id in (3)
```

5. Continue to load 3 segments, and trigger a compaction

```
alter table lineitem compact 'major'
```

6. Set auto compaction and continue to load 10 segments

```
set carbon.enable.auto.load.merge=true
set carbon.compaction.level.threshold=2,2
```

Show segment will show:

```
0: jdbc:hive2://carbon1:10000> show segments for table lineitem;
+--------------------+-----------+--------------------------+--------------------------+------------+--------------+------------+-------------+--+
| SegmentSequenceId  |   Status  |     Load Start Time      |      Load End Time       | Merged To  | File Format  | Data Size  | Index Size  |
+--------------------+-----------+--------------------------+--------------------------+------------+--------------+------------+-------------+--+
| 11                 | Success   | 2018-09-25 00:12:38.167  | 2018-09-25 00:12:38.789  | NA         | COLUMNAR_V3  | 430.94KB   | 2.43KB      |
| 10                 | Success   | 2018-09-25 00:12:36.041  | 2018-09-25 00:12:36.634  | NA         | COLUMNAR_V3  | 430.94KB   | 2.43KB      |
| 9                  | Compacted | 2018-09-25 00:11:54.541  | 2018-09-25 00:11:55.143  | 6.1        | COLUMNAR_V3  | 430.94KB   | 2.38KB      |
| 8                  | Compacted | 2018-09-25 00:11:44.913  | 2018-09-25 00:11:45.554  | 6.1        | COLUMNAR_V3  | 430.94KB   | 2.42KB      |
| 7                  | Compacted | 2018-09-25 00:11:38.273  | 2018-09-25 00:11:38.982  | 6.1        | COLUMNAR_V3  | 430.94KB   | 2.42KB      |
| 6.1                | Success   | 2018-09-25 00:11:54.541  | 2018-09-25 00:11:55.878  | NA         | COLUMNAR_V3  | 668.41KB   | 2.43KB      |
| 6                  | Compacted | 2018-09-25 00:11:28.183  | 2018-09-25 00:11:28.766  | 6.1        | COLUMNAR_V3  | 430.94KB   | 2.42KB      |
| 5                  | Compacted | 2018-09-25 00:09:33.949  | 2018-09-25 00:09:34.594  | 1.1        | COLUMNAR_V3  | 430.94KB   | 2.38KB      |
| 4                  | Compacted | 2018-09-25 00:09:31.689  | 2018-09-25 00:09:32.28   | 1.1        | COLUMNAR_V3  | 430.94KB   | 2.42KB      |
| 3                  | Compacted | 2018-09-25 00:09:15.968  | 2018-09-25 00:09:16.665  | 1.1        | COLUMNAR_V3  | 430.94KB   | 2.42KB      |
| 1.1                | Success   | 2018-09-25 00:09:33.949  | 2018-09-25 00:10:33.208  | NA         | COLUMNAR_V3  | 2.02GB     | 26.50KB     |
| 1                  | Compacted | 2018-09-25 00:05:30.803  | 2018-09-25 00:07:45.809  | 1.1        | COLUMNAR_V3  | 2.02GB     | 26.40KB     |
+--------------------+-----------+--------------------------+--------------------------+------------+--------------+------------+-------------+--+
```

7. Query specified segment

```
set carbon.input.segments.tpchcarbon_base.lineitem=11;
```

```
0: jdbc:hive2://carbon1:10000> select count(*) from lineitem;
+-----------+--+
| count(1)  |
+-----------+--+
| 10000     |
+-----------+--+
1 row selected (0.074 seconds)
```

Query performance and DataMap usage

1. Filter query on first sort column

```
select count(*) from lineitem where l_shipdate>'1992-05-03' and
l_shipdate<'1992-06-03' and l_returnflag='R';
```

Carbon:

```
0: jdbc:hive2://carbon1:10000> select count(*) from lineitem where l_shipdate>'1992-05-03' and l_shipdate<'1992-06-03' and l_returnflag='R';
+-----------+--+
| count(1)  |
+-----------+--+
| 372328    |
+-----------+--+
1 row selected (0.379 seconds)
```

Parquet:

```
0: jdbc:hive2://carbon1:10000> select count(*) from lineitem where l_shipdate>'1992-05-03' and l_shipdate<'1992-06-03' and l_returnflag='R';
+-----------+--+
| count(1)  |
+-----------+--+
| 372328    |
+-----------+--+
1 row selected (1.023 seconds)
```

2. Filter query on 4th sort column (forth column)

```
select count(*) from lineitem where l_receiptdate='1992-05-03';
```

Carbon:

```
0: jdbc:hive2://carbon1:10000> select count(*) from lineitem where l_receiptdate='1992-05-03';
+-----------+--+
| count(1)  |
+-----------+--+
| 21794     |
+-----------+--+
1 row selected (0.314 seconds)
```

Parquet:

```
0: jdbc:hive2://carbon1:10000> select count(*) from lineitem where l_receiptdate='1992-05-03';
+-----------+--+
| count(1)  |
+-----------+--+
| 21794     |
+-----------+--+
1 row selected (6.075 seconds)
```

If you explain the query, it will show most of the blocks are skipped

```
| == CarbonData Profiler ==
Table Scan on lineitem
 - total: 13 blocks, 39 blocklets
 - filter: (l_receiptdate <> null and l_receiptdate = 704851200000000)
 - pruned by Main DataMap
    - skipped: 10 blocks, 30 blocklets
```

3. Filter query on non-sort column

```
select * from lineitem where l_partkey=123456;
```

Carbon:

```
25 rows selected (2.484 seconds)
```

4. Create bloom filter datamap to improve filter query on l_partkey

```
create datamap bloom on table lineitem using 'bloomfilter'
dmproperties ('index_columns'='l_partkey');
```

Query again:

```
25 rows selected (1.382 seconds)
```

5. Aggregate query performance (TPC-H Q1)

```
select l_returnflag, l_linestatus, sum(l_quantity) as sum_qty,
sum(l_extendedprice) as sum_base_price, sum(l_extendedprice*(1-
l_discount)) as sum_disc_price, sum(l_extendedprice*(1-
l_discount)*(1+l_tax)) as sum_charge, avg(l_quantity) as
avg_qty, avg(l_extendedprice) as avg_price, avg(l_discount) as
avg_disc, count(*) as count_order from lineitem where l_shipdate
<= date('1998-09-02') group by l_returnflag, l_linestatus order
by l_returnflag, l_linestatus;
```

Carbon result:

```
1 row selected (4.421 seconds)
```

Create datamap by:

```
create datamap agg on table lineitem using "preaggregate" as
select l_returnflag, l_linestatus, sum(l_quantity) as sum_qty,
sum(l_extendedprice) as sum_base_price, sum(l_extendedprice*(1-
l_discount)) as sum_disc_price, sum(l_extendedprice*(1-
l_discount)*(1+l_tax)) as sum_charge, avg(l_quantity) as
avg_qty, avg(l_extendedprice) as avg_price,
avg(l_discount) as avg_disc, count(*) as count_order from
lineitem group by l_shipdate, l_returnflag, l_linestatus;
```

Query again:

```
1 row selected (1.054 seconds)
```

Demo result on 30GB TPCH data:

	CarbonData	Spark-Parquet
Transactional Write	Support	Not Support
Update/Delete/Compaction	Support	Not Support
Streaming	Support	Not Support
Filter query including first sort column: select count(*) from lineitem where l_shipdate>'1992-05-03' and l_shipdate<'1992-06-03' and l_returnflag='R'	1.7s	51.2s
Filter query on 4th sort column: select count(*) from lineitem where l_receiptdate='1998-05-03'	0.9	55.9
Filter query on non sort column: select * from lineitem where l_partkey=123456;	3.4	14.3
Full scan aggregation: select count(l_suppkey) from lineitem;	2.7	7.8
Full scan aggregation: TPCH Q1	11.7	11.9
TPCH Q1 with preaggregate	1.3	

Stream ingest and query

1. Start kafka server

```
nohup bin/zookeeper-server-start.sh config/zookeeper.properties &
bin/kafka-server-start.sh config/server.properties
```

2. Create topic “test”

```
bin/kafka-topics.sh --create --zookeeper localhost:2181 --
replication-factor 1 --partitions 1 --topic test
```

3. Create stream source table

```
CREATE TABLE LINEITEM_kafka (
 L_ORDERKEY INT ,
 L_PARTKEY INT ,
 L_SUPPKEY string,
 L_LINENUMBER int,
 L_QUANTITY double,
 L_EXTENDEDPRICE double,
 L_DISCOUNT double,
 L_TAX double,
 L_RETURNFLAG string,
 L_LINESTATUS string,
 L_SHIPDATE date,
 L_COMMITDATE date,
 L_RECEIPTDATE date,
 L_SHIPINSTRUCT string,
 L_SHIPMODE string,
 L_COMMENT string)
STORED AS carbondata
TBLPROPERTIES (
'streaming'='source',
'format'='kafka',
'kafka.bootstrap.servers'='localhost:9092',
'subscribe'='test',
'delimiter'='|');
```

4. Set tpchcarbon_base.lineitem table as the stream sink table

```
alter table lineitem set tblproperties ('streaming'='true');
```

5. Create stream ingest job

```
create stream job1 on table lineitem stmproperties
('trigger'='ProcessingTime', 'interval'='3 seconds') as select *
from lineitem_kafka;
```

```
+--------------+--------------------------------------+----------+--+
| Stream Name  |                JobId                 |  Status  |
+--------------+--------------------------------------+----------+--+
| job1         | 891a0778-6833-484d-bdb9-a30589975fe1 | RUNNING  |
+--------------+--------------------------------------+----------+--+
1 row selected (1.89 seconds)
```

You can show stream job by SHOW STREAMS command

```
0: jdbc:hive2://carbon15:10000> show streams on table lineitem;
+-------------+--------------------------------------+----------+------------------------+---------------------------+----------------------
--------+---------------------------------+--+
| Stream Name |                JobId                 |  Status  |         Source         |           Sink            |          Start Time
        |          Time Elapse            |
+-------------+--------------------------------------+----------+------------------------+---------------------------+----------------------
--------+---------------------------------+--+
| job1        | 891a0778-6833-484d-bdb9-a30589975fe1  | RUNNING  | default.lineitem_kafka  | tpchcarbon_base.lineitem  | Tue Sep 25 01:11:41 C
ST 2018  | 0 days, 0 hours, 0 min, 39 sec  |
+-------------+--------------------------------------+----------+------------------------+---------------------------+----------------------
--------+---------------------------------+--+
```

6. Start feeding kafka by script

```
./ingest_kafka.sh
```

ingest_kafka.sh:

```
for i in `seq 1 200`
do
cat /opt/dbgen/1w/lineitem_1w.tbl| /dev/kafka_2.11-
1.1.0/bin/kafka-console-producer.sh --broker-list localhost:9092 -
-topic test
sleep 3
done
```

It will ingest 30000 records every 3 seconds

7. Now you can query the lineitem table, its count will keep changing

8. Check the streaming segment

```
0: jdbc:hive2://carbon15:10000> show segments for table lineitem;
+--------------------+------------+--------------------------+--------------------------+------------+--------------+------------+------------
+--+
| SegmentSequenceId  |   Status   |     Load Start Time      |      Load End Time       | Merged To  | File Format  | Data Size  | Index Size
|
+--------------------+------------+--------------------------+--------------------------+------------+--------------+------------+------------
+--+
| 27                 | Streaming  | 2018-09-17 12:18:42.762  | NA                       | NA         | ROW_V1       | 8.70MB     | 313.0B
|
| 0                  | Success    | 2018-09-07 00:08:44.256  | 2018-09-07 00:13:15.149  | NA         | COLUMNAR_V3  | 1.99GB     | 21.03KB
|
+--------------------+------------+--------------------------+--------------------------+------------+--------------+------------+------------
+--+
```

9. Stop stream job

```
DROP STREAM job1;
```

10. Convert streaming files to columnar files:

```
alter table lineitem compact 'close_streaming';
```

Check the segment again:

```
0: jdbc:hive2://carbon15:10000> show segments for table lineitem;
+--------------------+------------+--------------------------+--------------------------+------------+--------------+------------+------------
+--+
| SegmentSequenceId  |   Status   |     Load Start Time      |      Load End Time       | Merged To  | File Format  | Data Size  | Index Size
|
+--------------------+------------+--------------------------+--------------------------+------------+--------------+------------+------------
+--+
| 28                 | Success    | 2018-09-25 01:20:58.238  | 2018-09-25 01:21:01.045  | NA         | COLUMNAR_V3  | 1.28MB     | 2.34KB
|
| 27                 | Compacted  | 2018-09-17 12:18:42.762  | 2018-09-25 01:20:58.228  | 28         | ROW_V1       | 8.70MB     | 313.0B
|
| 0                  | Success    | 2018-09-07 00:08:44.256  | 2018-09-07 00:13:15.149  | NA         | COLUMNAR_V3  | 1.99GB     | 21.03KB
|
+--------------------+------------+--------------------------+--------------------------+------------+--------------+------------+------------
+--+
```

It will also set the streaming table property in sink table to false. So if you want to start the streaming ingest job again, you need to set it to true first

```
alter table lineitem set tblproperties ('streaming'='true');
```