



ATSCALE

Cloud Analytics Performance Benchmark: Google BigQuery

By David Mariani & Krasimir Kovachki
2021-Q3




TABLE OF CONTENTS

Executive Summary	1
Introduction	2
Leveraging Google BigQuery for BI and Analytics	2
The Power of AtScale and Google BigQuery	3
Benchmark Methodology	4
Benchmark Measurements	4
Benchmark Dataset	5
Benchmark Queries	6
Test Harness	6
Configuration Tested	7
A Special Note on BI Engine	7
Query Performance for a Single User Test Methodology	8
Query Performance with Concurrency Test Methodology	8
Compute Cost Calculations	8
Summary Results	9
Query Performance Test Results	10
Concurrent Query Performance	11
Median Query Time by TPC-DS Query Test Results	13
Compute Cost Test Results	14
Complexity Test Results	15
Conclusion	19

Executive Summary

This benchmarking study was conducted to quantify the benefits of using the AtScale semantic layer platform with the Google BigQuery data platform to manage BI and analytics workloads. The comparative analysis was based on four defined measurements: Query Performance, Concurrent Query Performance, Compute Cost, and SQL Complexity. Using the standard TPC-DS (10TB) benchmarking framework, measurements were taken for raw Google BigQuery and for AtScale on Google BigQuery that showed the clear advantages for combining AtScale with Google BigQuery to accelerate and optimize BI and analytics programs.

Test	Improvement Factor with AtScale Google BigQuery
Query Performance ¹	4x Faster
Concurrent Query Performance ²	11x Faster
Compute Cost ³	2.7x Cheaper
Complexity ⁴	76% less complex SQL queries

Figure 1: Improvements with AtScale

This analysis is a refresh of a study first done in 2020 using the same methodology. The results illustrate improvement in Google BigQuery's raw performance, but with clear benefits for the combined solution

¹ Elapsed time for executing 1 query five times

² Elapsed time executing 1 (x5), 5, 25, 50 queries

³ Compute costs for cluster time for user concurrency test

⁴ Complexity score for SQL queries for number of: functions, operations, tables, objects & subqueries (AtScale = 258, TPC-DS = 1,057)

Introduction

The enterprise has entered into a new era of data warehousing. Driven by the increasing popularity of the public cloud, new cloud-based data platforms have become the dominant choice for enterprises managing their data. By offering customers the power of a relational, scale-out data platform without the overhead of managing it, cloud data platforms promise to make more data available at a lower cost with fewer data management headaches.

Leveraging Google BigQuery for BI and Analytics

Google BigQuery is a great choice of cloud data platform for a number of reasons. First, Google BigQuery is based on a serverless architecture, freeing customers from needing to size and manage compute clusters to scale workloads. This dramatically simplifies the life of customers since Google manages scale behind the scenes, autonomously. Second, Google offers a robust set of developer tools for running queries and loading data easily using JSON and CSV formats. Next, Google BigQuery is the best of the cloud data platform options for performing fast table scans on very large tables.

Google BigQuery is based on a serverless architecture, freeing customers from needing to size and manage compute clusters to scale workloads.

Finally, Google BigQuery has integrated AI/ML features directly into their SQL support making it easy to build, train and run machine learning models directly in the database

The Power of AtScale and Google BigQuery

While cloud data platforms reduce the maintenance cost and scaling headaches of managing data infrastructure for IT, they don't make data any easier to understand or access for analytics consumers, nor do they help IT better predict and control cloud costs. The AtScale platform works natively with cloud data platforms to deliver an analytics semantic layer for business intelligence (BI) and data science teams.

The AtScale semantic layer provides the following benefits:

1. It presents a **consistent** set of **business-friendly metrics** for BI and data science teams to consume data with the **tools of their choice**.
2. It provides an integration layer to support analytics **discoverability, governance,** and **security**.
3. It accelerates end-to-end query performance while **optimizing** data platform resources and **costs**.

By leveraging a graph-based semantic model, the AtScale platform sends queries to Google BigQuery using its data virtualization engine and pushes workloads to the Google BigQuery platform. By automatically creating and managing aggregate tables on Google BigQuery based on user query patterns, AtScale avoids costly atomic table scans and delivers superior query performance by re-writing queries to access those aggregate tables.

In this study, we will compare the performance, complexity and costs of these cloud data platforms with and without the AtScale platform.

Benchmarking Methodology

Benchmark Measurements

This benchmark uses four key metrics to compare Google BigQuery to Google BigQuery + AtScale. The metrics are designed to answer basic questions relevant to enterprise analytics leaders.

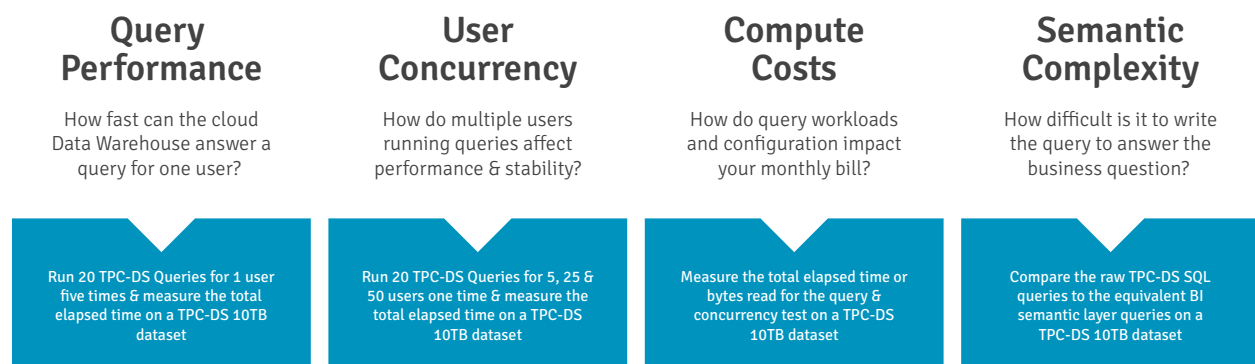


Figure 2: Benchmark Testing Topics

By automatically creating and managing aggregate tables on Google BigQuery based on user query patterns, AtScale avoids costly atomic table scans and delivers superior query performance by re-writing queries to access those aggregate tables.

Benchmark Dataset

We used the TPC-DS benchmark v2.11.0 from the Transaction Processing Council (TPC) for our tests. We chose the 10TB (scale factor 10,000) version for this benchmark to better measure scalability limits of each platform and to simulate a typical enterprise workload. This version's largest fact table (store_sales) at 28+ billion rows and the largest dimension (customer) at 65 million rows is a significant scale challenge for most data platforms. In addition, the TPC-DS benchmark is ubiquitous amongst the database warehouse vendors and we felt it represented a reasonable real-life analytics schema and set of queries.

Table Name	Row Size	Row Count
call_center	305	54
catalog_page	139	40,000
catalog_returns	166	1,440,033,112
catalog_sales	226	14,399,964,710
customer	132	65,000,000
customer_address	110	32,500,000
customer_demographics	42	1,920,800
date_dim	141	73,049
household_demographics	21	7,200
income_band	16	20
inventory	16	1,311,525,000
item	281	402,000
promotions	124	2,000
reason	38	70
ship_mode	56	20
store	263	1,500
store_returns	134	2,879,970,104
store_sales	164	28,799,983,563
time_dim	59	86,400
warehouse	117	25
web_page	96	4,002
web_returns	162	720,020,485
web_sales	226	7,199,963,324
web_site	292	78

THE TPC-DS 10TB DATASET HAS:

1

Multiple fact tables

2

Large fact tables

3

Large dimensions

Figure 3: TPC-DS 10TB Table Sizes

Benchmark Queries

We selected a representative set of 20 queries from the 99 TPC-DS queries set to keep the run time and costs of running the benchmarks within reason without having to downsize data size. The queries were chosen in no particular order and were selected to eliminate redundancy and to ensure the usage of most tables. It was imperative to benchmark the cloud data warehouse vendors with the largest data we could afford and test to reveal real-life differences in the respective platforms.

The following 20 TPC-DS queries were selected for the test:

TPC-DS Query #	Description	TPC-DS Query #	Description
2	Report the ratios of weekly web and catalog sales increases from one year to the next year for each week. That is, compute the increase of Monday, Tuesday, ... Sunday sales from one year to the following.	52	Report the total of extended sales price for all items of a specific brand in a specific year and month.
7	Compute the average quantity, list price, discount, and sales price for promotional items sold in stores where the promotion is not offered by mail or a special event. Restrict the results to a specific gender, marital and educational status.	53	Find the ID, quarterly sales and yearly sales of those manufacturers who produce items with specific characteristics and whose average monthly sales are larger than 10% of their monthly sales.
13	Calculate the average sales quantity, average sales price, average wholesale cost, total wholesale cost for store sales of different customer types (e.g., based on marital status, education status) including their household demographics, sales price and different combinations of state and sales profit for a given year.	55	For a given year, month and store manager calculate the total store sales of any combination all brands.
15	Report the total catalog sales for customers in selected geographical regions or who made large purchases for a given year and quarter.	56	Compute the monthly sales amount for a specific month in a specific year, for items with three specific colors across all sales channels. Only consider sales of customers residing in a specific time zone. Group sales by item and sort output by sales amount.
26	Computes the average quantity, list price, discount, sales price for promotional items sold through the catalog channel where the promotion was not offered by mail or in an event for given gender, marital status and educational status.	60	What is the monthly sales amount for a specific month in a specific year, for items in a specific category, purchased by customers residing in a specific time zone. Group sales by item and sort output by sales amount.
31	List counties where the percentage growth in web sales is consistently higher compared to the percentage growth in store sales in the first three consecutive quarters for a given year.	61	Find the ratio of items sold with and without promotions in a given month and year. Only items in certain categories sold to customers living in a specific time zone are considered.
33	What is the monthly sales figure based on extended price for a specific month in a specific year, for manufacturers in a specific category in a given time zone. Group sales by manufacturer identifier and sort output by sales amount, by channel, and give Total sales.	71	Select the top revenue generating products, sold during breakfast or dinner time for one month managed by a given manager across all three sales channels.
42	For each item and a specific year and month calculate the sum of the extended sales price of store transactions.	88	How many items do we sell between pacific times of a day in certain stores to customers with one dependent count and 2 or less vehicles registered or 2 dependents with 4 or fewer vehicles registered or 3 dependents and five or less vehicles registered. In one row break the counts into sells from 8:30 to 9, 9 to 9:30, 9:30 to 10 ... 12 to 12:30
48	Calculate the total sales by different types of customers (e.g., based on marital status, education status), sales price and different combinations of state and sales profit.	96	Compute a count of sales from a named store to customers with a given number of dependents made in a specified half hour period of the day.
50	For each store count the number of items in a specified month that were returned after 30, 60, 90, 120 and more than 120 days from the day of purchase.	98	Report on items sold in a given 30 day period, belonging to the specified category.

Figure 4: TPC-DS Test Queries

Test Harness

To ensure consistency for concurrency tests, we ran queries using v5.4.1 of [Apache JMeter](#). The instructions, documentation, utility scripts, results, and JMeter JMX files can be found in our GitHub repository and are available upon [request](#).

We designed the JMeter test suites to run the above 20 queries in the following four configurations:

- ▲ **1 concurrent user, 5 loops**
(averaging the result to even out cold starts)
- ▲ **25 concurrent users, 1 loop**
- ▲ **5 concurrent users, 1 loop**
- ▲ **50 concurrent users, 1 loop**

Configuration Tested

The following Snowflake configuration was used for the test:

Vendor	Configuration	Compute Cost per Hour ⁵
Google BigQuery	Monthly Fixed Rate Pricing (\$40,000 per month for 2,000 slots)	\$55.56

Figure 5: Data Platform Configurations

For the test, we used Google BigQuery’s “out of the box” configuration. We did not manually tune any of the TPC-DS queries and used the same clustering scheme for the TPC-DS tables as Snowflake’s, as defined in Snowflake’s sample TPC-DS 10TB dataset.

A Special Note on BI Engine

Google BigQuery offers a companion query acceleration service called “BI Engine”. At the time of this benchmark, BI Engine was in preview. We ran the Google BigQuery tests with and without BI Engine and found that BI Engine did not provide any measurable benefit in either scenario (Google Big Query raw, AtScale on Google BigQuery). We reported our results to the Google BigQuery team and they are researching our results. Once BI Engine is generally available, we will update these results to quantify the benefits of BI Engine for these benchmarks.

⁵Storage cost wasn’t factored in (only compute cost)

Query Performance for a Single User Test Methodology

To test raw query performance, we ran the 20 TPC-DS queries with one concurrent user five times and calculated the average elapsed time to finish each query. The elapsed time is simply the difference between the start and end time of the test as reported by JMeter. We disabled Google BigQuery's query caching for this test.

Query Performance with Concurrency Test Methodology

To test how each data warehouse performs with different levels of user concurrency, we ran each of the 20 TPC-DS queries with 1, 5, 25 and 50 concurrent users using JMeter. We added a 750ms sleep between each query start and using a single connection pool that was sized according to the number of threads for the test. We used 1 loop (iteration) for the 5, 25, and 50 thread test and 5 loops for the 1 thread test. The elapsed time is simply the difference between the start and end time of each thread test as reported by JMeter. We disabled Google BigQuery's query caching for this test.

Compute Cost Calculations

Google BigQuery has a few different pricing plans, fixed rate and on-demand. For this benchmark, we calculated costs using the fixed pricing plan in order to better align with the other platforms' time-based pricing plans.

We calculated the compute costs by multiplying the total end-to-end run time as reported by JMeter for the concurrency test by the cluster compute cost per hour like so:

$$\text{ConcurrencyRunTimeMinutes} / 60 * \text{ComputeCostPerHour}$$

We explicitly excluded storage costs from our calculations. We found that storage cost was nominal across all platforms and given that it's a fixed cost, it was not subject to variation in our testing scenarios.

Summary Results

We also ran the same 20 TPC-DS queries through the AtScale platform for Google BigQuery. AtScale's Acceleration Structures showed major benefits in accelerating query performance, improving user concurrency and reducing compute costs. AtScale's semantic layer also drastically reduced the complexity of the TPC-DS queries by hiding the joins and calculations from consumers. The illustration below shows the extent of the benefits AtScale provides on top of the Google BigQuery data warehouse:

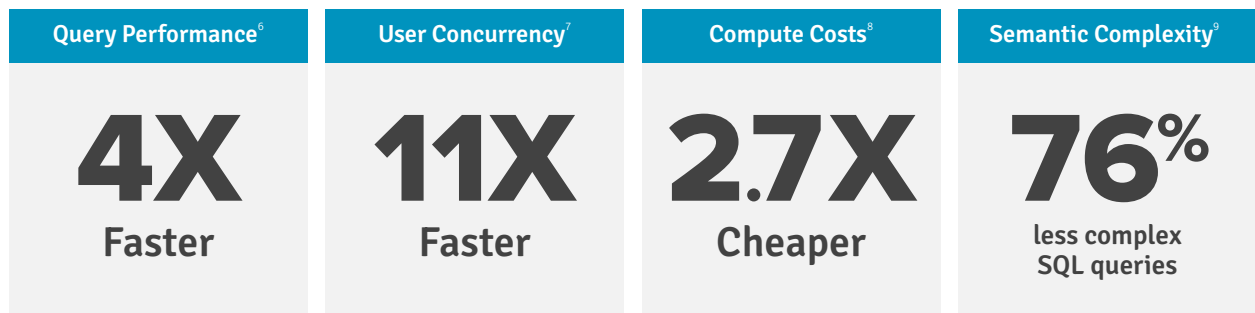


Figure 6: Improvements with AtScale

⁶ Elapsed time for executing 1 query five times

⁷ Elapsed time executing 1 (x5), 5, 25, 50 queries

⁸ Compute costs for cluster time for user concurrency test

⁹ Complexity score for SQL queries for number of: functions, operations, tables, objects & subqueries (AtScale = 258, TPC-DS = 1,057)

Query Performance Test Results

For the query performance test, we ran our 20 TPC-DS queries 5 times each using JMeter with a single thread. Even at a single concurrent user, we saw orders of magnitude improvement using AtScale on the Google BigQuery data warehouse in this test.

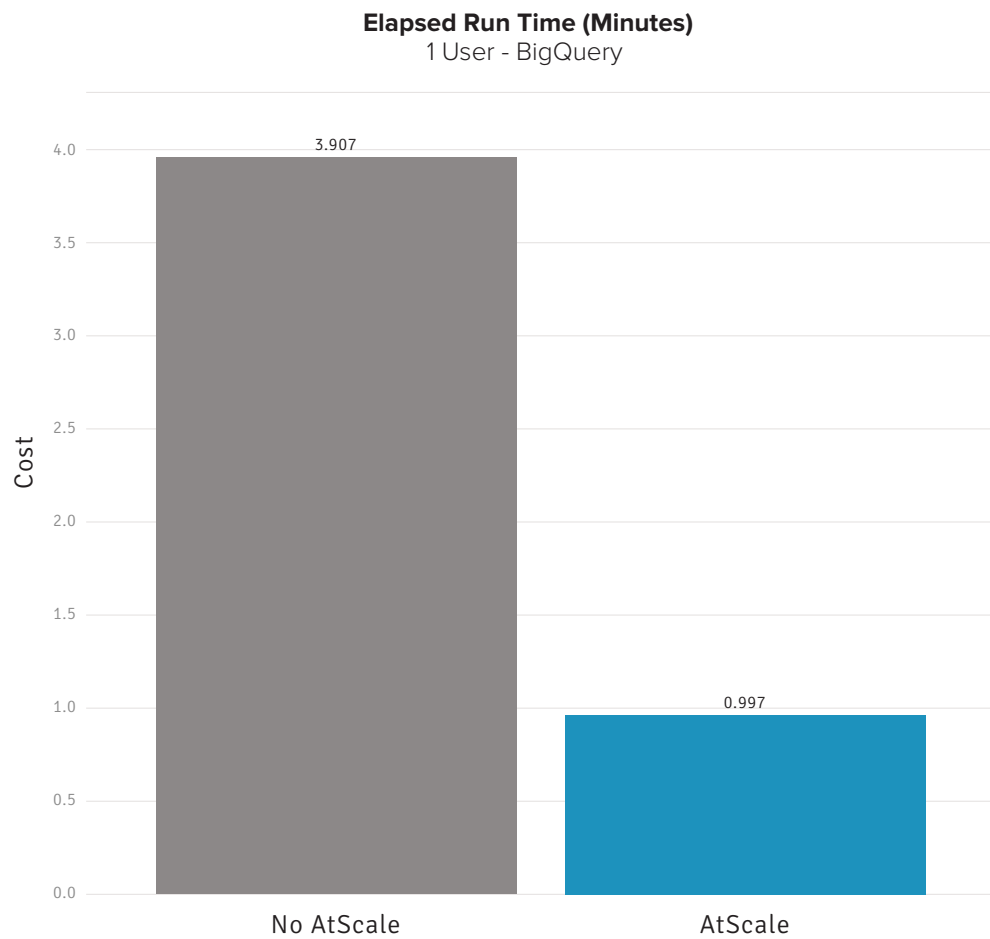


Figure 7: Elapsed Run Time for 1 Thread

Concurrent Query Performance

For the user concurrency test, we ran consecutive JMeter suites configured to execute 1, 5, 25, and 50 queries at the same time to simulate user concurrency. Each test ran 1 iteration with the exception of the 1 thread test which ran 5 iterations sequentially.

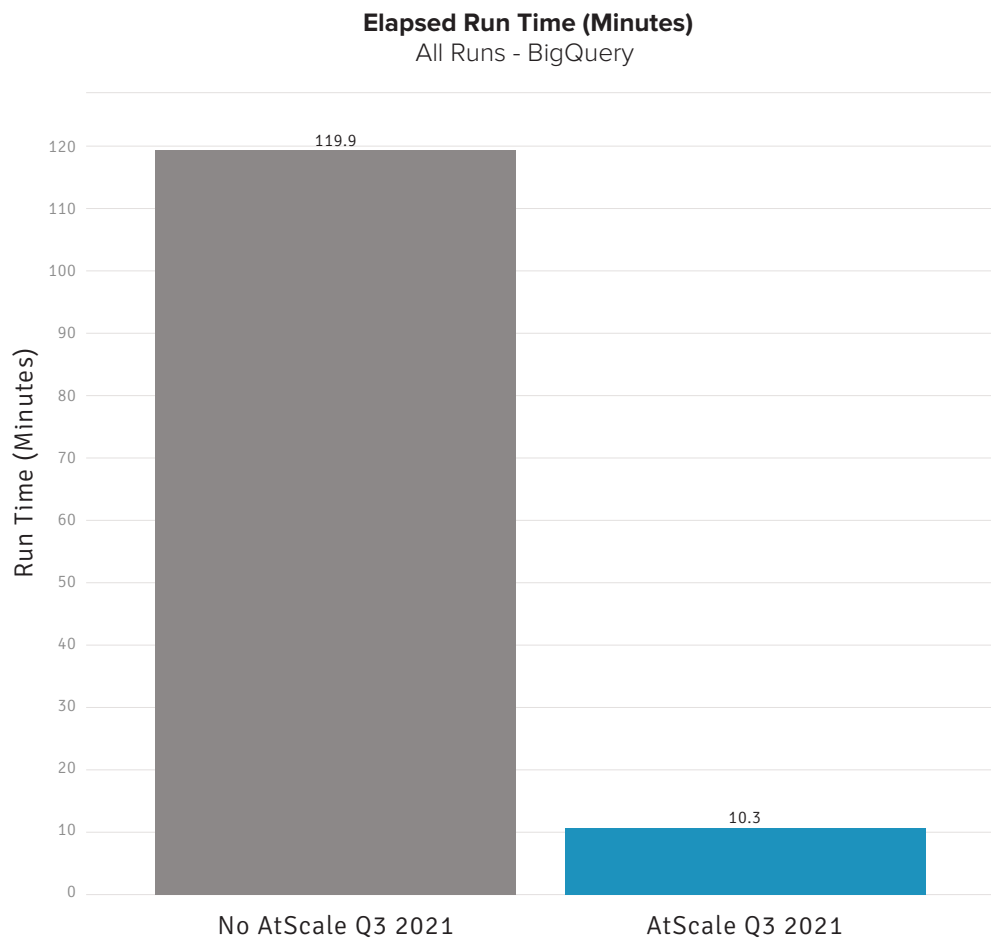


Figure 8: Elapsed Run Time for All Runs

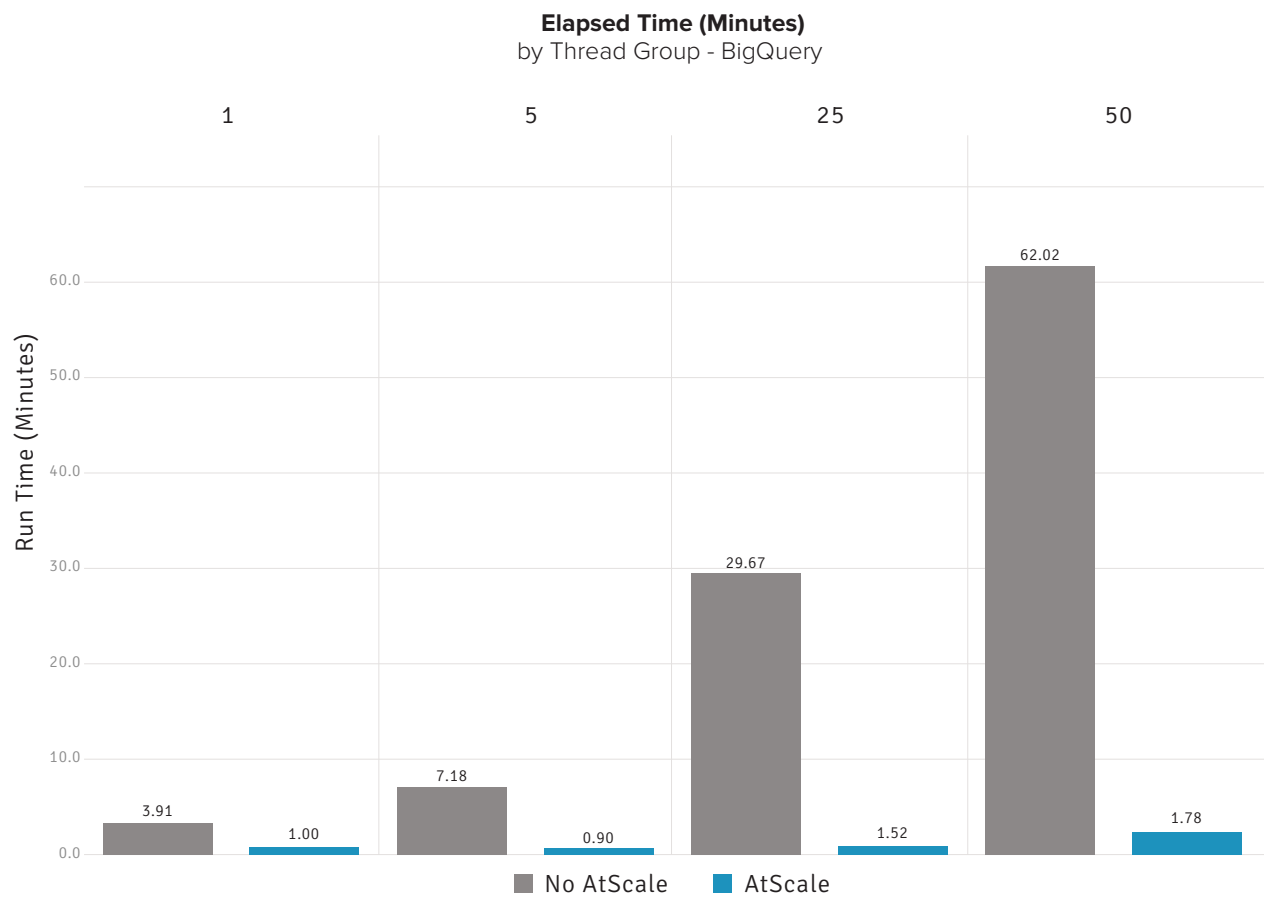


Figure 9: Elapsed Run Time by Thread

Median Query Time by TPC-DS Query Test Results

The following chart (logarithmic scale) illustrates the benefits of AtScale for each of the 20 TPC-DS queries (by TPC-DS Query number) tested with a median reference line overlay for comparison. This is the median elapsed query time for all runs (1, 5, 25, 50 concurrent users) so data platform load is taken into account. Notice that for Google BigQuery raw (without AtScale), the median query time is **almost 1 minute** versus Google BigQuery on AtScale at a median time of **1.7 seconds**. For interactive business intelligence, elapsed query times over 10 seconds are not typically not acceptable by users which may force IT to use data extracts or external caching solutions instead.

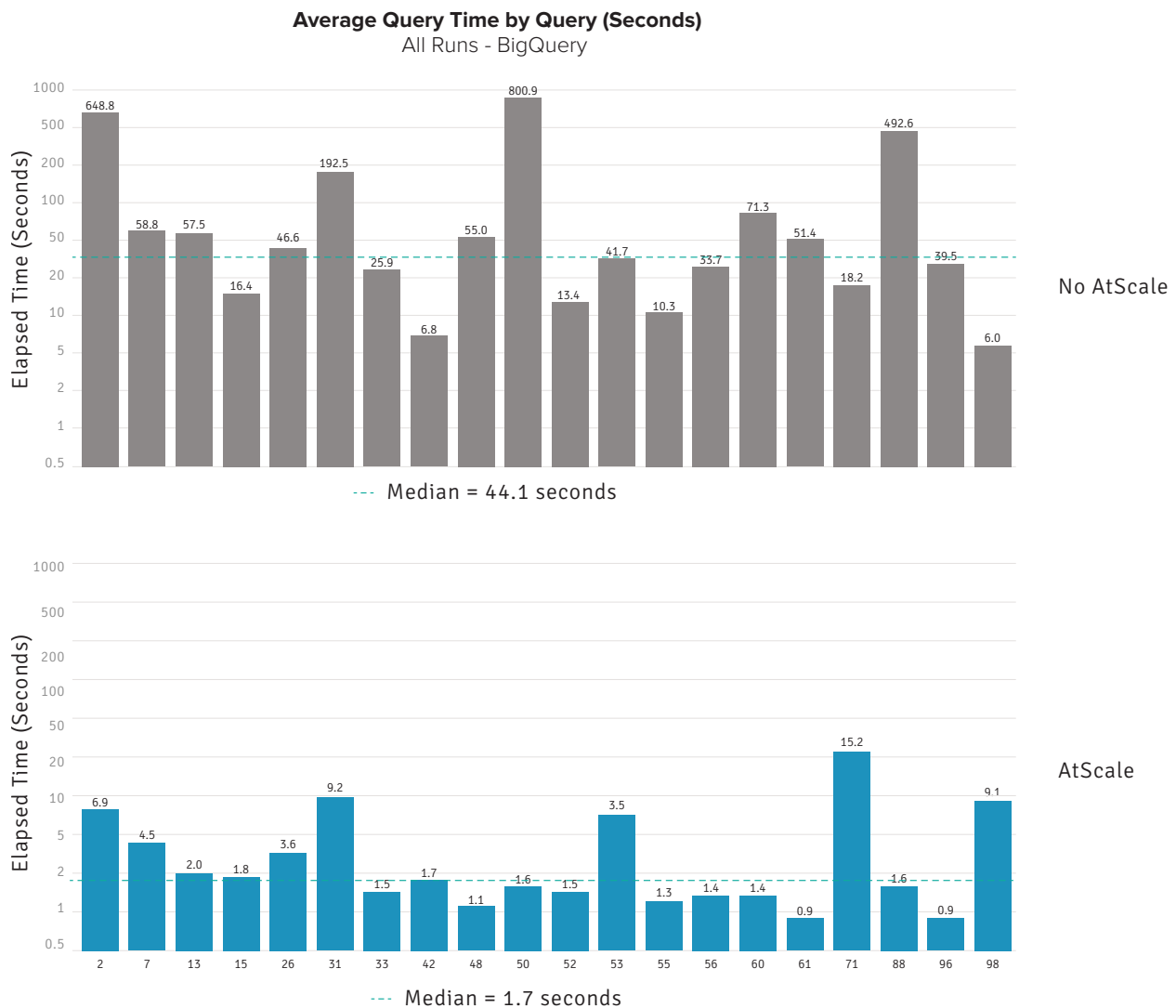


Figure 10: Average query time by TPC-DS query number with median

Compute Cost Test Results

You will also see the value that AtScale can bring to cost predictability. By minimizing the amount of data scanned, AtScale takes less time to run queries, with fewer resources used, which means more users can run queries at the same time (higher concurrency) without additional hardware or resources.

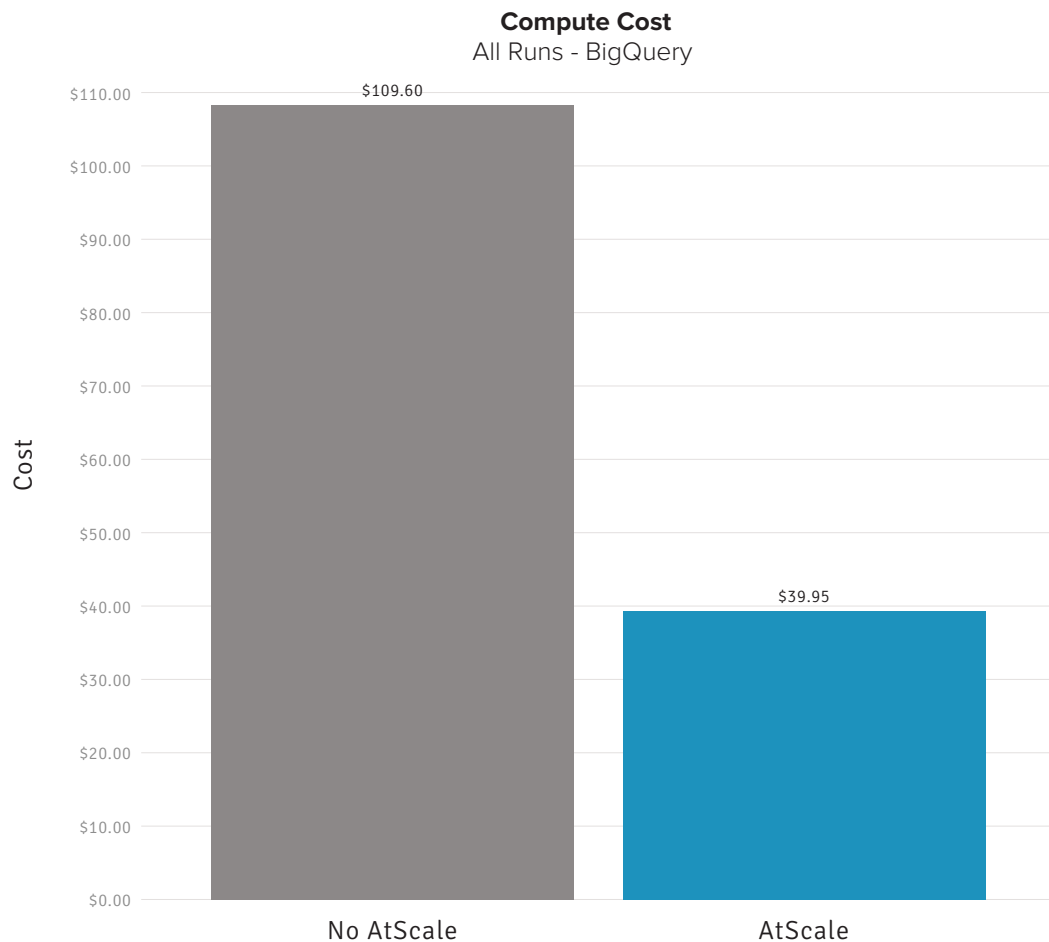


Figure 11: Compute Costs for All Thread Groups

You will notice that AtScale reduces costs of running the query test on Google BigQuery by almost 3 times for the fixed price (time-based) pricing model. With Google BigQuery's [on-demand pricing model](#) that charges by the amount of data scanned, AtScale delivers even better cost savings than reported here.

Complexity Test Results

The TPC-DS benchmark provides a good illustration of just how hard it can be to write SQL to answer a simple business question. Translating tables and star schemas into business logic is not an easy task. With today's BI tools, our business users are spending more and more time dealing with data engineering tasks rather than getting answers to their business questions.

For example, with query #60 of the TPC-DS benchmark, the business question is fairly straightforward but the SQL to express it is not.

BUSINESS QUESTION:

What is the monthly sales amount for a specific month in a specific year, for items in a specific category, purchased by customers residing in a specific time zone?

SQL TO ANSWER BUSINESS QUESTION:

TPC-DS Raw	
<pre> with ss as (select i_item_id,sum(ss_ext_sales_price) total_sales from store_sales, date_dim, customer_address, item where i_item_id in (select i_item_id from item where i_category in ('Jewelry')) and ss_item_sk = i_item_sk and ss_sold_date_sk = d_date_sk and d_year = 1999 and d_moy = 9 and ss_addr_sk = ca_address_sk and ca_gmt_offset = -6 group by i_item_id), cs as (select i_item_id,sum(cs_ext_sales_price) total_sales from catalog_sales, date_dim, customer_address, item where i_item_id in (select i_item_id from item where i_category in ('Jewelry')) and cs_item_sk = i_item_sk and cs_sold_date_sk = d_date_sk and d_year = 1999 and d_moy = 9 and cs_bill_addr_sk = ca_address_sk and ca_gmt_offset = -6 group by i_item_id), ws as (select i_item_id,sum(ws_ext_sales_price) total_sales from web_sales, date_dim, customer_address, item where i_item_id in (select i_item_id from item where i_category in ('Jewelry')) and ws_item_sk = i_item_sk and ws_sold_date_sk = d_date_sk and d_year = 1999 and d_moy = 9 and ws_bill_addr_sk = ca_address_sk and ca_gmt_offset = -6 group by i_item_id) select i_item_id, ss.total_sales + cs.total_sales + ws.total_sales total_sales from item, ss, cs, ws where i_item_id = ss.i_item_id and i_item_id = cs.i_item_id and i_item_id = ws.i_item_id </pre>	<pre> item where i_item_id in (select i_item_id from item where i_category in ('Jewelry')) and cs_item_sk = i_item_sk and cs_sold_date_sk = d_date_sk and d_year = 1999 and d_moy = 9 and cs_bill_addr_sk = ca_address_sk and ca_gmt_offset = -6 group by i_item_id), ws as (select i_item_id,sum(ws_ext_sales_price) total_sales from web_sales, date_dim, customer_address, item where i_item_id in (select i_item_id from item where i_category in ('Jewelry')) and ws_item_sk = i_item_sk and ws_sold_date_sk = d_date_sk and d_year = 1999 and d_moy = 9 and ws_bill_addr_sk = ca_address_sk and ca_gmt_offset = -6 group by i_item_id) select i_item_id, ss.total_sales + cs.total_sales + ws.total_sales total_sales from item, ss, cs, ws where i_item_id = ss.i_item_id and i_item_id = cs.i_item_id and i_item_id = ws.i_item_id </pre>

26,640 bytes

Figure 12: TPC-DS Raw SQL to answer question

As you can see, it's not at all obvious what the query is doing and obviously there's a lot of repetition which makes it very prone to error.

In response to this challenge, for this benchmark study, we defined an AtScale model that drastically simplifies user queries by translating the raw tables and schema into a business semantic layer. The following screenshot is the TPC-DS model expressed in AtScale Design Center:

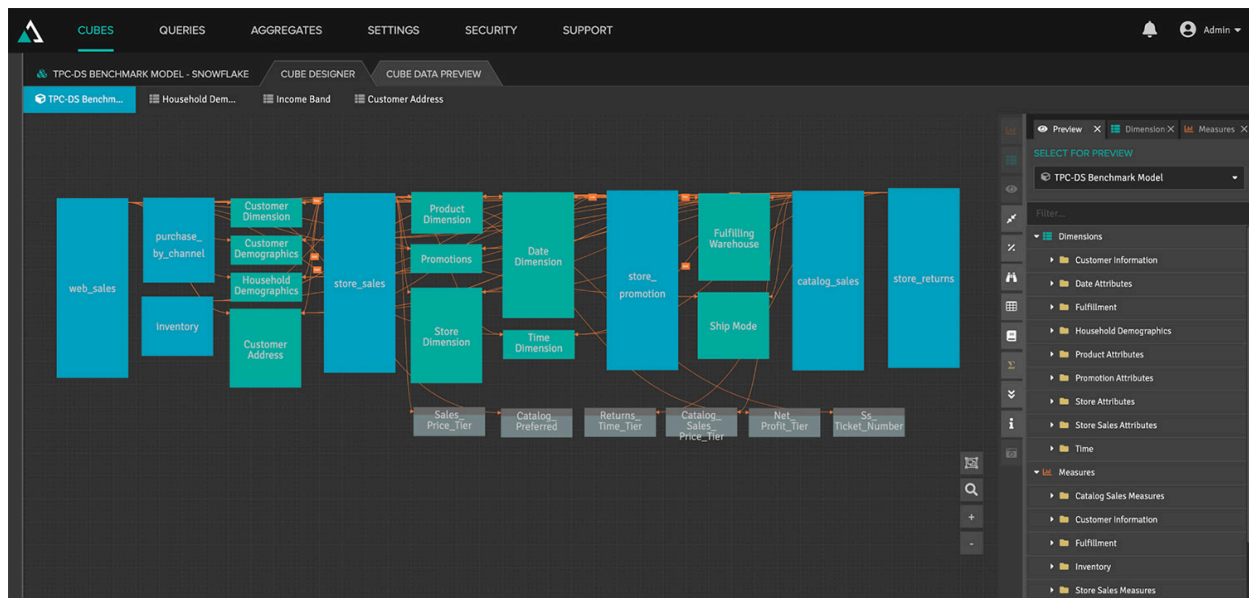


Figure 13: AtScale TPC-DS Data Model

Instead of writing complex SQL or engineering data models in the BI tool, this business question was easily answered with Tableau on AtScale as you can see below:

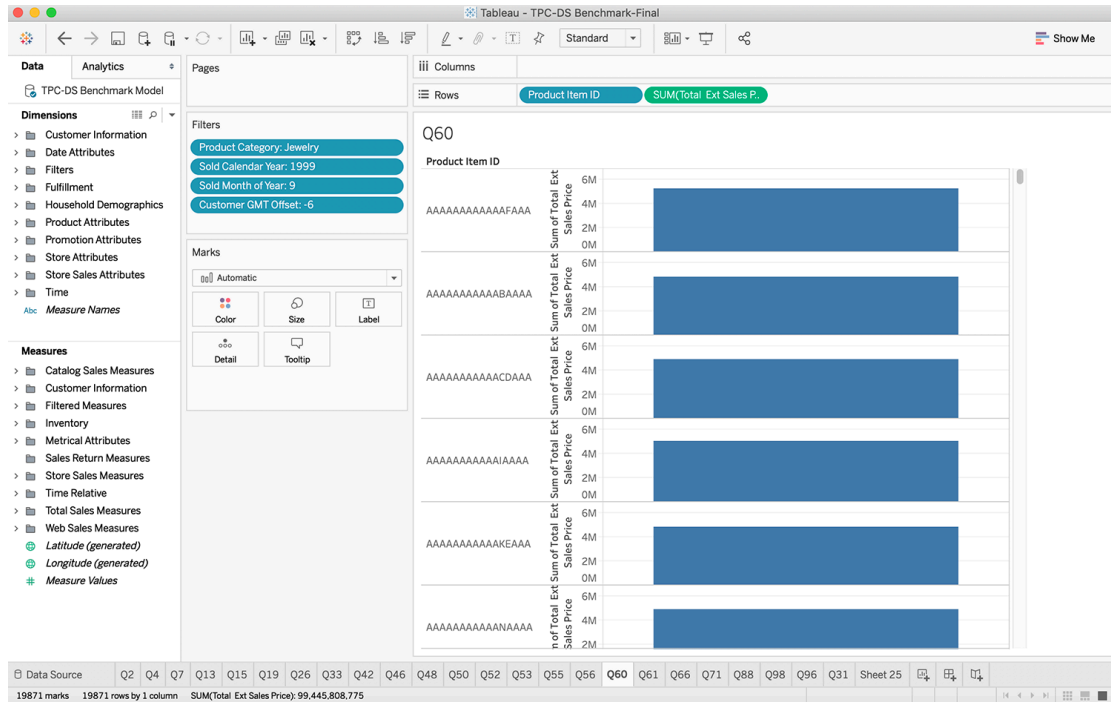


Figure 14: Tableau on AtScale TPC-DS Model for Query #60

The visualization above for TPC-DS query #60 generated the following SQL against AtScale:

AtScale SQL

```

SELECT
  `d_product_item_id` AS `d_product_item_id`,
  SUM(`Total Ext Sales Price`) AS `sum_total__ext_sales_price_ok`
FROM
  `tpc-ds benchmark model - snowflake`.`tpc-ds benchmark model`.`tpc_ds_benchmark_model`
WHERE
  `l Category` = 'Jewelry'
  AND `Sold Calendar Year` = 1999
  AND `Sold d_month_of_year` = 9
  AND `d_customer_gmt_offset` = -6
GROUP BY 1
    
```

18,593 bytes

Figure 15: AtScale SQL to answer question

As you can see, the SQL written against the AtScale semantic model is human readable and understandable. In addition, this semantic model provides important context for query optimization which delivers query acceleration, user concurrency improvements and cost reduction.

As a measure of complexity, we used an open source parser to break down each SQL statement into the following groups:

1. Number of functions used
2. Number of arithmetic operations
3. Number of tables accessed
4. Number of objects used and number of subqueries needed.

Configuration	Complexity Factor					
	# of Functions	# of Operations	# of Tables	# of Objects	# of Subqueries	Total Score
No AtScale	87	66	177	700	27	1,057
AtScale	36	2	21	198	1	258

Figure 16: Complexity score for TPC-DS benchmark with and without AtScale Semantic Layer

Conclusion

As you can see from the benchmark results, the future for data warehousing is definitely in the cloud. The cloud data platforms we tested prove that the cloud is a viable alternative with many performance and management advantages for data warehousing compared to the traditional on-premise options. However, there are key differences in performance, scalability and cost that need to be considered.

We also proved that the inclusion of a semantic layer like AtScale's can make the cloud data warehouses even better by:

- 1.**
Drastically simplifying queries for users
- 2.**
Insuring all users access the same, secure data
- 3.**
Increasing query performance by up to 4x
- 4.**
Improving user concurrency by up to 11x
- 5.**
Reducing cost by up to 2.7x

ABOUT ATSCALE

AtScale enables smarter decision-making by accelerating the flow of data-driven insights. The company's semantic layer platform simplifies, accelerates, and extends business intelligence and data science capabilities for enterprise customers across all industries.

ATSCALE

atscale.com