Redshift vs Postgres Sampling & Approximate Query Processing

-Khirod Sahoo, Amrit Bhat

INTRODUCTION

In the era of Big data, analysts seek to perform quick analytics and often sampling of data is done to estimate results for decision making. We wished to establish whether the use of these chosen samples using random sampling of a large dataset can be used to obtain reasonably accurate answers to the aggregation queries (GROUP-BY), in significantly shorter periods of time. Furthermore, we would like to know how the samples compare with the entire dataset in order to provide an adequate level of accuracy, and how much faster we can expect those answers to arrive. We used Approximate Query Processing(AQP) to estimate the results of two queries. AQP in database context is considered as an alternative to approximate answers that could have been obtained by running queries on a full dataset. It is designed primarily for aggregation like count, sum, average.

We initially wanted to compare the performance of two distributed systems- Snowflake and Redshift in terms of AQP, sampling execution times and overall query performance. But due to data loading issues, we changed the systems to test, while answering the same research questions. Inspired by the lessons learned in the class, we wanted to test the performance of a distributed system vs local machine. We used Postgres to run queries on a local machine. As we learned that distributed systems don't always perform the best in every scenario, we tried to evaluate this hypothesis by running different queries on data of different sizes and reported performance timings to answer our research questions.

We observed that the performance of Redshift remains almost the same in all the queries that we ran in research question 2 and 3 while Postgres performance changed linearly with the increase in data size and number of join orders. However, in research question 1, the sampling execution timings of Redshift were almost 6 times faster than Postgres on average. Also, in research question 2, we compared the results of two queries in full as well as sampled dataset and observed that the errors in the results are minimal and can be considered insignificant.

In the following sections, we discussed more about the methodology, data used, evaluated systems and results.

EVALUATED SYSTEMS

In our analysis, we evaluated 2 systems, one distributed system- Redshift and one local system- Postgres.

Redshift:

AWS Redshift is a service by Amazon AWS which is a fully managed, petabyte-scale data warehouse service in the cloud. It is a highly scalable data warehouse where you can cluster the required sizes as per the requirement and later scale if needed. AWS also hosts a cloud data storage service called S3 where data in multiple formats can be loaded and can be loaded into the cluster on Redshift.

Below is the architecture of AWS redshift.

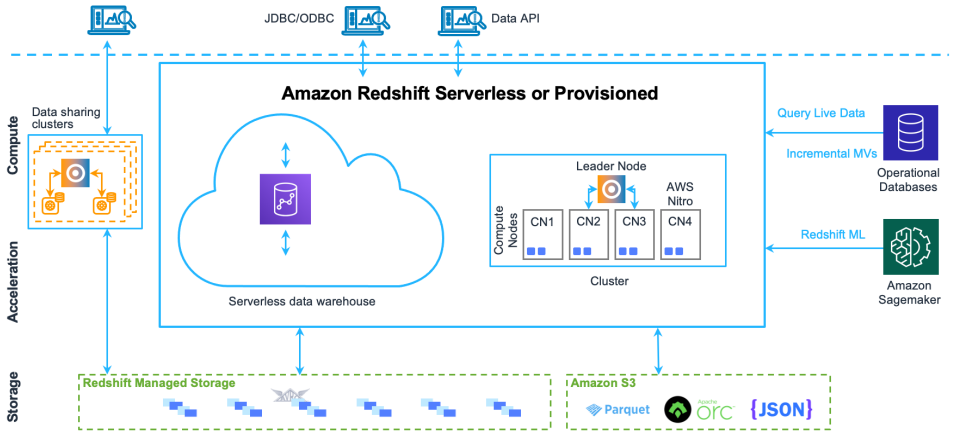

Figure 1

Amazon Redshift uses SQL to analyze structured and semi-structured data across data warehouses, operational databases, and data lakes. It's very easy to create databases and run SQL queries on Redshift.
In our project, we used 2 node clusters of 2 CPUs per node on AWS Redshift.

Postgres:

Postgres is an open source relational database system. Postgres allows transactional databases with Atomicity, Consistency, Isolation, Durability (ACID) properties, automatically updatable views, materialized views, triggers, foreign keys, and stored procedures. It is designed to handle a range of workloads from local machines to data warehouses or web services with concurrent users. It is the default database on MacOS server but also available for Windows and other servers.
Below is the architecture of Postgres.

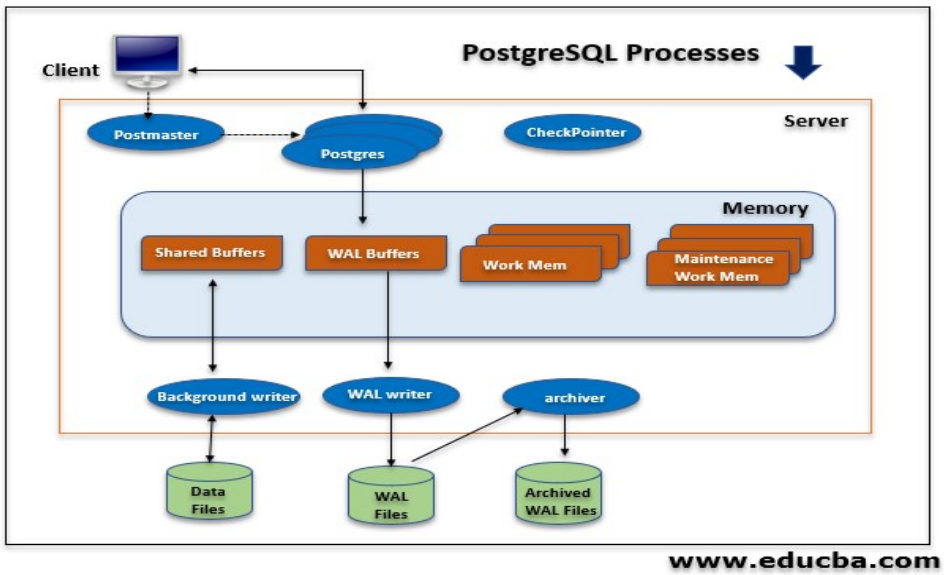

Figure 2

The local machine where we created the Postgres database has 24GB RAM and a 4 core CPU with Windows OS.

We used Postgres using a tool- DBeaver which is a free multi-platform database tool for developers, SQL programmers, database administrators and analysts. The platform is very intuitive and easy to create new databases and connect to different DB clients. Moreover , it offers SQL editor which is much more convenient over using Command Line Interface to run SQL queries.

DATA

We used the IMDB dataset in our project for answering the research questions. IMDB dataset first downloaded from https://datasets.imdbws.com/ had many data issues while loading in S3 bucket of AWS and hence, we used IMDB_pg dump from https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/2QYZBT.

PROBLEM STATEMENT AND METHODS

Research questions:

1. Are random sampling execution timings of both the systems similar?
2. How comparable are the results from Approximate Query Processing with full vs sampled dataset in both the systems?
3. How similar are the overall query performance in both the systems?

In order to answer the second question, we are comparing differences in Approximate Query Processing (AQP) between Redshift and Postgres using 50% of the full dataset and comparing

those results to query results on the full IMDB dataset. There were 5 steps in our methodology, as summarized in below chart:

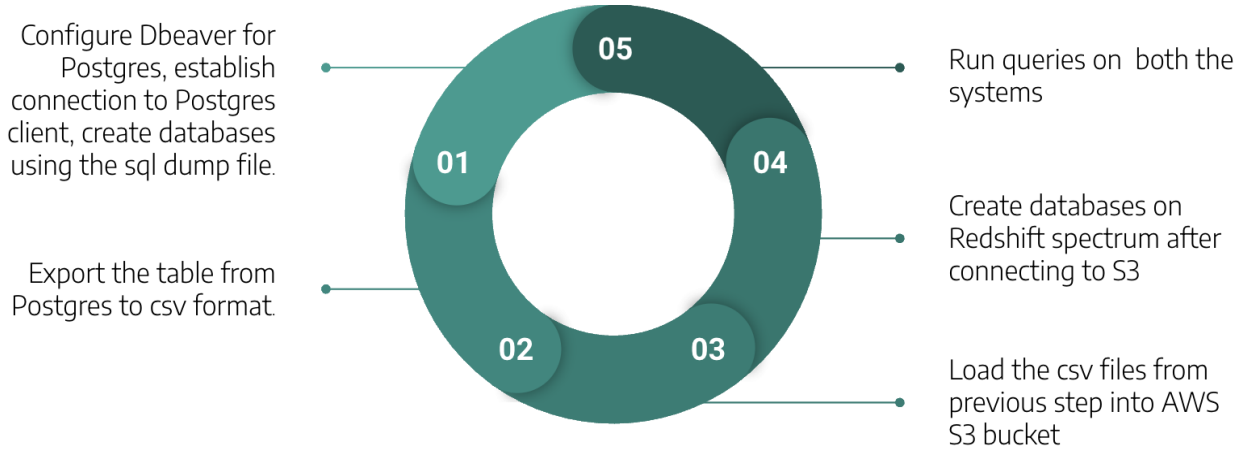

Figure 3

Data ingestion

For running queries on Postgres, we used the Dbeaver Postgres client to load complete data [1] from an sql dump file. Since we couldn't identify a way to create a database from a dump file on the Redshift server, we resorted to exporting the Postgres tables to CSV files which were then loaded into AWS S3 buckets. We created the corresponding database on Redshift spectrum from these uploaded CSV files and ran the below queries.

SQL Queries

1. *Sampling queries:*
   Random sampling was used to create a sample dataset with 50% and 30% of the complete data in cast_info and movie_info tables, which were the largest tables in terms of size of CSV files respectively (Cast Info: 1.2 GB, ~36 million rows, 6 columns & Movie Info: 900MB, ~14 million rows, 5 columns). These were as below.
   a. create table cast_info_s5 as select * from cast_info ci order by RANDOM() limit 36244344/2
   b. create table movie_info_s5 as select * from movie_info order by RANDOM() limit 14835720/2

2. *GROUP BY queries for Approximate Query Processing:*
   We modified two of the JOB queries [2] to include aggregations using the GROUP BY SQL command [3].

3. *JOB queries:* For understanding the overall performance differences between both

systems, we ran 6 standard queries from the JOB benchmark, without any modifications.

RESULTS

We noted the execution time of each of the queries in a document and performed graphical analysis to gain insights.

Research question 1. Sampling Execution time

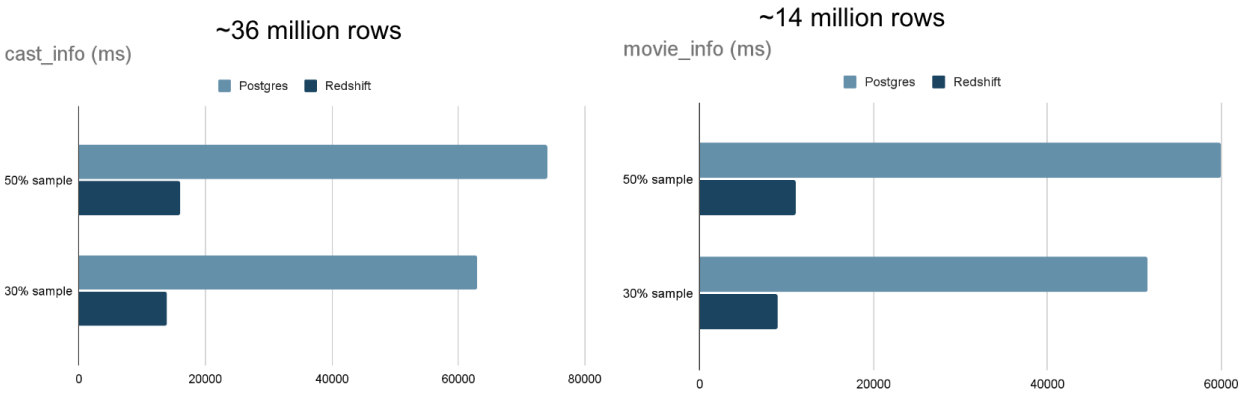

Figure 4

In both cases, we observed that Redshift is 6 times faster than Postgres in Random Sampling.

Research question 2. Approximate Query Processing comparison (Query results from full data and 50% sample of data were differenced to calculate error)

We ran two GROUP BY queries, both of which resulted in a number of aggregated rows calculating percentage distribution of each row id. Below graphs plot the difference in the percentage result computed on the full dataset and the 50% sample dataset

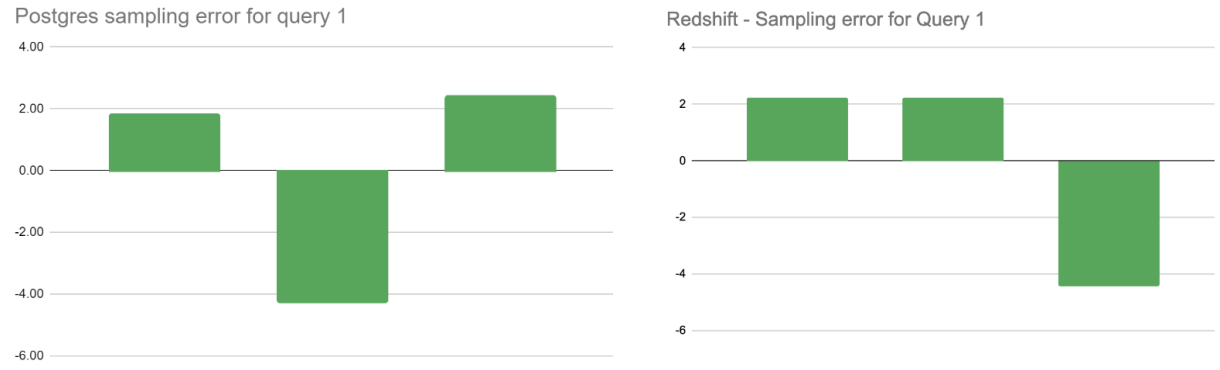

Figure 5

For all the rows, we see that the difference in percentage between the real value and approximate value is around 3 on an average for Query 1.

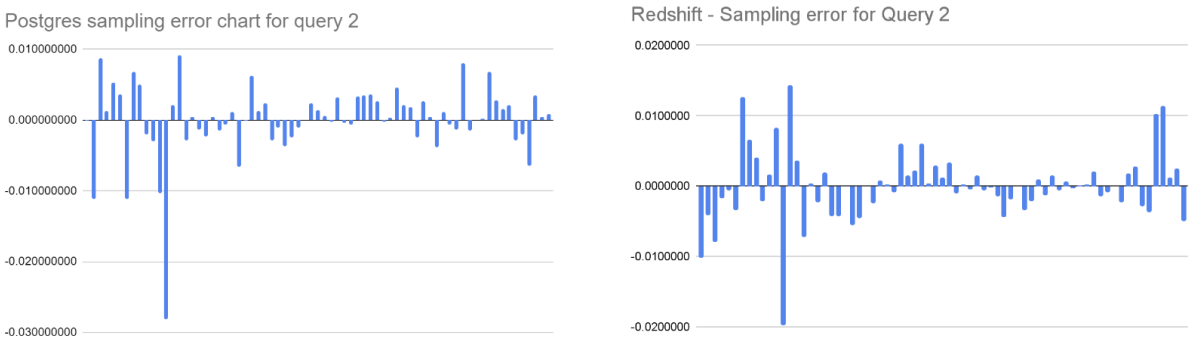

Figure 6

For all the rows, we see that the difference between the actual value and approximated value is less than 10^{-5} for Query 2.

The overall conclusion here is that the sampling error is close to 0 in both Redshift and Postgres. This means that using 50% samples using random sampling of a large dataset, we can obtain reasonably accurate answers to aggregation queries (GROUP-BY), in significantly shorter periods of time (6 times).

Research question 3. Overall Query performance

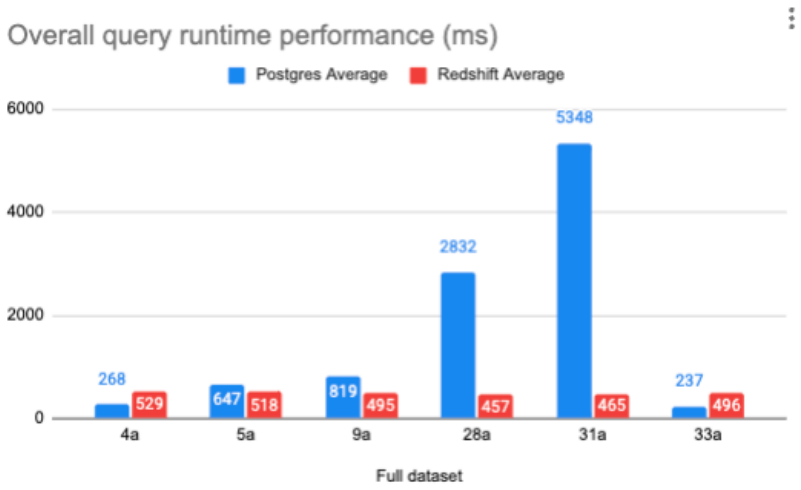

Figure 7

From above graph, there were 2 key takeaways:

- Postgres execution time increases with increase in number of joins and table sizes
- Redshift runtimes remained approximately the same in all the queries that we ran - no matter the number of joins and their cardinality.

CONCLUSION

Redshift is significantly faster than Postgres in terms of sampling execution time, almost 6X faster in our analysis. From the observations of our analysis, it was clear that distributed systems like Redshift are not always ideal to use, especially if the data we are querying is small and queries are not complicated (lesser number of join operations and predicates). Redshift's query execution timing remained almost constant in all the queries we ran. On the other hand, on a local machine, the query execution timing depended on the data size and the complexity of the queries. Also, we can conclude that sampled data is good enough for gaining quick insights and turning them into actions. As seen in above results, the error margin in the overall results and the approximated results is almost negligible and can be ignored.

REFERENCES

1. PostgreSQL Dump of IMDB Data for JOB Workload: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/2QYZBT
2. Join Order Benchmark: https://github.com/gregrahn/join-order-benchmark
3. Queries:https://docs.google.com/document/d/1D45XZfQFJQp8DtuREhW0AAQZnWhN2G3Ba5boH7-RjY4/edit
4. https://docs.aws.amazon.com/redshift/latest/dg/c_high_level_system_architecture.html
5. https://www.educba.com/postgresql-architecture/