Comparison of AWS Redshift and Snowflake on Real-World Data

Anushna Prakash and Preston Stringham

I. Introduction

AWS Redshift is a well-established database system that has an ability to scale and is widely used. Snowflake is a newer database system that can read data from Amazon S3 buckets and also boasts an ability to scale quickly and easily. However, how do these systems compare to one another on real-world, messy data? We will use the data available from IMDb under their personal and non-commercial license to write queries and compare the execution times on each system, with varying levels of complexity.

TPC-H is a commonly-used data set that was created to be of industry-wide relevance. It is used for benchmarking across different systems. The paper "How Good are Query Optimizers, Really?" by Viktor Leis et al. performed analyses that challenged using TPC-H as a benchmarking tool by comparing some unnamed database systems as well as PostGres and HyPer on 2013 IMDb data. While the join-order benchmark (JOB) queries written for that paper had increasing numbers of joins, they did not include any aggregations[1]. Our paper differs from that in that we will compare performance in terms of query runtime on queries with and without aggregations, with varying numbers of joins to increase complexity. We will use TPC-H data with an increasing number of joins and aggregations. We also revisit the IMDb data from the same source, with the data having grown from 3.9GB to 5.5GB since 2013. In this way, we can compare how Redshift and Snowflake perform on the standard benchmarking dataset, but also on real-world data to see if the performance is noticeably different between platforms.

II. Evaluated Systems

One of the two systems that we are comparing is Amazon Redshift. Redshift is a popular cloud-based database system that boasts the ability for its users to change the number of nodes in the database cluster for more processing power[2]. It has at least two nodes, a minimum of one compute node and one leader node. The leader node accepts connections from clients, parses the request, and performs aggregations when required. The worker node(s) does the heavy computational lifting, by doing the query processing and data manipulations.

The other system we are using is Snowflake. Snowflake allows users to access their data through S3 buckets, and shares that data with all virtual warehouses. Warehouses can elastically scale up the number of nodes within t-shirt sizes like x-small, small, medium, etc. having nodes that scale up by a factor of 2 for each size. There is a cloud services layer as well which is the so-called brain of the system, handling the "virtual warehouses, queries, transactions, and all the metadata that goes around that: database schemas, access control information, encryption keys, usage statistics and so forth"[3].

III. Problem Statement and Methods

The problem that we are trying to solve through this project is how popular database systems scale with the number of joins and aggregations. This is based on the work done by Leis

---

[1] "How Good are Query Optimizers, Really?" Leis et al. 2015.
https://courses.cs.washington.edu/courses/csed516/21au/papers/how-good-vldb-2015.pdf

[2] "Amazon Redshift and the Case for Simpler Data Warehouses" Anurag Gupta et al. 2015
https://courses.cs.washington.edu/courses/csed516/21au/papers/amazon-redshift-sigmod-2015.pdf

[3] "The Snowflake Elastic Data Warehouse" Benoit Dageville et al. 2015
https://courses.cs.washington.edu/courses/csed516/21au/papers/snowflake-2015.pdf

et al.'s, "How Good Are Query Optimizers, Really?" In this paper, a similar process of comparing popular database systems was done, but the queries used had no aggregation of any kind. This is the motivation for our work as aggregations are used very frequently among data scientists and analysts. To do this, we follow the steps of Leis et al., we use the IMDb and TPC-H databases to run our queries against. We will use Amazon's Redshift and Snowflake as our two benchmarking platforms. Both of these database systems are used at large scales by companies around the world. It is for this reason that we feel these database systems are perfect candidates for benchmarking our queries.

While datasets like the TPC-H are commonly used for benchmarking performance, the data can be a bit too simplistic and unskewed, which causes it to resemble actual use cases less. In this paper, we use the data available from IMDb to measure performance on increasing the number of joins and with aggregations. We expect to see a drop in performance as the number of both joins and aggregations increase. The IMDb dataset consists of seven tables that combined are about 5.5GB of data.

As mentioned in the introduction, our queries put an emphasis on aggregation as this not only differs from the work of Leis et al. but also addresses a key concept used by data scientists and analysts on a daily basis. We have compiled a suite of queries to test the database systems against. This suite contains queries varying in number of joins as well as aggregations in order to see how other database systems compare.

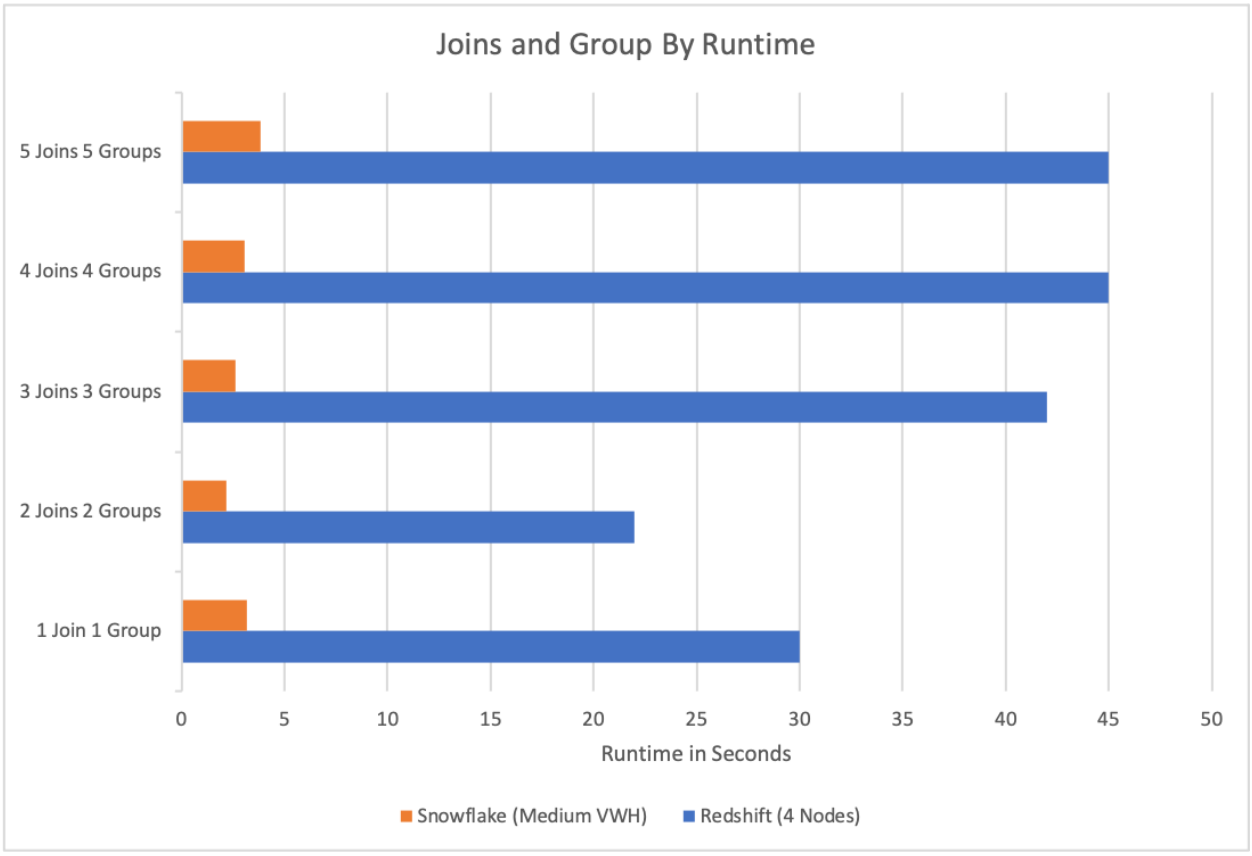
IV. Current Progress

With regards to how we split our work, Anushna handled preparing and ingesting the data into the various systems, while Preston wrote queries and performed timing. We helped each other when necessary, for example figuring out how to handle the array structures in the current IMDb data which was difficult to import into Redshift, converting file types from tsv to csv, and met frequently to work on the project and collaborate. We have also divided the work for writing the final paper.

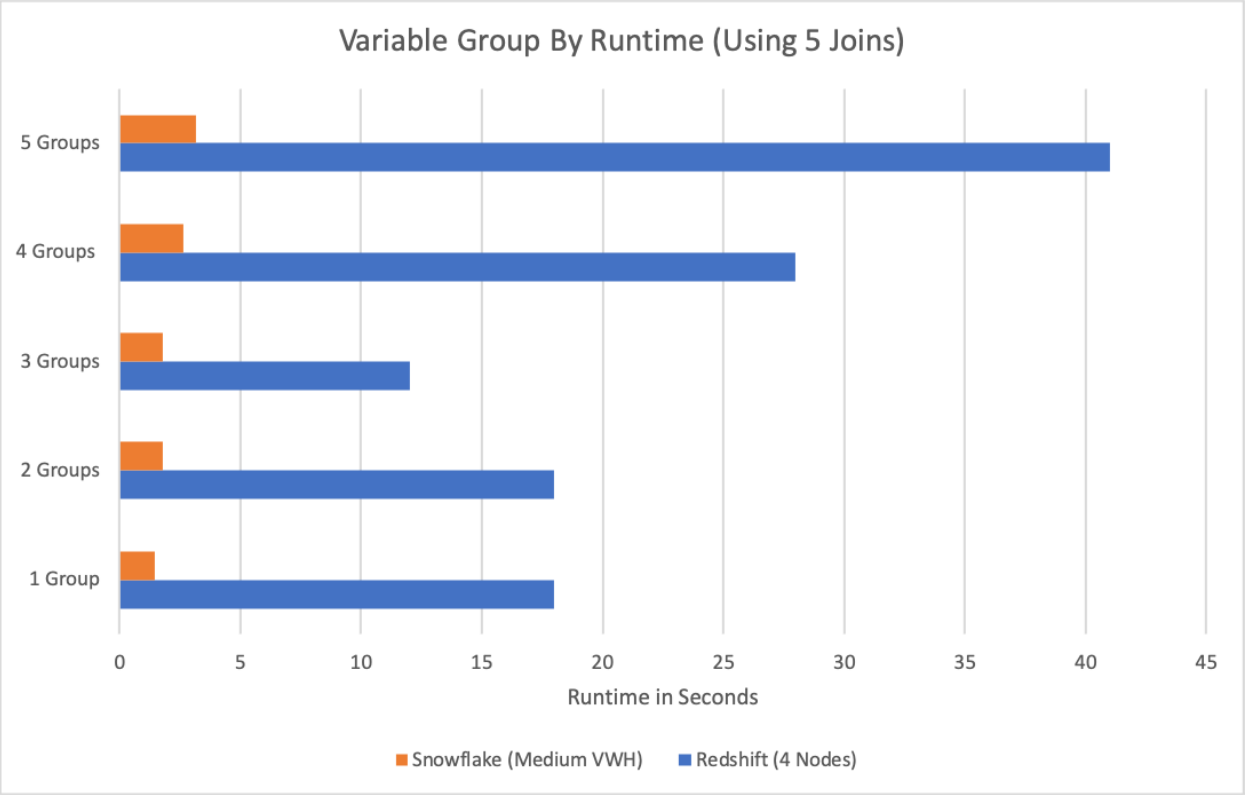
Currently, the TPC-H data is ingested into both platforms and has had the same queries run on both. The preliminary results of query runtime using warm cache timing is shown in Section V: Preliminary Results. The IMDb data has been ingested into Redshift and has had queries performed on it as well. The IMDb data has not yet been ingested in Snowflake as we are running into numerous technical issues with the process, including converting file types and allowing Snowflake access to the S3 bucket containing the data. This will need to be completed before December 7th. The queries for both data sets are fully written and will not require editing between platforms, minimizing the amount of work remaining after the data is ingested into Snowflake.

V. Preliminary Results

Below are our current results from running our queries. We have created queries which increase linearly with respect to the number of joins and group by elements.



Here we can see that Redshift using 4 nodes on our queries performed poorly when compared to Snowflake on a Medium virtual warehouse. This does make sense considering the amount of overhead that comes with using a large SaaS platform such as AWS's Redshift. In general, we found that the runtime increased as the number of joins and groups increased, which does correspond to what we learned in the JOB paper. We then ran queries that focused just on the group by clause in particular.



For this test, we took the query used from the first plot with 5 joins and varied the number of groups. We see that it clearly takes more time for both Redshift and Snowflake to form the groups, but Redshift in particular behaved much more inefficiently especially with a larger number of groups. Snowflake, on the other hand, performed much more to our expectations. It clearly increased linearly as the number of groups increased.

From now until December 7th, we plan on getting results using the IMDb dataset which, as mentioned previously, was not trivial for us. We will also begin to work on our presentation. Once we have ingested the IMDb dataset into Snowflake and run our queries, we will be able to create our final results which we will add to our presentation. In addition, we will also continue to work on our final report.

VI. Summary

What we have found so far is that the overhead in Redshift is causing much variation in the runtime of queries with an increasing number of groups. Queries run in Redshift do not behave as we would expect. We still have more investigation to do regarding the IMDb dataset as we were unable to load the data into Snowflake before the completion of the milestone. However, we have found a solution to importing the IMDb dataset to Snowflake, but we will not be able to provide results at the time of the milestone. Queries have been written already to test the IMDb dataset, so we should be able to produce results quickly once we have all of the data ingested into Snowflake.