Comparing Redshift to Snowflake Using Join Order Benchmark

Ross Friscia
rafrisci@uw.edu

1. Introduction

In the modern data landscape, there are a bevy of options to choose from when deciding where to store your data. One way to determine what database system is best is to choose the system that most efficiently retrieves the data. While the data returned to the end user looks the same using different systems, the backends are anything but. The data can either be stored in one server or across multiple nodes and clusters. Further data can be stored in row format or columnar. Finally, the query optimizers used to estimate the optimal query retrieval plan differ between systems. All of these factors can impact how well a system can retrieve information and scale as more data is included into the system. One way to compare system performance is to load the same database into both systems and compare how fast the systems retrieve the query information. For this project I will compare Amazon Redshift to Snowflake using Join Order Benchmark.

2. Evaluated Systems

Snowflake and Amazon Redshift are both distributed systems. The following subsections take a deeper dive into how the systems are structured.

2.1 Amazon Redshift

Amazon Redshift is a Mass Parallel Processing Data Warehouse that can work with large amounts of data and workloads at a high performance [1]. Redshift also markets itself as capable of possessing extremely fast processing times and high scalability. Redshift is a distributed system because the query load is split between a number of nodes. The user sends instructions to a leader node, which then distributes the query load to compute nodes possessing the data needed for the query. This relationship can be seen in **Figure 1**. Redshift also uses columnar storage for the data as it reduces I/O storage.

2.2 Snowflake

Snowflake is a Data Cloud that provides data storage, processing, and analytic solutions [2]. It also boasts its own SQL query engine designed for cloud processing. They claim their offerings are faster, easier to use, and more flexible than other database systems. Like Redshift, Snowflake uses columnar storage for its database storage. The query processing is done by virtual warehouses, clusters that are separate from the data storage clusters. The virtual warehouse has a compute cluster made up of multiple compute nodes. An example of Snowflake architecture can be seen in **Figure 2.**

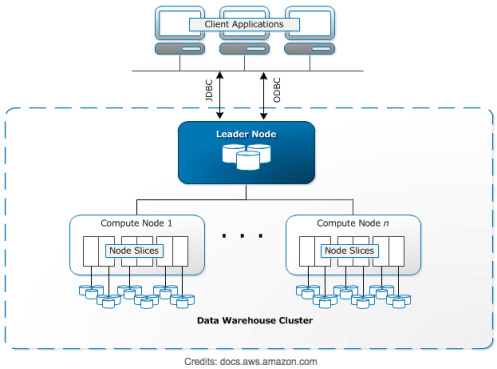

Figure 1

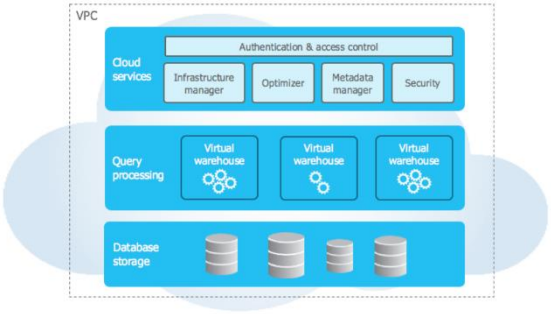

Figure 2

3. Problem Statement and Method

3.1 Problem Statement

While are many different ways to approach a comparison between Redshift and Snowflake, this analysis will focus on the following questions:

1. Which system completes queries faster?
2. Do the systems perform differently when increasing the computational power?
3. Do the systems perform differently with queries with more joins?

3.2 Methodology

3.2.1 Join Order Benchmark Data and Queries

Join Order Benchmark (JOB) is a benchmark that was introduced in "How good are query optimizers, really?" [3]. JOB involves querying IMDB dataset having 21 columns. When in a zipped folder, the total size of the data is 1.26 GB. There are 114 different queries the creators of JOB have made, each having a different number of joins and aggregations to test the query optimizers. There is an active GitHub repository where the queries are stored and a link to the data can be accessed [4]. For this analysis, 10 of the JOB queries are chosen, with an emphasis on selecting queries with different numbers of joins.

3.2.2 Data Ingestion

Just like how Snowflake and Redshift are similarly constructed systems that are ultimately different, ingesting the data into each system followed the same steps, but were different. Each system requires the csv files to be loaded into staging, then schema queries are run creating the tables in the database, and finally queries are run to load the csv data into the created tables.

For Snowflake, the csv's are loaded into staging using snowsql, a command line client that connects to Snowflake. Once the csv's are loaded, the data ingestion queries are run with the condition ON_ERROR = 'CONTINUE' as some data in the IMDB data is not formatted correctly.

For Redshift, the csv's are loaded into an Amazon S3 bucket for staging. Once the csv's are loaded, the data ingestion queries are run with the condition MAXERROR = x; where x is some number in the range of 1 to 100,000.

The number of rows for each table that are unable to be loaded are the same for both systems. **Table 1** breaks down the number or error rows unable to be loaded for each table. The table_name column is the name of the table, rows_loaded is the number of rows successfully loaded into the table for both systems, and row_err is the number of rows with errors that were not loaded into the table for both systems. For cast_info, movie_info, and person_info the number of error rows are greater than 100,000. Since 100,000 is the largest number of errors MAXERROR will allow to ignore, the csv's for these tables are split into smaller csv's using Python and loaded into the tables one at a time. For this stage of the analysis, Snowflake's claim of ease of use is true against Redshift.

3.2.3 Scaling the Data Systems

Scaling the data systems for Snowflake and Redshift are once again similar, but different. Redshift you can directly choose which types of nodes and the number of nodes in the cluster. For this project Redshift is evaluated using 2, 4, and 8 dc2.large nodes. Snowflake scales by changing the virtual warehouse size [5]. While you cannot see the exact storage limits on the different Snowflake virtual warehouses, virtual warehouse sizes are chosen by the sizes closest in cost/second to their Redshift equivalents. Snowflake small is compared to 2 Redshift nodes, medium to 4 nodes, and large to 8 nodes.

table_name	rows_loaded	row_err
aka_name	900662	676
aka_title	361376	3
cast_info	36124530	118792
char_name	3136382	3412
comp_cast_type	4	0
company_name	234825	172
company_type	4	0
complete_cast	135086	0
info_type	113	0
keyword	134110	60
kind_type	7	0
link_type	18	0
movie_companies	2604067	5062
movie_info	14355817	355351
movie_info_idx	1380035	0
movie_keyword	4523930	0
movie_link	29997	0
name	4167453	38
person_info	2025155	802302
role_type	12	0
title	2527799	170

Table 1

3.2.4 Timing the Queries

The queries selected to test range from 5 to 12 joins. The times recorded are done using warm cache query time and averaging five runs of the query.

4. Results

4.1 Which System Completes Queries Faster?

Figures 3 and 4 display the average query runtimes by query and nodes for Snowflake and Redshift.

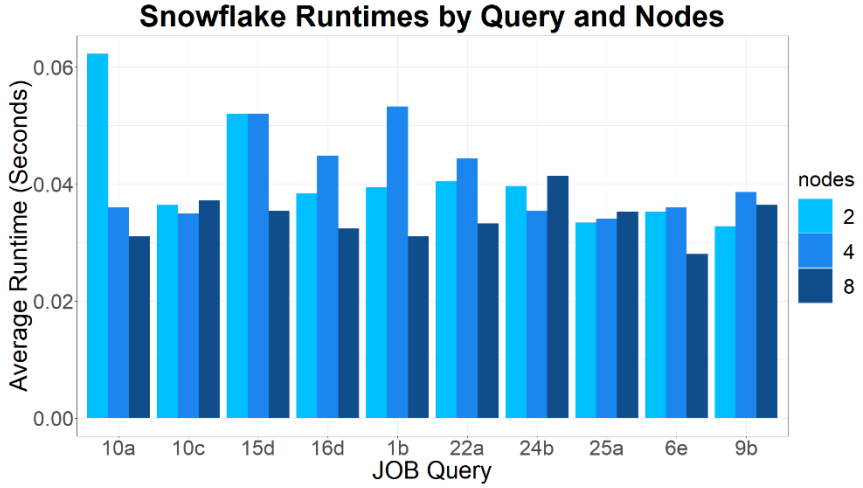

Figure 3

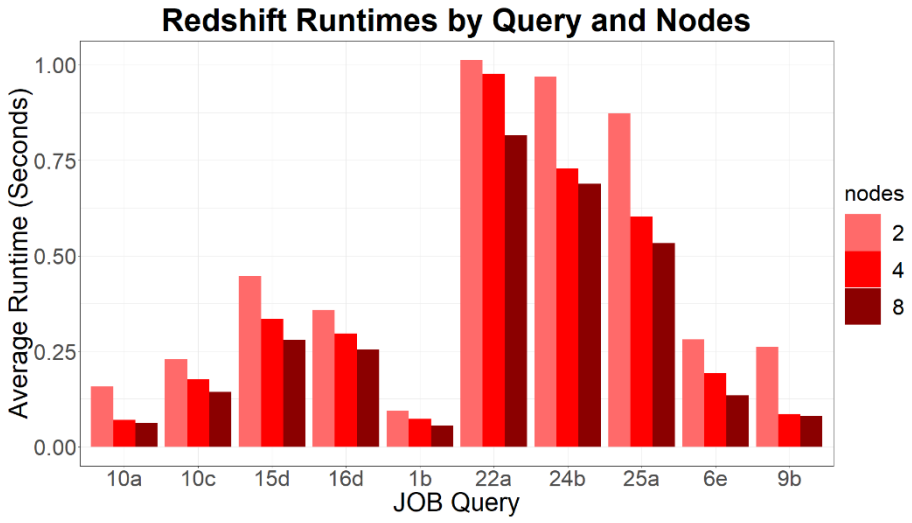

Figure 4

From these figures it appears that Redshift and Snowflake are similar in runtimes except for queries 22a, 24b, and 25a. Overall, it appears Snowflake run the queries faster.

4.2 Do the Systems Perform Differently When Increasing the Computational Power?

Looking at **Figure 4**, it appears that Redshift does consistently perform better when its computational power is increased. However, looking at **Figure 3**, Snowflake does not have such an obvious trend. In **Figure 5** the query average runtimes are averaged to compare the total average runtimes per node for each system.

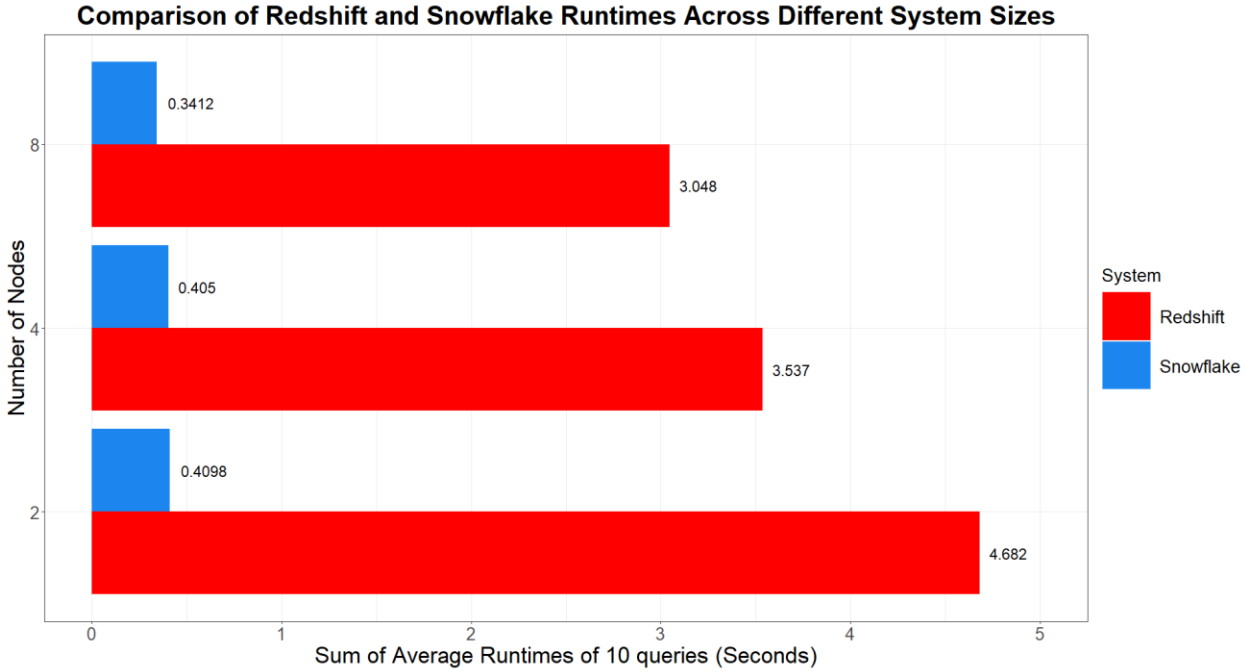

Figure 5

Redshift has an obvious decrease in runtimes as the number of nodes is increased. Snowflake has decreases too, but they are not as drastic. Despite this, the Snowflake runtimes are significantly faster across all node settings compared to Redshift.

4.3 Do the Systems Perform Differently with Queries with More Joins?

To compare how Redshift and Snowflake handle queries with different amount of joins the medium Snowflake virtual warehouse is compared to the 4 node Redshift cluster. The results are in **Figure 6**.

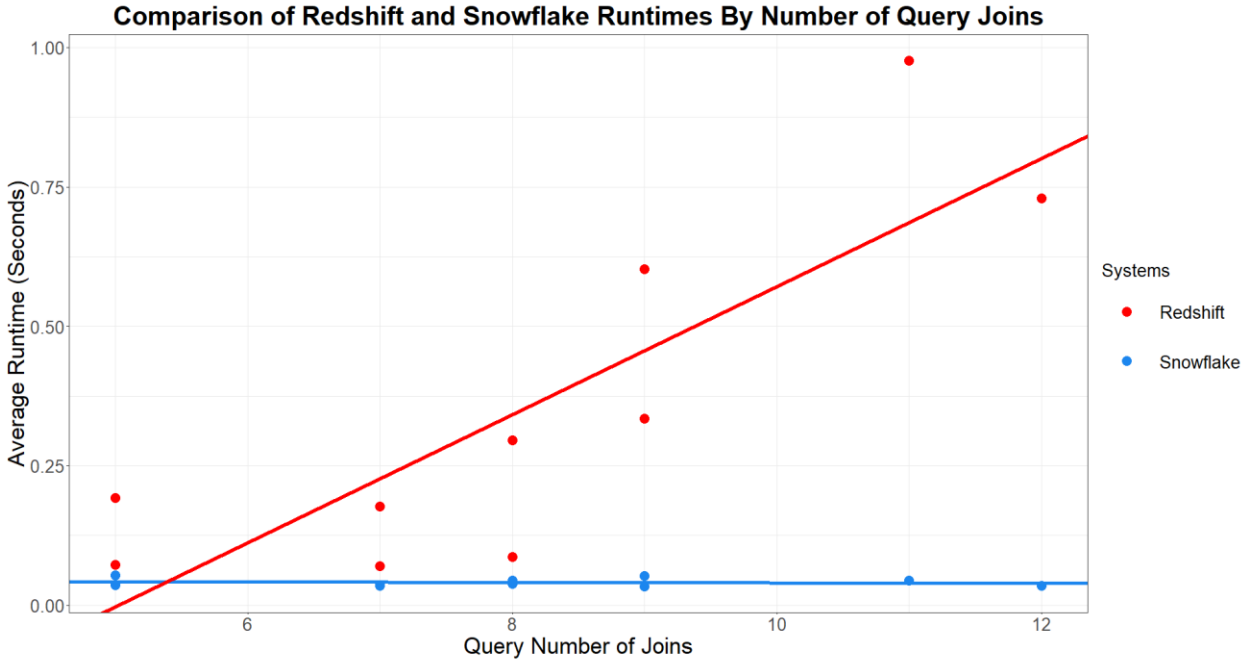

Figure 6

The runtimes for Snowflake queries do not appear to be impacted by the number of joins. On the other hand, Redshift's query runtime steadily increases with the number of joins.

5. Conclusion

Snowflake performs better in every test. This is despite its performance increases not being as drastic as Redshift's when given more resources. Snowflake handles queries with more joins much better, is a little better with queries that have few joins, and maintains its speed superiority at all tested scales. Not only that, but Snowflake is the easier system to ingest data into. The only edge Redshift has over Snowflake is the ease of uploading documents into staging. Snowflake is the easier and faster system to work with.

6. References

[1] https://hevodata.com/blog/redshift-architecture/
[2] https://docs.snowflake.com/en/user-guide/intro-key-concepts.html
[3] Leis, Viktor, et al. "How good are query optimizers, really?" Proceedings of the VLDB Endowment 9.3 (2015): 204-215.
[4] https://github.com/gregrahn/join-order-benchmark
[5] https://docs.snowflake.com/en/user-guide/warehouses-overview.html