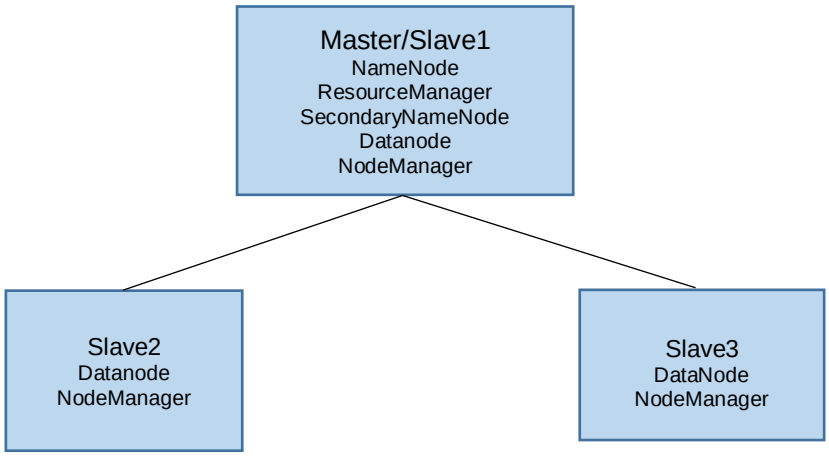
The TPC-DS Benchmark of CarbonData 2.1.0 and Parquet 1.10.1 on Spark 2.4.5

TPC-DS is the industry standard benchmark for measuring the performance of decision support solutions. We used this TPCDS to compare CarbonData (2.1.0 version) and Parquet on Spark 2.4.5 execution engine with Data of 500 GB Scala Factor.

Hardware

CPU: Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz - 48 CPU
Memory: 378 GB DDR4 RAM
Hard Disk: 12 x 4 TB SATA 7200 RPM HDD
Total Nodes: 3



Configurations

CarbonData Properties

```
# Number of cores used while loading per executor, Used 20 cores while loading the data.
carbon.number.of.cores.while.loading = 20

#carbon sort size
carbon.sort.size = 500000

#compressor for writing/reading CarbonData file
carbon.column.compressor = snappy

#System property to enable or disable local dictionary generation
carbon.local.dictionary.enable = true

#when enabled complete row filters will be handled by carbon in case of vector. If it is disabled then only page level pruning
will be done by carbon and row level filtering will be done by spark for vector. There is no change in flow for non-vector
based queries.
carbon.push.rowfilters.for.vector = false

#this property defines how the tasks are split/combined and launch spark tasks during query
carbon.task.distribution = blocklet

#Unsafe Memory
carbon.unsafe.working.memory.in.mb=2048

#property to be specified for caching level (Block/Blocket)
'CACHE_LEVEL'='BLOCKLET'

#table blocklet size in MB
'table_blocklet_size'='64'

#table block size in MB
'table_blocksize'='64'
```

Spark Properties

Data Loading Properties
spark.master=yarn-client
spark.executor.instances=3
spark.executor.cores=20
spark.executor.memory =180G
spark.yarn.am.memory = 5G
spark.yarn.driver.memoryOverhead =10G
spark.yarn.executor.memoryOverhead = 5G
spark.driver.memory = 50G

Query Execution Properties
spark.master=yarn-client
spark.executor.instances = 20
spark.executor.cores=4
spark.executor.memory = 30G
spark.yarn.am.memory = 5G
spark.yarn.driver.memoryOverhead = 3G
spark.yarn.executor.memoryOverhead = 4G
spark.driver.memory = 40G

Common Properties for Data Loading and Query Execution
spark.scheduler.mode = FIFO
spark.shuffle.statistics.verbose = TRUE
spark.sql.authorization.enabled = TRUE
spark.sql.crossJoin.enabled = TRUE
spark.thriftserver.proxy.enabled = TRUE
spark.sql.innerjoin.addNullFilter = TRUE
spark.acls.enable = FALSE
spark.sql.shuffle.partitions = 200
spark.sql.parquet.filterPushdown=true
spark.hadoop.datanucleus.schema.autoCreateAll=true
spark.sql.hive.convertMetastoreCtas=false
spark.sql.optimizer.bloomFilterPruning.enabled=false
spark.io.compression.codec=snappy

Compression Ratio

The following depicts the compression ratio between Carbon and Parquet.

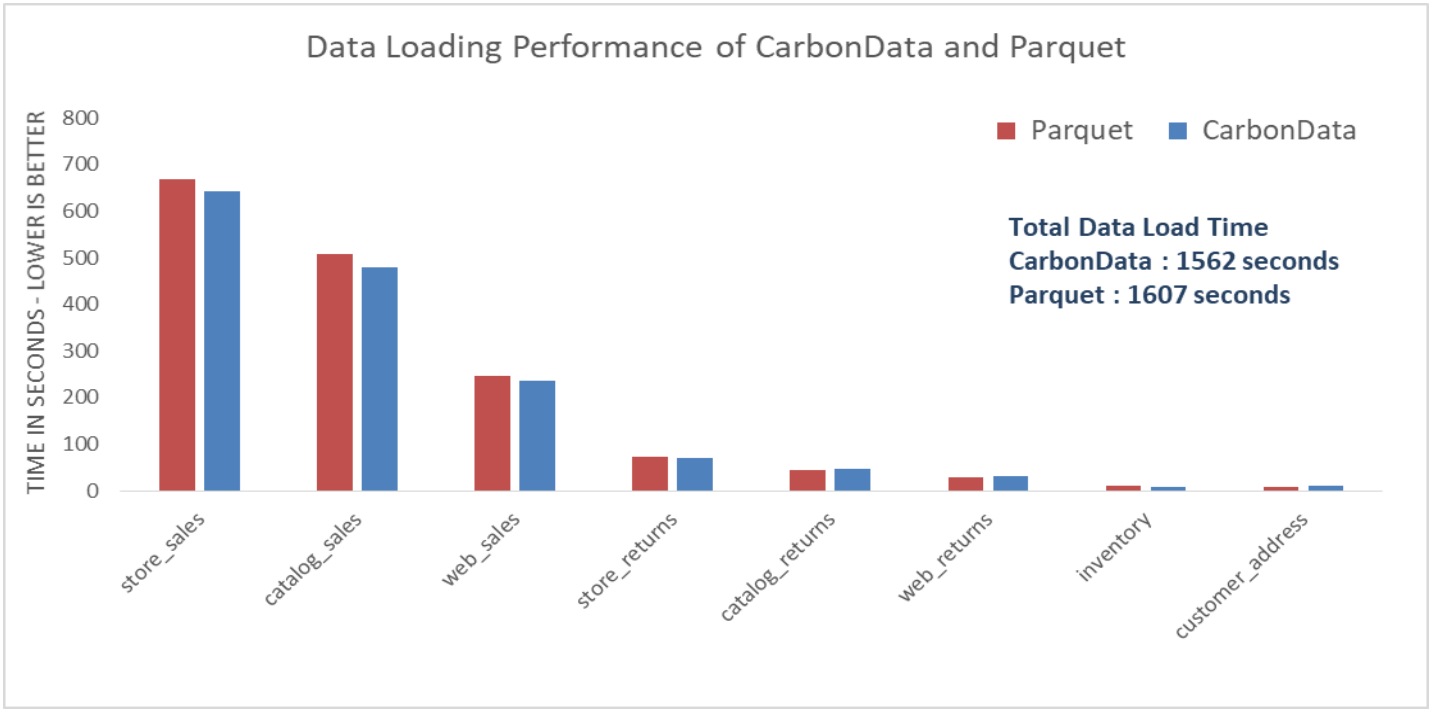
Formula
Compression Ratio = ((O - S)/S) * 100
O = Raw Data Size
S = Store size (Parquet or Carbon)

TPCDS Scala Factor: 500 GB
Parquet Store Size = 155.7 GB
CarbonData Store Size = 127.0 GB

Parquet Compression Ratio: 180.65
CarbonData Compression Ratio: 244.00

Loading Performance

The below chart depicts the data loading performance of CarbonData 2.1.0 (no_sort) and parquet.

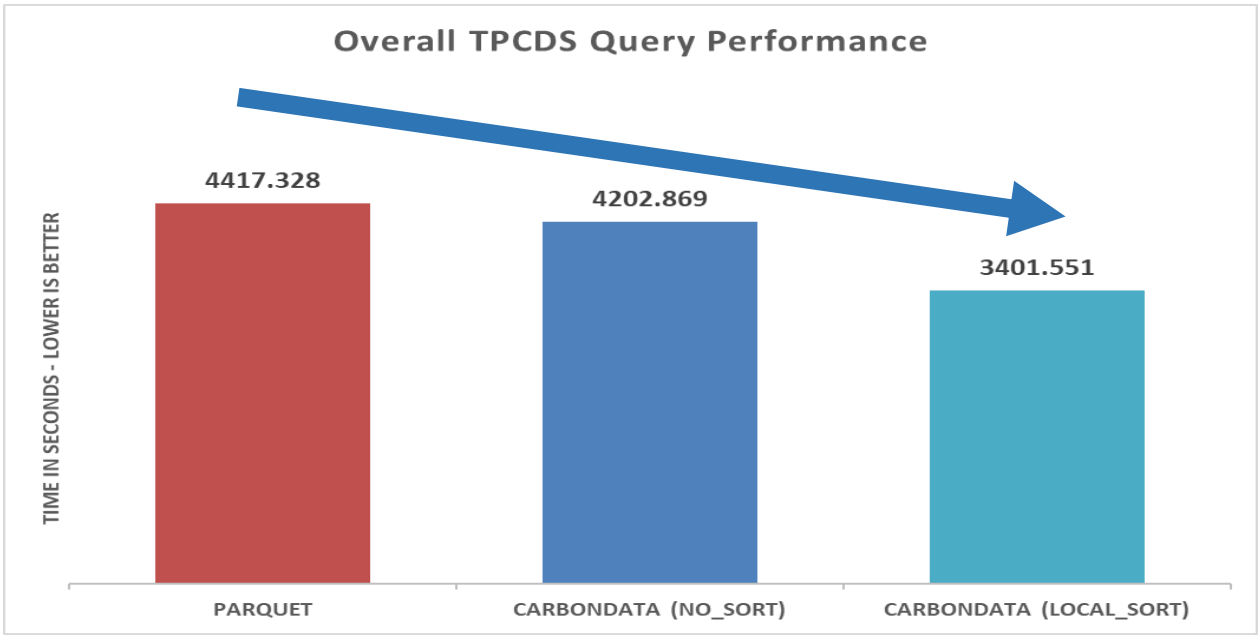


Summary:
CarbonData 2.1.0 (no_sort) has better performance in data loading compared to parquet.

Query Performance

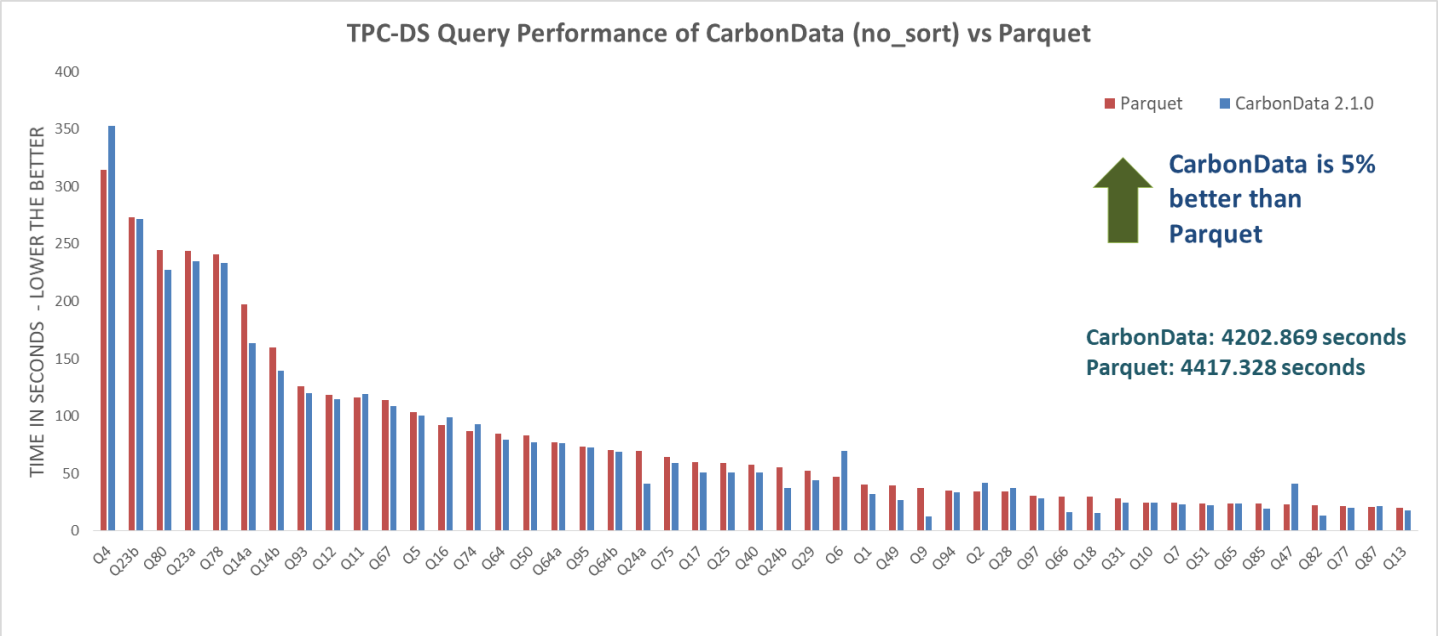
TPCDS Query Performance of CarbonData 2.1.0 with no_sort and local_sort compared to Parquet on Data Size of 500GB Scala Factor.

1. Overall Query Performance of CarbonData 2.1.0 compared to Parquet



Summary:
1 .CarbonData query performance is better than Parquet with default configurations.
2. Carbondata has multiple performance optimizations to suite various scenarios.
3. CarbonData query performance with local_sort has improved by 23% compared to parquet.

2. The below chart depicts the performance of CarbonData 2.1.0 and Parquet. We have loaded Carbon with **sort_scope as no_sort** option and Parquet loaded directly.

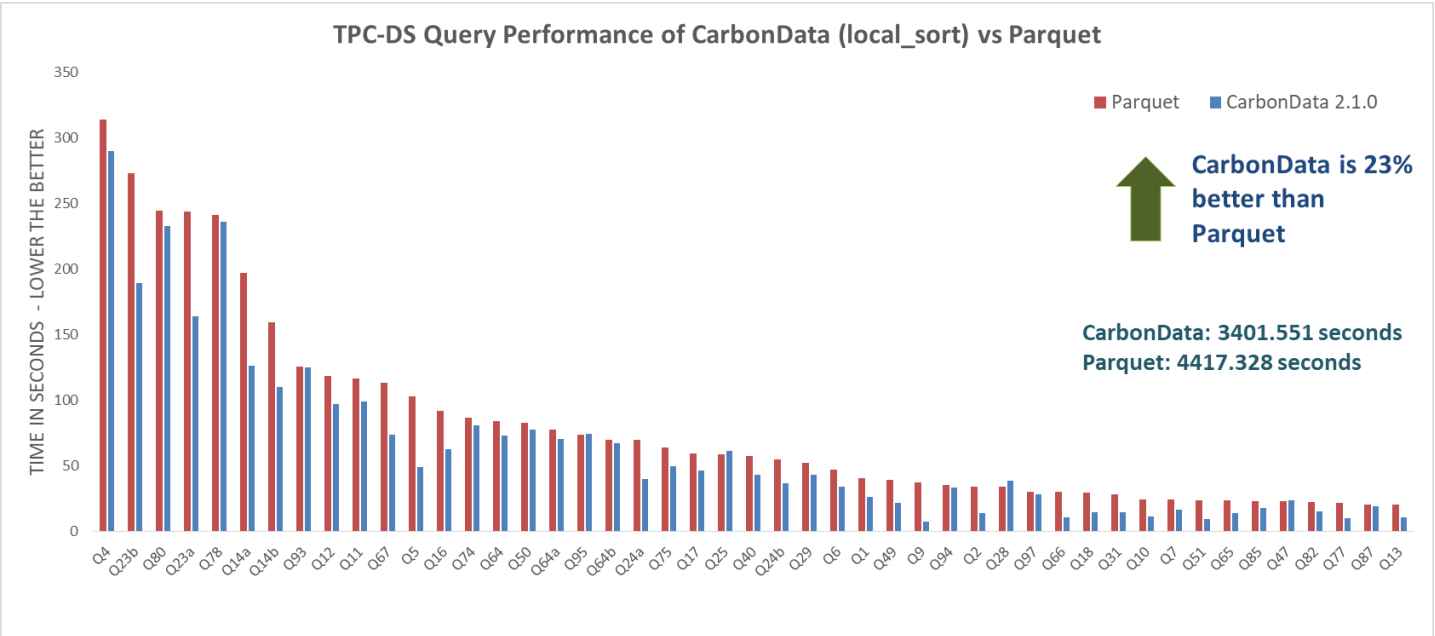


Summary:

CarbonData 2.1.0 query performance is 5% better compared to Parquet.

3. The below chart depicts the performance of CarbonData 2.1.0 and Parquet. We have loaded Carbon with **sort_scope as local_sort** option and Parquet loaded directly.

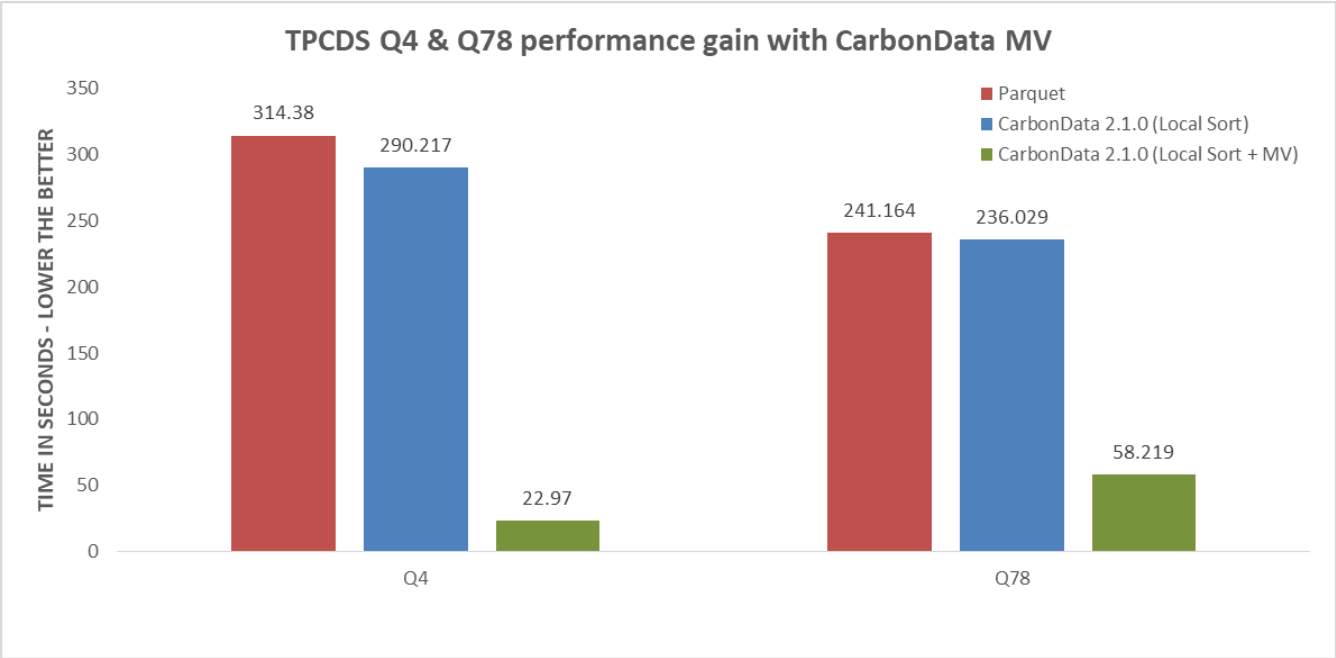
TPCDS Query Performance of CarbonData 2.1.0 (sort_scope='local_sort') compared to Parquet.



Summary:

CarbonData 2.1.0 query performance is 23% better compared to Parquet.

4. The below chart depicts the performance gain in CarbonData with MV enabled for Q4 and Q78 compared to parquet and CarbonData without MV



Summary:

CarbonData MV is configured for 2 queries and is having significant query performance improvement compared to parquet.

Note: If MV is configured for all the slow TPCDS Queries, Overall query performance can be improved by huge factor compared to parquet and carbondata without MV.

Query Performance Readings

Database Name	Parquet	CarbonData 2.1.0	CarbonData 2.1.0 Local Sort
Options/Table properties	spark.sql.parquet.compression.codec=snappy	carbon.column.compressor = snappy carbon.local.dictionary.enable = true carbon.push.rowfilters.for.vector = false sort_scope=no_sort	sort_scope=local_sort
Load Performance	1607 seconds	1563 seconds	3721 seconds
TPCDS Raw Data Size	436.9 GB	436.9 GB	436.9 GB
Table Format Database Size	155.7 GB	127.0 GB	127.1 GB
Query ID	Time in Seconds	Time in Seconds	Time in Seconds
tpcds_query1	40.562	32.092	26.771
tpcds_query2	34.238	41.988	14.121
tpcds_query3	5.192	37.031	4.83
tpcds_query4	314.38	352.74	290.217
tpcds_query5	103.429	100.32	49.525
tpcds_query6	46.993	69.965	34.604
tpcds_query7	24.369	23.176	16.506
tpcds_query8	11.337	15.719	5.588
tpcds_query9	37.34	12.637	8.006

tpcds_query10	24.672	24.703	11.499
tpcds_query11	116.603	118.949	99.341
tpcds_query12	118.641	114.723	97.239
tpcds_query13	20.375	17.664	10.855
tpcds_query14a	197.17	163.259	126.462
tpcds_query14b	159.801	139.51	109.972
tpcds_query15	12.501	7.914	5.435
tpcds_query16	92.064	99.069	62.614
tpcds_query17	59.784	51.008	46.512
tpcds_query18	29.86	15.298	14.65
tpcds_query19	17.396	8.892	6.644
tpcds_query20	9.007	3.7	2.995
tpcds_query21	2.871	2.02	1.964
tpcds_query22	2.794	2.945	2.822
tpcds_query23a	243.926	234.644	164.389
tpcds_query23b	273.414	271.768	189.62
tpcds_query24a	70.013	40.886	40.128
tpcds_query24b	55.094	37.355	36.761
tpcds_query25	59.192	50.832	61.476
tpcds_query26	16.754	9.494	5.527
tpcds_query27	15.372	9.249	5.684
tpcds_query28	34.182	37.113	39.064
tpcds_query29	52.625	43.978	43.588
tpcds_query30	14.314	10.144	10.81
tpcds_query31	28.355	24.724	14.932
tpcds_query32	9.471	6.006	4.904
tpcds_query33	17.155	9.173	6.248
tpcds_query34	6.018	5.215	5.596
tpcds_query35	15.659	12.685	10.872
tpcds_query36	10.695	9.291	6.247
tpcds_query37	10.646	6.547	8.124
tpcds_query38	17.283	18.505	17.076
tpcds_query39a	3.167	3.199	3.939
tpcds_query39b	2.285	2.767	2.736
tpcds_query40	57.777	51.107	43.196
tpcds_query41	1.55	1.57	1.468
tpcds_query42	6.239	4.517	2.283
tpcds_query43	4.799	5.016	7.565
tpcds_query44	8.876	8.736	8.936
tpcds_query45	9.056	6.058	6.119
tpcds_query46	9.144	9.84	7.193
tpcds_query47	23.095	40.913	23.614
tpcds_query48	10.464	12.246	9.296
tpcds_query49	39.458	26.969	21.833
tpcds_query50	82.944	77.545	77.937
tpcds_query51	24.175	22.67	9.901
tpcds_query52	5.357	4.57	2.332
tpcds_query53	5.388	5.608	3.76
tpcds_query54	19.503	11.839	10.946
tpcds_query55	4.51	5.364	2.381
tpcds_query56	17.153	9.822	6.401

tpcds_query57	11.017	14.874	14.304
tpcds_query58	13.094	9.204	6.064
tpcds_query59	12.823	13.12	14.265
tpcds_query60	12.776	9.159	6.053
tpcds_query61	11.8	13.088	9.171
tpcds_query62	4.624	3.135	3.473
tpcds_query63	5.237	5.491	3.77
tpcds_query64	84.593	79.527	73.208
tpcds_query64a	77.532	76.384	70.388
tpcds_query64b	70.225	68.86	67.173
tpcds_query65	23.934	23.73	14.096
tpcds_query66	30.257	16.73	10.711
tpcds_query67	113.828	109.023	74.055
tpcds_query68	14.866	17.806	6.517
tpcds_query69	13.944	11.8	9.089
tpcds_query70	10.848	15.303	11.091
tpcds_query71	12.218	10.676	5.889
tpcds_query72	14.467	11.36	11.475
tpcds_query73	5.4	7.337	4.351
tpcds_query74	87.164	93.155	81.007
tpcds_query75	64.272	59.362	49.792
tpcds_query76	12.295	11.617	9.6
tpcds_query77	22.03	19.978	9.981
tpcds_query78	241.164	233.612	236.029
tpcds_query79	8.647	22.757	5.932
tpcds_query80	244.786	227.663	232.866
tpcds_query81	11.211	10.713	10.942
tpcds_query82	22.601	13.654	15.488
tpcds_query83	9.022	7.081	4.53
tpcds_query84	7.395	6.879	6.07
tpcds_query85	23.564	19.44	18.031
tpcds_query86	6.954	9.319	4.201
tpcds_query87	20.748	21.572	19.081
tpcds_query88	16.432	23.251	16.923
tpcds_query89	6.23	10.597	4.557
tpcds_query90	4.052	3.717	2.529
tpcds_query91	4.688	4.828	3.605
tpcds_query92	6.568	4.816	3.295
tpcds_query93	125.918	120.008	125.35
tpcds_query94	35.403	33.847	33.555
tpcds_query95	73.797	72.489	74.654
tpcds_query96	2.411	4.257	2.401
tpcds_query97	30.621	28.402	28.728
tpcds_query98	6.509	14.722	3.645
tpcds_query99	12.876	5.239	11.562
Total	**4417.328**	**4202.869**	**3401.551**