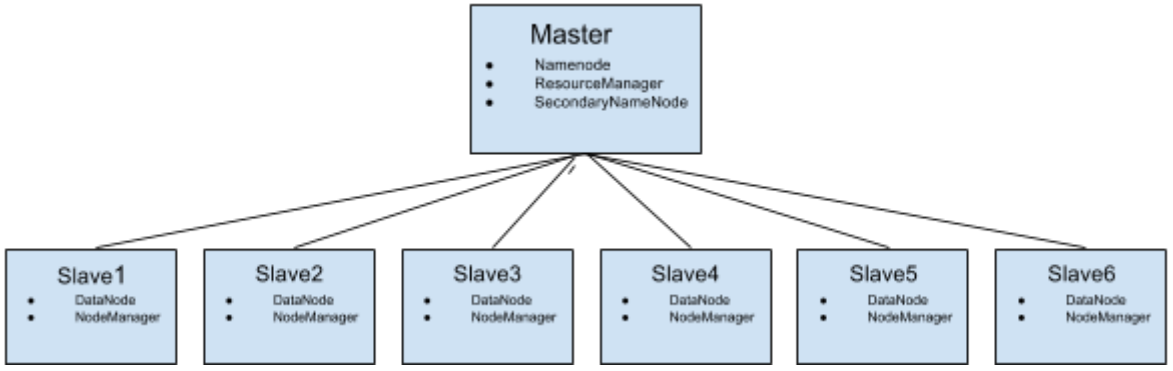
Background

This document explains the TPCH report of CarbonData(1.2 version) and Parquet on Spark execution engine.

Hardware

CPU : Intel® Core™ i7-6700 Quad-Core
Memory : 64 GB DDR4 RAM
HardDisk : 2 x 2 TB SATA 6 Gb/s 7200 rpm HDD



Configurations

Carbon Properties

```
carbon.number.of.cores.while.loading=5
carbon.lock.type=HDFSLOCK
enable.data.loading.statistics=true
enable.unsafe.sort=true
offheap.sort.chunk.size.inmb=128
sort.inmemory.size.inmb=3072
carbon.enable.vector.reader=true
enable.unsafe.in.query.processing=true
enable.query.statistics=true
carbon.blockletgroup.size.in.mb=128
enable.unsafe.columnpage=true
carbon.unsafe.working.memory.in.mb=3072
```

Spark Conf

```
spark.executor.extraJavaOptions = -XX:CompressedClassSpaceSize=512m
-Dcarbon.properties.filepath=carbon.properties
spark.driver.extraJavaOptions = -Djetty.version=x.y.z
-Dcarbon.properties.filepath=/opt/spark-2.1.0-bin-hadoop2.7/conf/carbon.properties
spark.random.port.min = 23000
spark.random.port.max = 23999
spark.shuffle.manager = SORT
#spark.ui.port = 23060
spark.kryoserializer.buffer.max = 128m
spark.am.memory = 2g
spark.master=yarn-client
spark.yarn.dist.files=/opt/spark-2.1.0-bin-hadoop2.7/conf/carbon.properties
spark.yarn.executor.memoryOverhead=10240
spark.network.timeout=6000
spark.sql.warehouse.dir=hdfs://master:9000/spark-warehouse
spark.io.compression.codec=snappy
hive.exec.dynamic.partition.mode=nonstrict
hive.exec.dynamic.partition=true
hive.exec.max.dynamic.partitions=5000
spark.shuffle.reduceLocality.enabled=false
```

Scripts and data

TPCH Scripts which are used in this test is Shared in the following link.

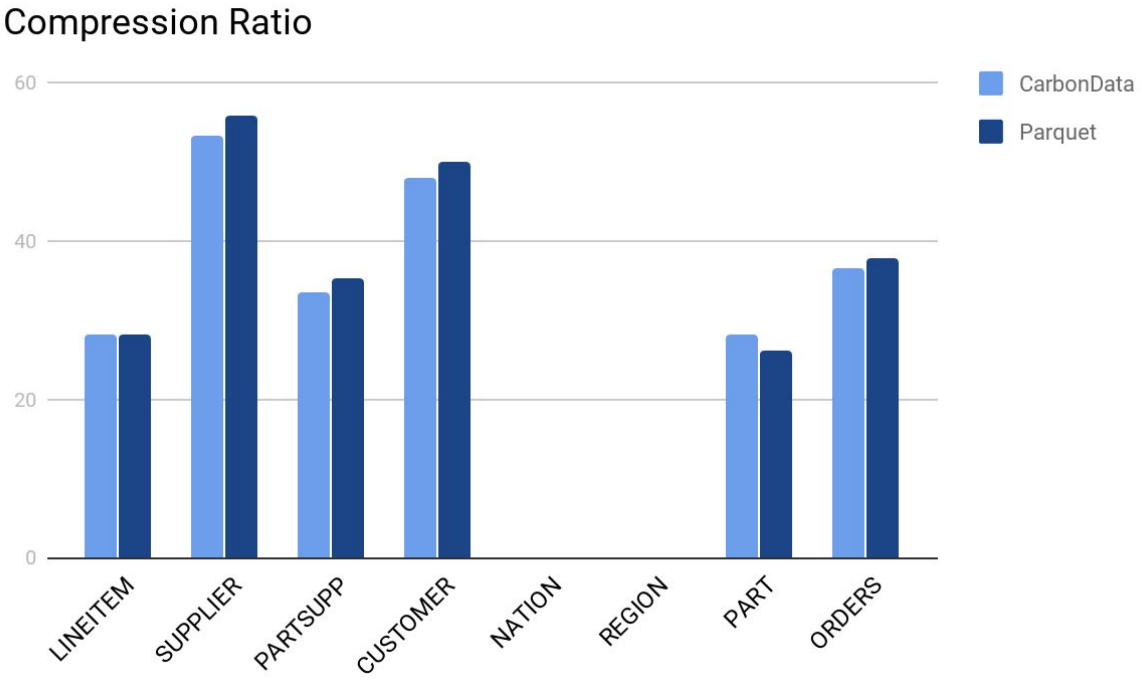
https://drive.google.com/open?id=0B4TWTVbFSTnqSXZibIFkZEJ0LXM

Data Size : 200 GB (Generated using https://github.com/electrum/tpch-dbgen)

Compression Ratio

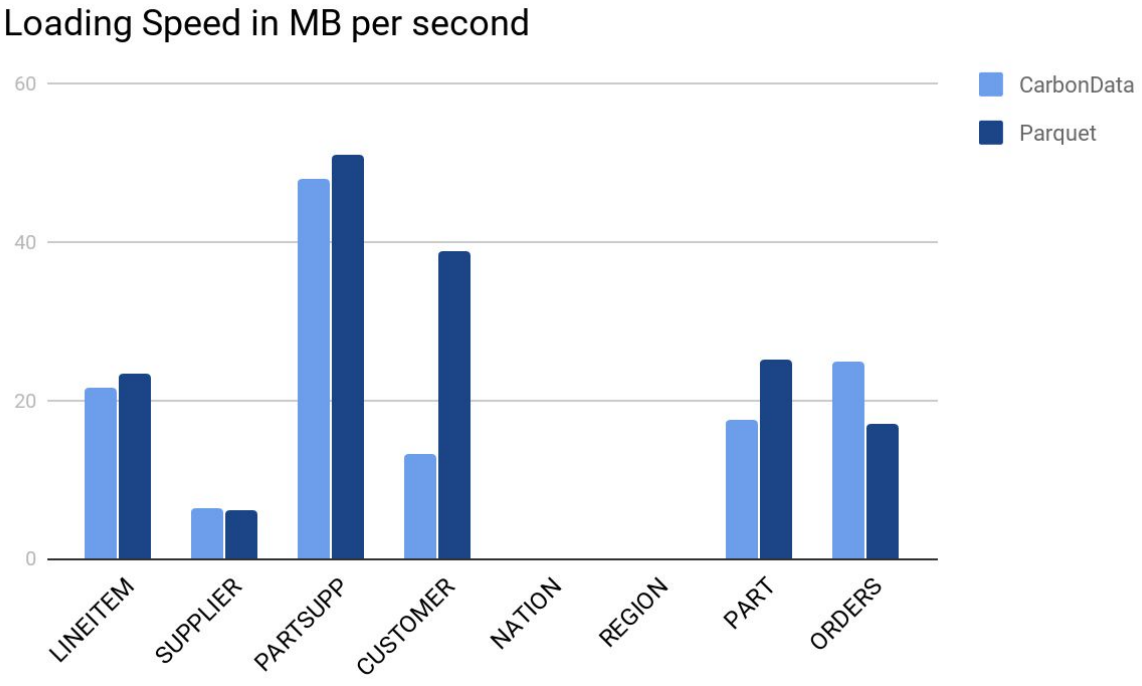
The following chart depicts the compression ratio between Carbon and Parquet.

Note : Lower bar is better performance



Loading Performance

The following chart depicts the loading performance between carbon and parquet. It is shown in MB per second per each node.
Note : Higher bar is better performance

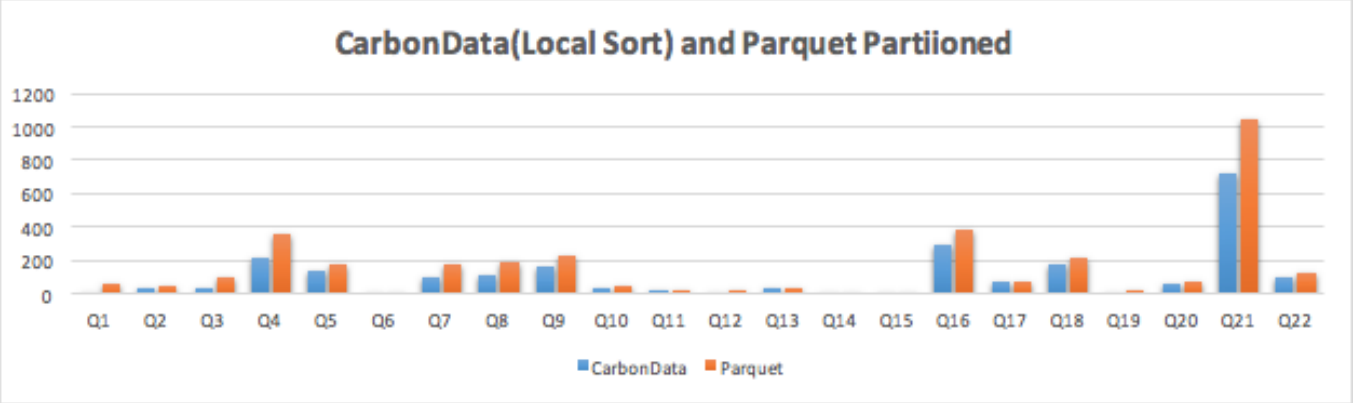


Query Performance

The following chart depicts the performance of Carbon and Parquet. To have a fair comparison we have done test in 2 ways.

1. Carbon loaded with node local sort (default) and Parquet is partitioned
2. Carbon loaded with no sort option and Parquet loaded directly.

Note : Lower bar is better performance



Queries	CarbonData(Local Sort)-seconds	Parquet(Partitioned)-seconds
Q1	13.1	55
Q2	29.3	53
Q3	39.3	94
Q4	218	363
Q5	134	178
Q6	3.8	4.1
Q7	102	178
Q8	111	185
Q9	167	226
Q10	37.1	53.7
Q11	21	26

Q12	13	24.2
Q13	33	36.3
Q14	5.2	13.2
Q15	7.2	9.2
Q16	291	389
Q17	72	69
Q18	180.1	222.1
Q19	5.7	18.2
Q20	58	71
Q21	720.1	1048.8
Q22	95.6	131.5