Analysis of Amazon Redshift Parallel Ingestion

Dane Jordan
Master of Science Data Science
University of Washington
Seattle, WA
danejordan@live.com

***Abstract*—Amazon Redshift supports parallel data ingestion. When data is ingested from a single file, Amazon Redshift performs a serialized load, which takes longer than a parallel load. To perform parallel ingestion, the data must be split into multiple files.**

In this paper, we test and analyze parallel ingestion on Amazon Redshift looking specifically how ingestion times change based on the number of nodes on a cluster and the number of files the data has been parsed into. For the purposes of this study the "Node type" being used is dc2.large as using this node type reduces the costs of the analysis.

I. Introduction

Traditional DBMSs (database management system) allow users to execute queries quickly and efficiently on a structured dataset that has already been massaged into the correct format and loaded into a database. While very fast and efficient, they can be expensive, and the amount of time required to ingest data can be significant, slowing down the time to the first query result.

Loading the data from a single file forces Amazon Redshift to perform a serialized load, which takes longer. If there is a defined sort column in the table, then a VACUUM (resort rows) must run at the end of the ingestion.

To decrease the time to run a query on a dataset, the data can be ingested in parallel using multiple smaller files rather than one large file. With the concept of parallel ingestion, we pose research questions regarding how the number of nodes on a cluster or number of files used during the parallel ingestion, effects the time required to load the data. We also look at a couple of special cases regarding how the file is split into multiple files and sets of files in which the number of files ingested are close to the number of nodes on the cluster, but not equal.

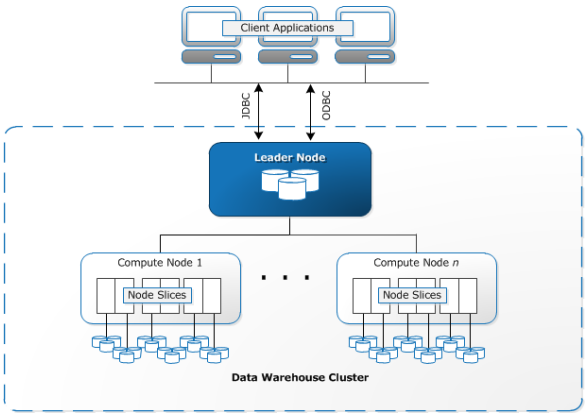
We see that there is direct relationship between parallel ingestion times and the number of file splits. Also, the number of nodes on a cluster decreases the ingestion times.

II. Evaluated Systems

A. Overall Architecture

Amazon Redshift is a relational DBMS that utilizes parallel processing, columnar data storage, and data compression to perform fast queries on large-scale datasets. While single-node clusters are available on Redshift, for the purposes of this analysis only multi-node clusters were used.

Figure 1: Node Relationships [3]



The data warehouse system architecture consists of a client application that is connected to a leader node on an Amazon Redshift cluster with a specified number of compute nodes, each with their own number of node slices. The compute nodes are run on an isolated network that is not directly accessed by the client but by the leader node that is connected to the client.

The client application in this case is SQL WorkBench. It is connected to the leader node via a JDBC connection. The number of nodes per cluster used for this analysis were 2, 4, 8, 16, and 32. For the dc2.large node type, each node contains 2 slices.

B. Highlighted Features

The focus of the analysis is on the data ingestion times into Redshift as this is one of the slower steps when using a DBMS. The focus is on utilizing Redshift's parallel processing capability by splitting a 7.6 gigabyte lineitem.tbl table from a 10GB dataset into smaller tables that are subsets of the original (i.e. two tables that each contain one half of the original table's data and are approximately 3.8 gigabytes in size each). The other feature that will be used when conducting the analysis is the number of nodes on the cluster. We will investigate how increasing or decreasing the number of nodes in relation to the number of files the data has been split into effects the ingestion time of the data.

III. Workload / Evaluation Method

While testing the data ingestion times is the most important part of the study to perform the analysis on Redshift's parallel DBMS, most of the work involved to

begin the study is done getting the data prepared and in a storage space such that Redshift can access the data. For this study, the data has been stored using Amazon's S3 (Simple Storage Service).

For more information on commands and code, please see the appendix.

Before beginning any operations to prepare the data, it is first obtained from another S3 bucket using AWS CLI (Amazon Web Services Command Line Interface). The data is transferred to an S3 bucket where we have direct access to the contents.

Using Python 3 and the boto3 library to connect securely to S3, the data is both read from S3 and downloaded from S3 to a local machine. To determine whether reading the data as an object or downloading the data and working with it locally is faster, the times are recorded for each. It is determined that it is more efficient to download the data upfront and work with it locally, as it is much quicker to pass over multiple times and we do not have to worry about memory issues when parsing the files later.

Prior to beginning any analysis, we began with creating the following research questions, and studies that would be performed to potentially answer them.

Unless explicitly noted, when multiple files are ingested, they are assumed to be evenly distributed (number of rows) in "chunks" such that a table split into four separate files would be the first quarter rows in the first file, the second quarter rows in the second file, etc.

1. **Does ingesting data in parallel decrease ingestion time?**
    - **a.** 1 file, 2 nodes
    - **b.** 2 files, 2 nodes
2. **Is the ingestion time using unevenly balanced files longer than that of evenly balanced files, assuming the same number of files and same number of nodes on a cluster? Also, is the ingestion time using unevenly balanced files shorter than that of a single file with the same number of nodes on a cluster?**
    - **a.** 2 uneven files, 2 nodes
    - **b.** Compare a with RQ1.b
    - **c.** Compare a with RQ1.a
3. **Does increasing the number of files beyond the number of nodes on a cluster decrease ingestion time?**
    - **a.** 4 files, 2 nodes
    - **b.** Compare a with RQ1.b
4. **Does increasing the number of nodes on a cluster beyond the number of files decrease ingestion time?**
    - **a.** 2 files, 4 nodes
    - **b.** Compare a with RQ1.b
5. **Does ingesting an odd number of files, such that the number of files is one less than the number of nodes on a cluster, take longer than ingesting the same number of files as there are nodes on a cluster? Conversely, does ingesting an odd number of files, such that the number of files is one greater than the number of nodes on a cluster, take longer than ingesting the same number of files as there are nodes on a cluster?**
    - **a.** 15 files, 16 nodes
    - **b.** 16 files, 16 nodes
    - **c.** 17 files, 16 nodes

After defining the research questions, using python 3, the lineitem.tbl is split multiple ways: 2 files, 4 files, 8 files, 16 files, and 32 files. There were also some special case splits including: 2 files alternating (i.e. all odd records go to the first file and all even records go to the second file), 2 files random (i.e. each file received a record at random, however, the two file sizes are even), 2 files uneven, 15 files, and 17 files.

These sets of files were subsequently uploaded to S3 using boto3 and AWS CLI into their own respective folders (each set of files received its own directory, so that each directory contained the entire lineitem.tbl). Once uploaded to S3, the Redshift clusters could be created, and the data can then be ingested using SQL WorkBench.

IV. Results

The ingestion times were recorded in a tabular format for all possible combinations between file splits and number of nodes on a cluster. Each combination is run warm start three times and the average ingestion time is recorded. First, we will look at the results pertaining to the research questions, then we will take a more holistic view of the data and visualize the analysis.

RQ1

Nodes	1 file ingest time (s)	2 files ingest time (s)
2	285	186

Yes, ingesting data in parallel decreases ingestion time. This is because the data is split into multiple chunks and ingested simultaneously.

RQ2

Nodes	1 file ingest time (s)	2 uneven files ingest time (s)	2 files ingest time (s)
2	285	220	186

Yes, the ingestion time using unevenly balanced files is longer than that of evenly balanced files.

Yes, the ingestion time using unevenly balanced files is shorter than that of a single file.

This is expected as uneven splits result with one file larger than the other. The larger file then takes longer to ingest than the smaller file. Although uneven, splitting a larger file in any way reduces the size of a single file.

RQ3		
Nodes	**2 files ingest time (s)**	**4 files ingest time (s)**
2	186	186
No, increasing the number of files beyond the number of nodes on a cluster does not decrease ingestion time. When there are more files than there are nodes on clusters, at least one node ends up being tasked with ingesting multiple files. Any node ingesting multiple files will take longer than other nodes that were only tasked with ingesting a single file of the same size.		

RQ4	
Nodes	**2 files ingest time (s)**
2	186
4	159
Yes, increasing the number of nodes on a cluster beyond the number of files does decrease ingestion time. This increase is minimal and not as impactful on ingestion times as parallelizing.	

RQ5			
Nodes	**15 files ingest time (s)**	**16 files ingest time (s)**	**17 files ingest time (s)**
16	41	41	51
No, ingesting an odd number of files, such that the number of files is one less than the number of nodes on a cluster, does not take longer than ingesting the same number of files as there are nodes on a cluster. The number of tasks assigned to the nodes is one less, however, that node with one less task must still wait for the other nodes to complete. Yes, ingesting an odd number of files, such that the number of files is one greater than the number of nodes on a cluster, does take longer than ingesting the same number of files as there are nodes on a cluster. This is because a single node has an extra task compared to the other nodes which must wait for it to complete.			

For the most part, the results are indicative of what we would expect when adjusting the parallel component (number of files) and the number of nodes on a cluster. Research question #4 raises some questions about why the ingestion time would decrease when increasing the node size is increased beyond the number of files that are being ingested, however, this is most likely related to the total number of slices, which is twice the number of nodes. Figures 2, 3, and 4 shows the overall trend when directly comparing parallel processing capability and the number of nodes on a cluster.

Figure 2: Ingestion Times (heatmap)

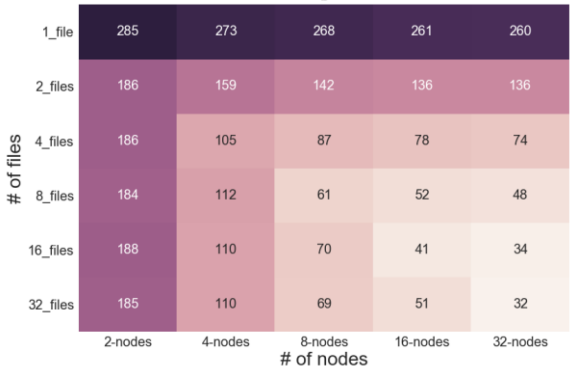


Figure 3: Ingestion Times (line graph)

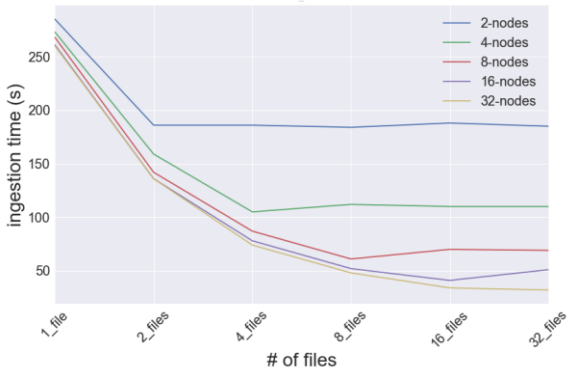
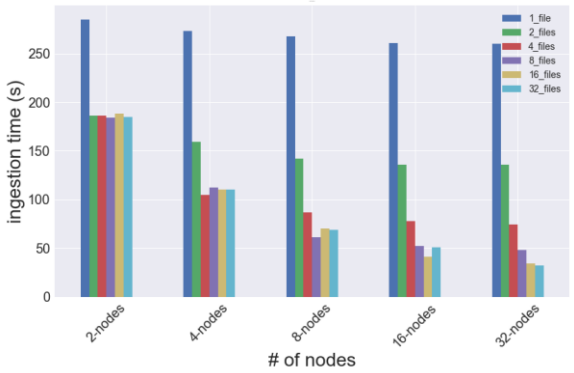


Figure 4: Ingestion Times (bar graph)



Pertaining to research question #2, we also experimented with other variants on a two-file split for parallel processing as mentioned earlier. The 2-file split variants include:

- 2_files – Two files split such that the first half of the records were in one file and the second half of the records were in a second file.
- 2_files_alternating – Two files split such that the odd records were in one file and the even records were in a second file.
- 2_files_random – Two files split such that there is a 50% chance a record could go into one file or the other.
- 2_files_uneven – Two files split such that the file size of one file is double the size of the other.

Figure 5: 2 File Splits (line graph)

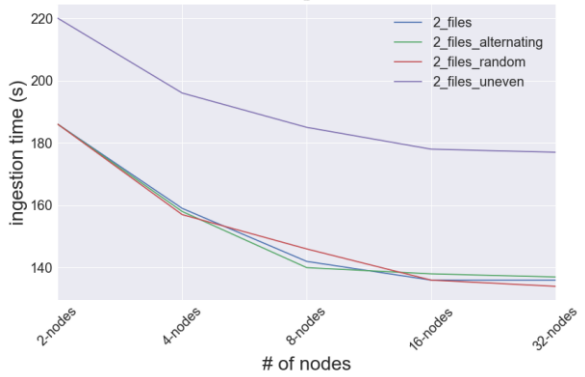
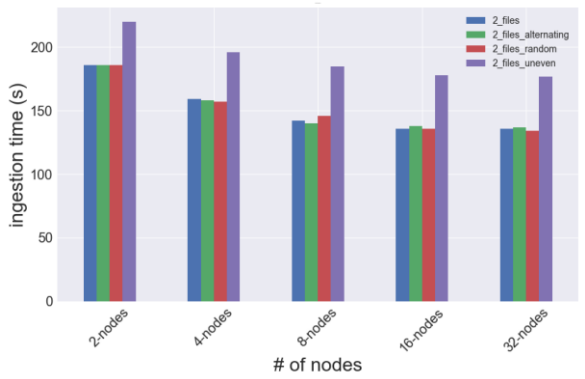


Figure 6: 2 File Splits (bar graph)



Figures 5 and 6 show the results for the ingestion times based on the two-file split methods above.

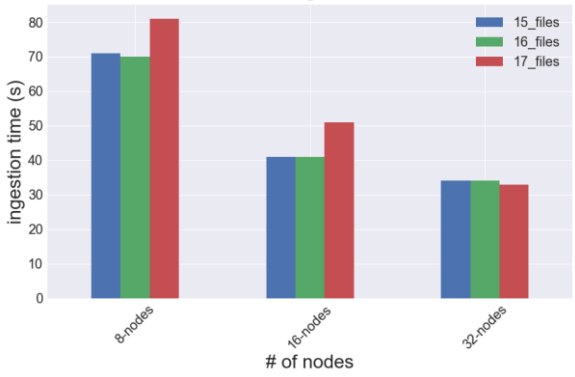
We see that the only method for splitting files that seems to affect the ingestion time is when the files are split unevenly. This seems to make sense as the file size (in bytes) is more determinant of the ingestion time than the ordering of the records in the files.

We determine that for research question #5, only a file split count that is beyond the number of nodes on a cluster caused a longer ingestion time. Figure 7 shows the ingestion times pertaining to file splits of 15, 16, and 17 files on clusters with 8, 16, and 32 nodes.

V. Conclusion

Amazon Redshift is a powerful DBMS capable of ingesting data in parallel and significantly decreasing the time until the first query can be run against a dataset. The number of nodes on a cluster directly affects the ingestion time of a dataset; the more nodes, the faster the ingestion. There does appear to be a minimum bound ingestion time as the rate at which the ingestion time decreases is also decreasing with each exponential node increase. Ultimately, the overhead required to ingest in parallel across many nodes would most likely reach a point where it is detrimental to the ingestion time and no longer provides any added benefit.

Figure 7: Odd Number File Split (bar graph)



Similarly, the number of files created to parallelize the ingestion also seems to have a limit as the ingestion time from an increase in the number of files in a split appears to decrease at a decreasing rate. Eventually, the overhead required to handle such a large split of files would counteract the parallelization and no longer be optimal for minimizing the ingestion time.

Based on the various two-file splits, we see that the ingestion times are unaffected by the ordering of the records in the split, but are affected directly by the file sizes in a split (even file size versus uneven file size). Further investigation could lead us to analyze file splits of more files with more uneven bounds per file to determine how the leader node handles the distribution of the files (i.e. 2 nodes and 6 files, where the files are split in such a way that they are all different sizes, but can be split into two groups of 3 for a total file size that is equal). Would such an example yield the same results as 2 nodes with 6 evenly distributed files?

Amazon does recommend splitting large files to optimize ingestion time, but rather than splitting based on the number of file subsets, data should be split to a size per file. They provide documentation on the best way to optimize your ingestion times based on your node sizes, node types, and data size. Redshift's massively parallel processing (MPP) is highly efficient and allows for the ingestion of data to be optimized, vastly reducing the time to begin querying data.

References

[1] DeWitt and Gray, "Parallel Database Systems: The Future of High Performance Database Systems," Communications of the ACM. 1992.

[2] Anurag Gupta, Deepak Agarwal, Derek Tan, Jakub Kulesza, Rahul Pathak, Stefano Stefani, and Vidhya Srinivasan. 2015. Amazon Redshift and the Case for Simpler Data Warehouses. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15).

[3] Amazon Redshift System Overview. (n.d.). Retrieved November 04, 2017, from http://docs.aws.amazon.com/redshift/latest/dg/c_redshift_system_overview.html