Wisconsin Benchmark

Santiago Tobon & Nhan Le

Systems Chosen

PostgreSQL & BigQuery

Why PostgreSQL?

- We have used it once before but wanted to refine our skills with using it.
- It’s a object-relational database, allowing for the support of user-defined objects.
- Uses write-ahead logging making it highly fault-tolerant.
- Supports a extensive list of data types.
- Was convenient using postgres on GCP since it is easy to set up a vm instance for Postgres

Why BigQuery?

- BigQuery allows us to run complex analytical SQL based queries under large data sets.
- BigQuery is good for scenarios where data does not change often and we want to use cache as it has built-in cache.
- We can reduce the load on our relational database.
- BigQuery is performing much better for long running queries.

What were our Goals?

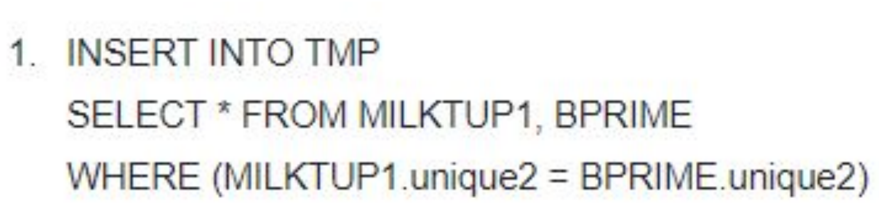
- To gain more experience using Postgres and BigQuery.
- To measure the performance of both systems and compare them to each other.
- To understand the join algorithm affect to the performance in Postgres and BigQuery.
- To analyze the performance of two systems based on clustered index and aggregate.

Experiment #1 Pre-Test

Describe: The benchmark was design to study the impact of the complexity of queries on performance of different database systems. It also helps evaluate the performance of join algorithms for different systems. This experiment is good to test the performance of Postgres and BigQuery using a JOIN.

Proceed: We are going to explore the performance in `MILKTUP1` table which contains 1,000,000 tuple relations. Then, we will execute the query on the right. This query will join the `MILKTUP1` with the `BPRIME` which contains 100,000 tuples. As a result, we will collect the execution time and analyze which systems would run faster.

Expected result: The query will run faster on Postgres than BigQuery.



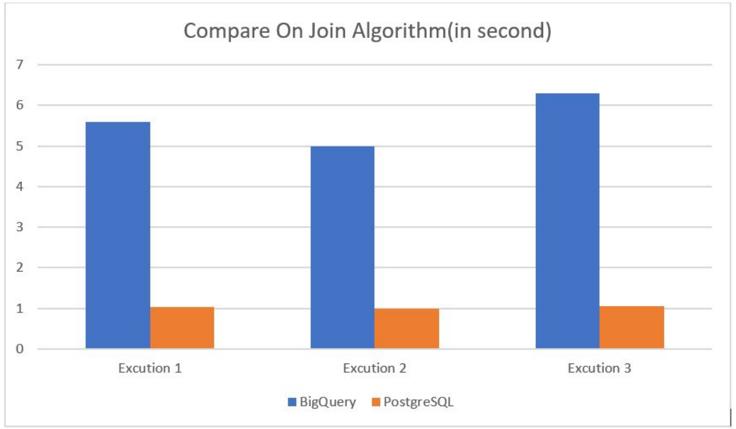
```
1.  INSERT INTO TMP
    SELECT * FROM MILKTUP1, BPRIME
    WHERE (MILKTUP1.unique2 = BPRIME.unique2)
```

Experiment #1 Post-Test

Expected result? Yes.

Lesson learned: Postgres is running faster because Postgres hash on the join predicate. Instead, BigQuery uses hash to shuffle the left and right tables until the matching key ends up. As a result, it is more expensive as data needs to be moved.

BigQuery	Postgres
Elapsed time (sec)	
5.6	1.0
5.0	1.034
6.3	1.053
AVG: 5.633	AVG: 1.029

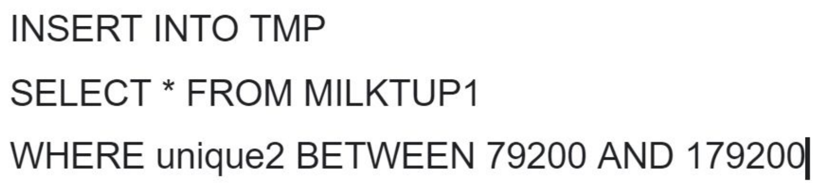


Experiment #2 Pre-Test

Describe: The selection queries in the Wisconsin Benchmark explore the effects on the performance. The experiment that we are testing is 10% selection via on clustered index. This experiment is good to test the performance of Postgres and BigQuery using clustered index.

Proceed: We are going to explore the performance in `MILKTUP1` table which contains 1,000,000 tuple relations. Then, we will execute the query on the right. This query will run a clustered index on both system BigQuery and Postgresql. As a result, we will collect the execution time and analyze which systems would run faster.

Expected result: The query will run faster on BigQuery than Postgres.



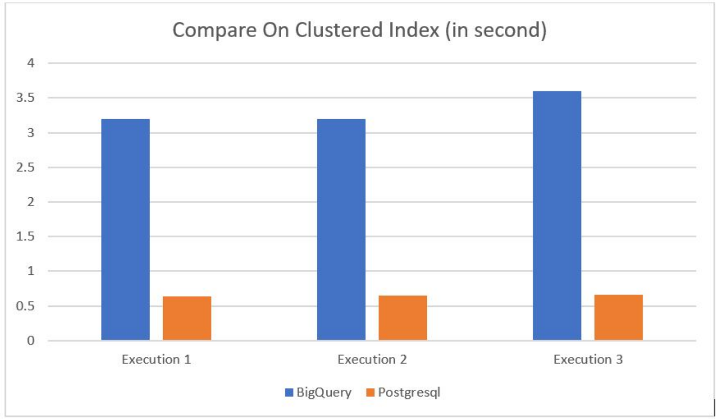
```
INSERT INTO TMP
SELECT * FROM MILKTUP1
WHERE unique2 BETWEEN 79200 AND 179200
```

Experiment #2 Post-Test

Expected results: No.

Lesson learned: As we have learned from BigQuery, it doesn't support the clustered index. Instead, it will scans all full columns to satisfy the condition. However, Postgres does scan base on the cluster index and is fast to get the result. Therefore, Postgres would run faster than BigQuery.

BigQuery	Postgres
Elapsed time (sec)	
3.2	0.641
3.2	0.653
3.6	0.663
AVG: 3.333	AVG: 0.652



Experiment #3 Pre-Test

Describe: Updating query is the weakest aspect of the benchmark including insert tuple, update key attribute, update non-key attribute and delete tuple. The principal objective of these queries is to examine the impact of clustered and non-clustered indices on cost of updating, appending or deleting a tuple. In this experiment, we are focusing on update key attribute with clustered index.

Proceed: We are going to explore the performance of this query in `MILKTUP1` table by updating a key attribute. This table contains 1,000,000 tuple relations. We will run the query on the right to compare the execution time on both BigQuery and Postgresql.

Expected Result: Postgresql will run faster than BigQuery.

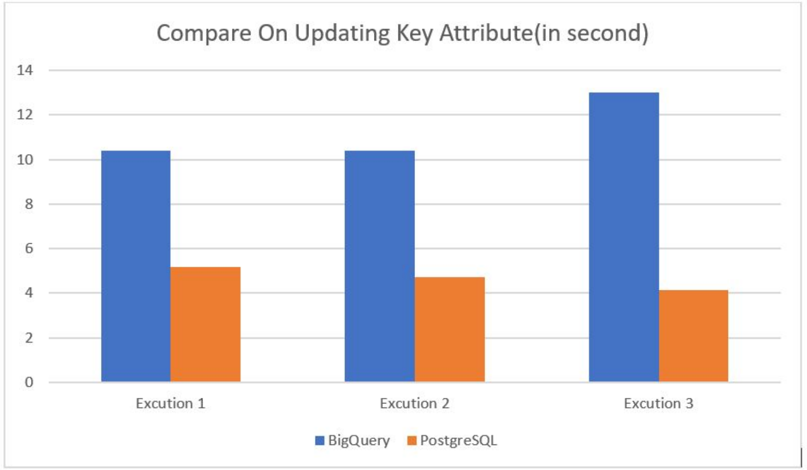
```
UPDATE MILKTUP1
SET unique2 = 1000001 WHERE unique2 = 970001
```

Experiment #3 Post-Test

Expected result: Yes.

Lesson learned: Repeatedly from the previous experiment, BigQuery updated value by scanning every single row until it finds the matching key and update data. Otherwise, Postgres gets the value from accessing the key in the condition and update data. Therefore, Postgres is faster than BigQuery.

BigQuery	Postgres
Elapsed time (sec)	
10.4	5.173
10.4	4.738
13.0	4.132
AVG: 11.267	AVG: 4.681

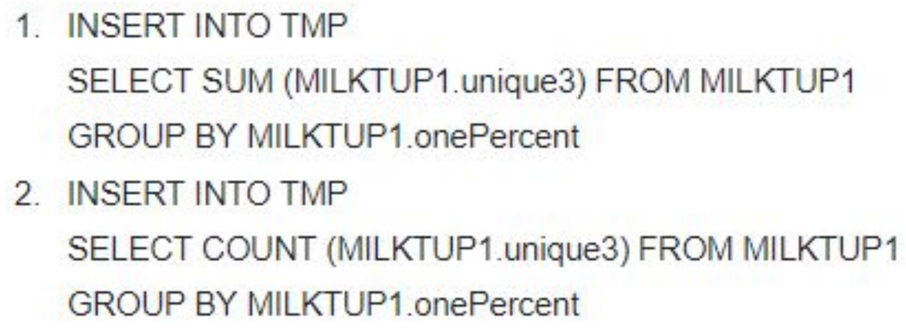


Experiment #4 Pre-Test

Describe: The benchmark was design to study the impact of the complexity of queries on performance of different database systems. It also helps evaluate the performance of aggregate functions for different systems. This experiment is good to test the performance of Postgres and BigQuery using a SUM and COUNT.

Proceed: We are going to explore the performance in MILKTUP1 table which contains 1,000,000 tuple relations. Then, we will execute the query on the right. This query will count and sum the unique3 values in MILKTUP1. As a result, we will collect the execution time and analyze which systems would run faster.

Expected result: The queries will run faster on Postgres than BigQuery.



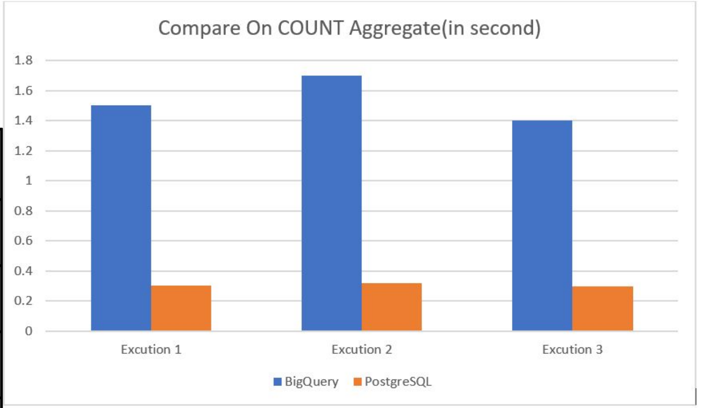
```
1. INSERT INTO TMP
   SELECT SUM (MILKTUP1.unique3) FROM MILKTUP1
   GROUP BY MILKTUP1.onePercent
2. INSERT INTO TMP
   SELECT COUNT (MILKTUP1.unique3) FROM MILKTUP1
   GROUP BY MILKTUP1.onePercent
```

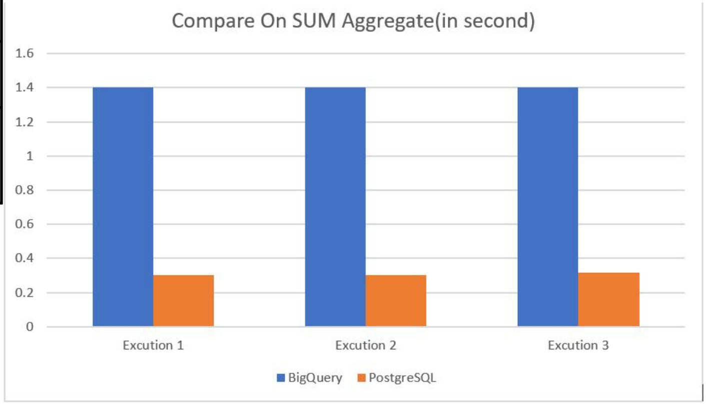
Experiment #4 Post-Test

Expected result: Yes.

Lesson learned: In BigQuery, an aggregate function is a function that summarizes the rows of a group into single value. Otherwise, it computes a single result from multiple input rows in Postgres.

COUNT		SUM	
BigQuery	Postgres	BigQuery	Postgres
Elapsed time (s)		Elapsed time (s)	
1.5	0.320	1.4	0.301
1.7	0.305	1.4	0.304
1.4	0.297	1.4	0.317
AVG: 1.533	AVG: 0.307	AVG: 1.4	AVG: 0.307





Summary/Conclusion

- For Postgres joins with small tables, Postgres tends to run faster than BigQuery. However, when it is a long running query, it may show a different result.
- Postgres performs better having a clustered index compared to BigQuery since BigQuery doesn't support clustered indices.
- Postgres is better for smaller databases since it has better query performance, but BigQuery has its value as being more scalable.

Lessons Learned

- The bigger the dataset, the better the performance using BigQuery.
- BigQuery is more stable than Postgres in term of increasing the big dataset. Postgres need to configure the memory in configuration file when the dataset is growing bigger.
- We should have started doing our experiments sooner since we had to redo them a couple of times.
- Would like to have done the experiments with another system like MySQL or Azure.