panoply

Benchmarking Tableau Performance

Panoply vs Amazon Redshift

Table of Contents

Faster Visualizations from Data Warehouses 3

The Plan 3

The Criteria 3

Learning the Data: The Panoply Machine-Learning Approach 4

Performance Testing 4

Panoply Optimizations: Keys, Compression, and Queries 4

The Results 6

What We Learned 8

Faster Visualizations from Data Warehouses

Harnessing big data insights is an integral part of modern business analysis. Now that integration and consumption of big data have hit the mainstream, the new hurdle for business intelligence visualization is improving rendering performance and visualization speed. Finding data management tools that can reduce the time to visualization is the next great challenge to business intelligence's agility and responsiveness.

Tableau, a leader in visual analytics platforms, approached Panoply, the world's only Smart Data Warehouse, to determine how it would optimize and manage dashboard runtimes and query requests submitted to a dataset through Tableau's Business Intelligence (BI) Tool. Once determined, these same testing criteria were utilized in queries against Amazon's Redshift Data Warehouse.

The performance study was managed by Tableau. The database used was at baseline: no performance enhancements like compression, caching, sort keys, distribution keys, etc. were applied to either warehouse dataset.

The test would allow IT professionals, database architects, data scientists and others to gain insights about the performance and optimization capabilities of the providers, and give business users information on the rendering speed, ongoing optimization capabilities, and machine learning capabilities of each data warehouse. Top line performance is a key factor in tool selection, as ongoing optimization by data scientists and engineers requires time and expertise. Panoply's platform-based automatic optimization eliminates the need for ongoing manual optimizations, and the test assisted study managers in assigning metrics to the efficiencies these optimizations created. We'll examine the testing process, analyze the results and explore key takeaways for the improvement of business intelligence practices utilizing Panoply and Redshift.

The Plan

Tableau's goal is always to get the best, most useful visualization to the user in the fastest response time. The performance test was designed for the Tableau program to execute a set of dashboards, which in turn would generate queries against both data warehouses. Nothing was changed in either dashboard between tests except the connection string: the data source was either a direct connection to Redshift or an "indirect" connection to Redshift through the Panoply platform.

This testing mechanism simulated a user's total reliance on the tool to auto-generate a query – not the user's ability to write or understand SQL. This simulation most closely matches the typical Tableau user experience of creating self-service visualizations utilizing drag-and-drop operations, not complex code. No additional steps were taken to optimize any of the queries used. The sole responsibility for optimizing queries for fast execution rested with each data warehouse and the user in charge of optimization, usually an engineer within the IT department.

The Criteria

Devising criteria that were both robust and realistic was an important part of the performance testing framework. The following parameters were used:

- Synthetic TPC-DS datasets were leveraged for testing
- Datasets ranged between 4GB to 200GB
- The primary fact table contained between 180M to 3B rows of data
- To achieve parity, the same data was loaded into both Panoply and Redshift

Once the data was loaded into each data warehouse, it was put through a series of tests with results reported by Tableau. Tableau's benchmark dashboards were used throughout the testing for both Panoply and Redshift query results. The results below are based on several rounds of testing and demonstrate optimization results via increasing or decreasing dashboard render times. Both dashboard and individual query execution times were tracked. However, because a user's perception of performance is tied to how quickly the dashboard they run completes the rendering process, we focused on the rendering time of the visualizations themselves.

Learning the Data: The Panoply Machine-Learning Approach

As a Smart Data Warehouse, Panoply's learning process starts when the data is ingested and the first query is executed. Once the analyst begins to query the ingested data, Panoply begins its work. This learning and knowledge gathering process is critical and contributes to better performance and enhanced decision-making processes within the platform.

Panoply is designed to optimize automatically based upon the usage patterns of the analyst or data scientist initiating the queries. In the test, these optimizations were performed within Panoply's automated platform by a proprietary set of machine learning algorithms, which learned the business logic and adjusted for optimal performance. Some of the automated optimizations included data compression, distribution key creation, and sort key creation.

Performance Testing

The importance of Panoply's ability to learn the data showed its significance during the performance testing. The tests were designed as straight "apples-to-apples" comparisons. To ensure comparison equivalency was possible, special care was taken to create a uniform size and composition of the Redshift cluster backing Panoply, and to that of the standalone "vanilla" Redshift cluster. The first baseline run column was labeled as 'Redshift-Baseline' or 'Panoply-Baseline'. Each row in the chart displayed a different dashboard executing one or more queries. This testing methodology was performed with Panoply as well as the vanilla Redshift.

During the test period, three batches of data were run against a Panoply Smart Data Warehouse and an Amazon Redshift Warehouse. The performance was sampled in each batch with changing intervals to identify performance improvements. The data warehouse and all optimizations were cleared as each new batch of data was introduced into the system.

With these testing parameters in place, Panoply immediately began to "learn the data" and identify helpful usage patterns upon initial analysis. Based on the patterns identified, Panoply then determined which optimizations to make and to what extent they should be implemented.

Since all optimizations are managed within Panoply, the platform can adapt to a variety of use cases. The more frequently the data is queried, the more deeply the platform understands the data and the more precisely the machine intelligence will adapt to improve performance.

Panoply Optimizations: Keys, Compression, & Queries in Data Learning

Panoply applies distribution and sort key setting, compression, query, and view materialization optimizations during its data learning process.

Distribution & Sort Keys

The distribution key is essential to optimization because it directly impacts performance, and defines how data is the spread across nodes. Panoply has automated the process of designating a distribution key based on "learned data".

The spread of data within a node is determined by the sort key. Panoply can intelligently select the correct column as a sort key. In Redshift, sort keys allow for less data scanning when the user utilizes WHERE clauses in a query. In Tableau, a user-defined filter which is "covered" by a sort key in Redshift generally enhances performance significantly.

Redshift does not utilize constraints on the primary or foreign keys – these are informational only. Panoply, however, can add constraints through its internal processing when it is being utilized by other tools in their query building process. There are two options for defining constraints within a Panoply database. One is setting them by user via Panoply UI. The other is to allow Panoply's algorithm to identify fields with id/id_ pattern values. Based upon these results, Panoply will define the field(s) as primary keys/foreign keys (PK/FK).

Compression

Compression is an effective method of reducing the size of stored data. By reducing the data size, query performance can increase due to reductions in disk I/O. The more data used, the more important compression becomes. Redshift selects the compression configuration during the initial copying of the database, and then continues compression of established configuration.

Panoply also handles compression on the initial copy, but then continues to monitors the data over time, adjusting the compression configuration as needed, particularly after changes or introductions of new data segments. By periodically optimizing via compression, Panoply improves the overall responsiveness of the dataset to rendering and visualization requests.

Queries

Panoply keeps track of the user's queries, specifically their usage of tables in joins and fields in ON clauses. Based upon this information, Panoply defines the distribution key and style appropriate for each table and generally for the database. This allows future queries to better leverage the distribution keys and more easily access data from different nodes. For example, a field that is often used when joining a certain table would be a statistically strong candidate for a distribution key.

In the case of query caching, Panoply can dynamically define queries to cache. Every query executed on top of Panoply goes through a proxy server. The query is analyzed for metadata purposes (execution time, row count, number of executions, etc.), and a learning algorithm evaluates the aggregated statistics and makes the decision to cache the query or remove it from cache. Panoply uses layers of memory throughout its infrastructure, allowing it to dynamically allocate the cached results between database storage and the proxy external memory.

View Materialization

Query information is combined with an analysis of the query patterns Panoply observes during testing to optimize for view materialization. Panoply takes advantage of both cached results and materialized views, and decides to cache all or a portion of the queries in memory based upon user statistics from queries. By analyzing usage, runtime and frequency, Panoply decides which queries or views should be materialized or cached.

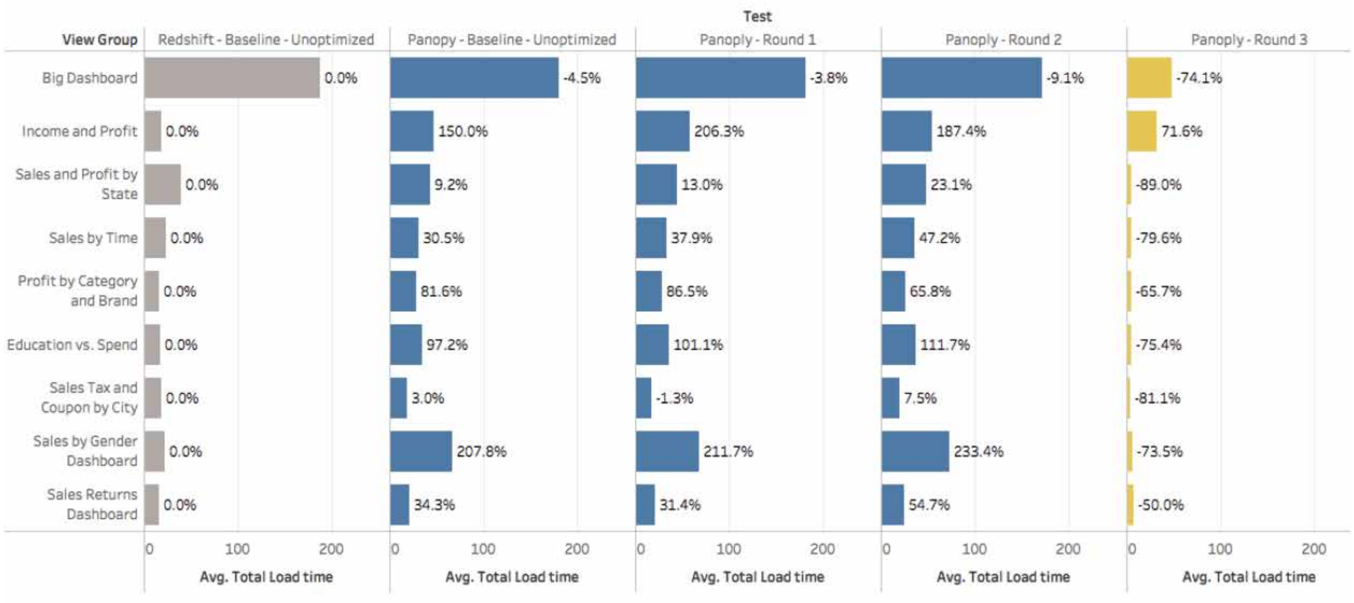
Panoply intelligently decides if queries are completely cached in memory, segregated or if an aggregated query is saved. In case of a segregated query, Panoply enabled Tableau to pull it from memory instead of pulling the data entirely from the data warehouse, which allows for a significantly faster return. This decision made a notable difference in query optimization. Aggregate results were materialized using the following methods:

- **Materialize Views** – The results of the view are calculated and saved as tables (these are refreshed every time new data enters the tables the view depends upon).

- **Materialize Queries (Cached Results)** – Panoply re-calculates frequently used queries and saves the result as a table. By using Panoply's proxy, whenever the user is querying the same query they can identify it and query the materialized result automatically.

As datasets continue to grow and segments of data are repeated, optimization becomes much more important. Left unoptimized, growing datasets would cause performance slowdowns and impede data visualizations in Tableau. Panoply's platform-based optimizations helps users avoid this issue.

Figure 1: Dashboard Load Time - 180M Rows



The Results

Tableau Comparison – Dashboard 180M Rows

As shown in Figure 1 below, Panoply progressively improved in comparison to an unoptimized Redshift. The original normalized schema was used, so all the complex joins were being handled directly by Panoply. The organization of the tables, columns and associated attributes were being managed within Panoply while eliminating the need for data redundancy and increasing data integrity.

Round 1

Optimizations consisted of data compressions, showing a decrease in dashboard runtime and adjusting the Panoply performance to that of the AWS performance. Compression was an effective method of reducing the size of the stored data. By reducing the size of the data, the query performance had a noted increase due to reductions in disk I/O.

Round 2

Optimizations consisted of changing distribution style keys and setting sort keys and constraints, producing another decrease in dashboard runtimes. Panoply automated distribution style by systematically determining the best method to dispense rows of data based on the numbers of joins and aggregations between rows of data. It ensured the most effective constraints were implemented – ones that did not impede overall performance by becoming too burdensome.

Round 3

Optimizations consisted of query caching showing a larger decrease in dashboard runtimes. Panoply intelligently decided if queries were completely cached in memory, segregated or aggregated. In the case of a segregated query, instead of pulling the data entirely from the data warehouse, Panoply enabled Tableau to pull it from memory, which is significantly faster. This decision made a noticeable difference in optimization.

Each dashboard presented a unique set of queries to be executed against each data warehouse. Redshift execution time averaged around twenty seconds. Panoply ranged from thirteen seconds to as high as eighty seconds. The largest decrease of 89% is shown in the Sales and Profit by State dashboard. When comparing all the results against a non-optimized vanilla Redshift in Dashboard 180M, Panoply's average query execution runtime was 45% lower than the Redshift baseline. Comparing the runtime of the "Big Dashboard", Panoply achieved a decrease of 74% in its final query execution run.

Figure 2: Dashboard Load Time - 380M Rows

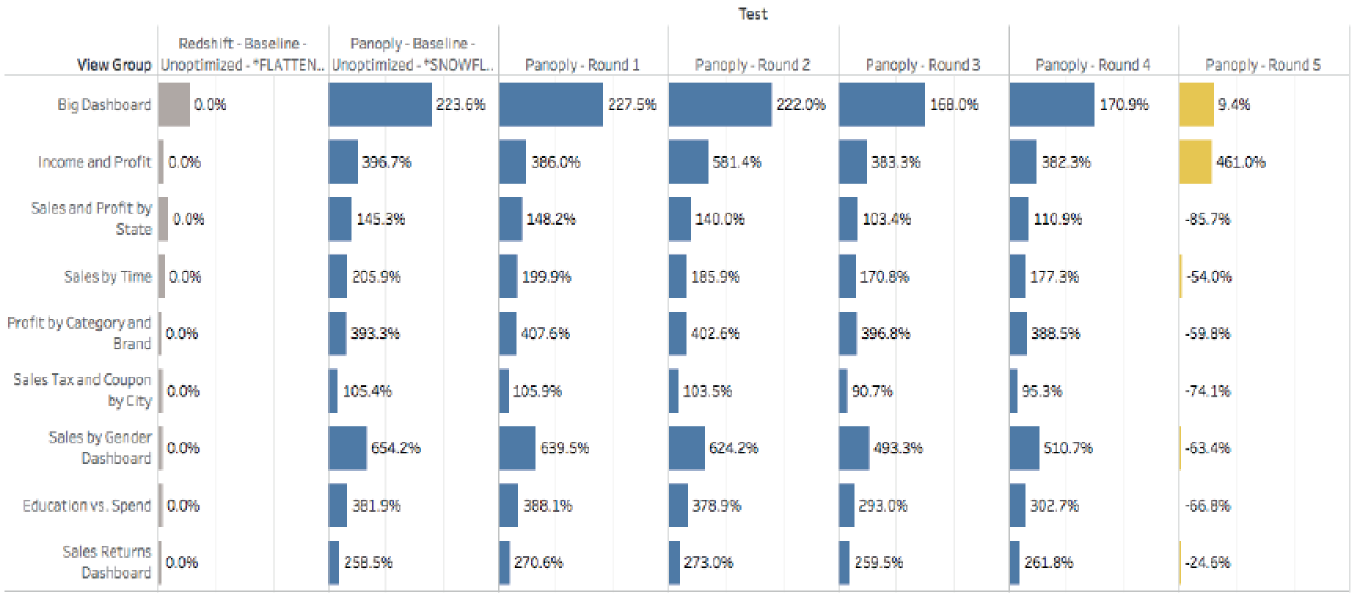
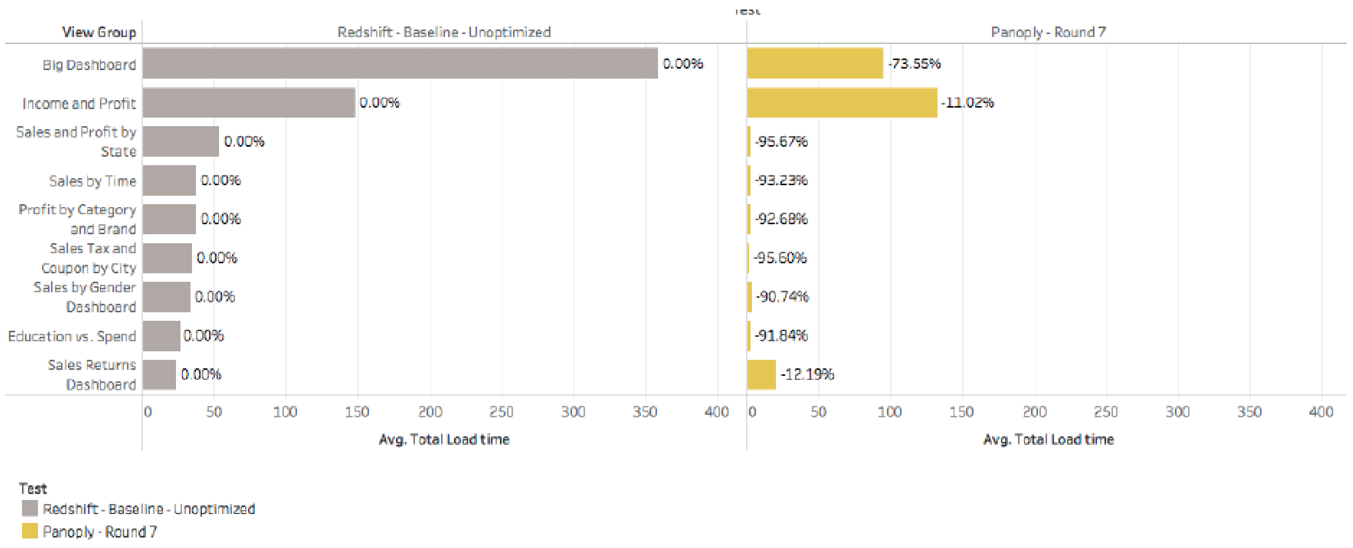


Tableau Comparison – Dashboard 380M Rows

The design of the schema played an important role in what happened next. Consider the results of the second (380M row) run. In this test, Redshift was optimized by using a denormalized schema to minimize joins. Redshift was given an "advantage" of using a denormalized schema while Panoply was "burdened" with an unoptimized complex multidimensional schema. In some cases, Panoply was still faster than (unoptimized) Redshift, but in others, Redshift was faster. The design of the schema contributed to the results. The information that Tableau collected was based on query results and execution for each dashboard.

Figure 2 above reflects the challenges of the optimization using a complex schema. Database design plays a significant role in this process. Because of the more complex schema (and therefore additional joins), Panoply's first baseline runs are two to four times slower. However, by the end of the tests, and despite facing a more challenging database design

Figure 3: Dashboard Load Time - 3B Rows



than Redshift, Panoply rendered visualizations two to three times faster due to its ability to learn the data and optimize based on its own platform-based algorithms.

Three billion rows of data was a substantial amount of data to process, and there were table structure and data layout variables that impacted load time. Tableau took some time to render 360k rows, with rendering performance dependent largely on the complexity of the data and the simulated lack of sophistication by the end user. As mentioned previously, the test was designed to mimic an end user situation where the operator was not experienced in building complex queries or filters and instead relies solely on the tool to complete the task. Most users will have filters in place that would return far fewer than 360k rows of data, decreasing both the query and rendering times. Although the impact of the large dataset on Tableau's rendering time is a contributing factor, Panoply still exceeded Redshift as shown in the Big Dashboard.

Income and Profit query improvements were small, at eleven percent faster than Redshift. Results displayed within the chart show progressive improvement in comparison to the unoptimized baseline of Redshift. The Sales and Profit by State query showed the most significant decrease in load time at 95.6%. The overall performance in Figure 3 above averages at 65% in comparison to the baseline of Redshift.

What We Learned

A Panoply cluster in its raw form can start off with performance metrics similar to a unoptimized Redshift cluster. Gradually, however, the proprietary Panoply algorithms begin to learn the data and act upon the information derived from the queries. As these processes run and the Panoply platform continues to learn, the result is continuous performance improvement. After two to three weeks of running dashboards, Panoply reached an optimal runtime that outperformed the unoptimized Redshift cluster by up to 90%.

The conclusions we draw from viewing these results is that data design, data configuration, and data management tool selection can have a marked impact on visualization rendering performance in Tableau. Panoply's platform-based machine learning algorithms eliminate the difficult IT work of optimization and configuration, streamlining data analysis for analysts and data scientists and providing a faster, more agile database response for all Tableau users.