Cloud Analytics Database Performance Report

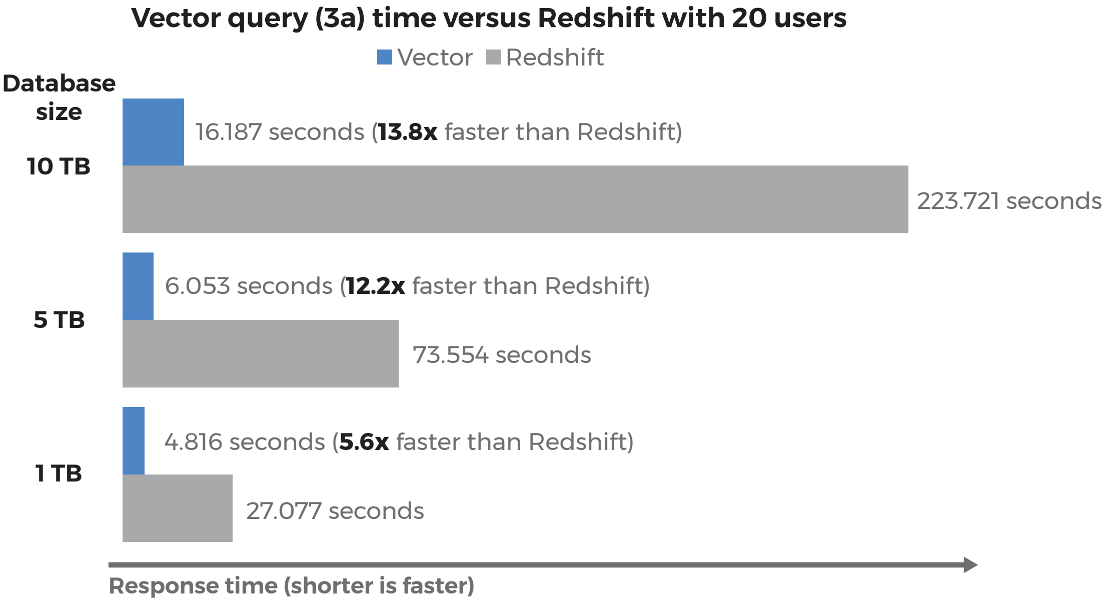
Actian Vector up to 14x faster than Amazon Redshift

MCG Global Services Benchmark Results

ACTIAN™

Key insights

- Independent benchmark performed by MCG Global Services
- Test based on Berkeley AMPlab Big Data benchmark workload
- Tested on 5-node AWS cluster at 1, 5 and 10 TB database sizes
- 10 TB database includes a 58 Billion row table used in join
- Performance advantages were most pronounced on complex joins and as data set size grew
- The complex 10 TB join took **Actian Vector under 17 seconds to run** while **Amazon Redshift** took close to 4 minutes



Checkout Actian Vector's performance advantage today!

Activate for free in the AWS or Azure cloud or download our free community edition at www.actian.com/vce

Cloud Database Performance Benchmark

Product Profile and Evaluation:
Actian Vector and Amazon Redshift

By William McKnight and Jake Dolezal

March 2018

Sponsored by

ACTIAN™

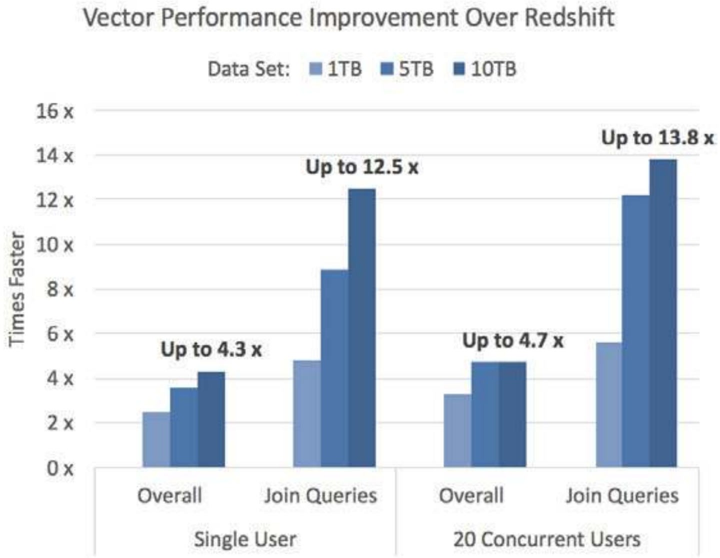
Executive Overview

Data-driven organizations rely on analytic databases to load, store, and analyze volumes of data at high speed to derive timely insights. Data volumes within modern organization’s information ecosystems are rapidly expanding—placing significant performance demands on legacy architectures. Today, to fully harness their data to gain competitive advantage, businesses need modern scalable architectures and high levels of performance and reliability to provide timely analytical insights.

To address this need, we conducted this benchmark study, which focuses on the performance of cloud-enabled[1], enterprise-ready, relationally-based, analytical-workload solutions from Actian Vector and Amazon Redshift. The intent of the benchmark’s design was to simulate a set of basic scenarios to answer fundamental business questions that an organization from nearly any industry sector might encounter and ask.

The benchmark tested the scalability of corporate-complex workloads—independently—in terms of data volumes. The tests were based on the enterprise-representative UC Berkeley AMPLab Big Data Benchmark with the dataset sizes being extended to 1, 5, and 10 TB of data to simulate real world Big Data demands. The testing was conducted using comparable hardware configurations on Amazon Web Services (AWS) EC2 Instances, deployed into an AWS Virtual Private Cloud (VPC) within the same Placement Group.

Overall, the benchmark results were insightful in revealing the query execution performance of Actian Vector and Redshift scaling up by data volume and concurrency—revealing some of the performance differentiators in the two products. The most eye-popping finding was executing the benchmark queries containing joins, with Actian Vector performing 12.5 (single user) to nearly 14 times (20 concurrent users) faster than Redshift.



In our experience, performance is the most important aspect of a database selection, but it is only one aspect and many factors should be considered.

[1] We took the cloud deployment as a given and did not compare the on-premise version of Actian Vector. Of course, Redshift is only available in the cloud on AWS.

Big Data Analytics Platform Offerings

Big Data analytics platforms load, store, and analyze volumes of data at high speed, providing timely insights to businesses. This data is structured, semi-structured, or unstructured from a variety of sources, namely machine, sensor, log, sentiment, clickstream, and geo-spatial data as examples. These analytics-driven businesses leverage this data, for example, for performing clickstream analysis to market new promotions, operational analytics to drive efficiency, and predictive analytics to evaluate credit risk and detect fraud. Often organizations leverage a mix of relational analytical databases and data warehouses, Apache Hadoop, and NoSQL databases to gain the analytic insights they desire to optimize their business performance.

This paper focuses on relational analytical databases in the cloud as deployments in the cloud are at an all-time high and poised to expand dramatically. The cloud offers opportunities to differentiate and innovate with these database systems at a much more rapid pace than ever before possible. Further, the cloud has been a disruptive technology, as cloud storage tends to cost less, enables more rapid server deployment and application development, and offers elastic scalability vis-a-vis on-premise deployments. For these reasons, and others, many data-driven companies are increasingly migrating to the cloud to maintain or gain momentum as a company. This paper focuses on benchmarking Actian Vector and Amazon Redshift, two relational analytical databases based on massively parallel processing (MPP) and columnar based database architectures that scale and provide high-speed analytics. It should be noted while the benchmark measures cloud-based performance of both offerings, Vector, unlike Redshift, is also available as an on-premise offering. In additional Vector is available for developers as a free on-premise community edition as download and in the AWS marketplace with single-click support.

About the Platforms

	Actian Vector	*Redshift*
Company	Actian	Amazon
Released	2014	2014
Current Version	5.0	1.0.1583
Storage	Hadoop HDFS	Conventional[2]
SQL	ANSI SQL 2003	PostgreSQL 8
Massive Parallel Processing (MPP)	✓	✓
Columnar	✓	✓
Cloud	✓	✓
On-premise	✓	

[2] The new Redshift Spectrum product offers storage on S3, but this was not utilized for this benchmark in favor of conventional on-disk storage for a more apples-to-apples comparison.

Benchmark Setup

The benchmark was executed using the following setup, environment, standards, and configurations.

Data Preparation

The data sets used in the benchmark were an extension of the original UC Berkeley AMPLab BDB dataset.

AMPLab BDB Data Set

The pre-existing Big Data Benchmark (BDB) that we modeled our datasets after was provided by the UC Berkeley AMPLab. The data was sourced from the BDB S3 bucket made publicly available at s3n://big-data-benchmark/pavlo/. For more on the AMPLab BDB Data Set, please see https://amplab.cs.berkeley.edu/benchmark/ .

Extended BDB Data Set

To assess the performance of these two platforms at real-world scale, the original Berkeley BDB data sets were extended in size. For these tests, new data was generated. To be consistent with the same generation methods of the Berkeley BDB, the same Intel Hadoop Benchmark tools were used. The data preparation scripts were modfied from the original, published by the AMPLab to generate the data using a generic Amazon Linux instance on AWS and store the extended BDB data set on S3. (The original Berkeley BDB data preparation scripts use a Hadoop instance to generate the data, which was not part of this benchmark.) The script simply replicated the same data generation method as the AMPLAb scripts. The part files were then uploaded to an S3 bucket.

The extended BDB data set has the exact same schema as the original Berkeley BDB data set, which consists of two tables—rankings and uservisits[3]. The schema of these two tables are detailed below.

Additionally the extended data sets were scaled up to 10TB. A table describing the sizes of these data sets appears below as well.

[3] The documents set of unstructured data in the original Berkeley BDB was not replicated or used in this benchmark, since we were not testing the unstructured use case.

Rankings	UserVisits
pageURL varchar(300)*	**sourceIP** varchar(116)
pageRank int	**destURL** varchar(100)*
avgDuration int	**visitdate** date
	adrevenue float
	useragent varchar(256)
	countrycode char(3)
	languagecode char(6)
	searchword varchar(32)
	duration int

*The tables can be joined on Rankings pageURL and Uservisits destURL.

Data Set	**Rankings**		**UserVisits**		
Name	Row Count	Bytes	Row Count	Bytes	**Total**
MCG 1TB	0.3 billion	0.02TB	5.8 billion	0.98TB	**1TB**
MCG 5TB	1.2 billion	0.10TB	29 billion	4.90TB	**5TB**
MCG 10TB	2.5 billion	0.50TB	58 billion	9.50TB	**10TB**

Just like the original Berkeley BDB data set, the files are segmented into parts. For the 1TB data set, the rankings and uservisits data are segmented into 6,000 parts apiece, bringing the total to 12,000 files per TB. Each part of the uservisit data sets contain 982,000 rows per part. The uservisit data is a detailed log of website clickstream activity, and the rankings table is a summary of the user visit activity. Since the rankings data is created in tandem with the uservisits data—such that the two tables can be joined on the page URL fields—rankings has 1 row for every 24 rows of uservisits data—on average. The serial number of the part files was padded to 6 digits (e.g., part-000023) to allow for the large quantities of part files.

The major difference between our generated datasets and the original Berkeley BDB datasets (other than volume) was that our sets were generated in natural date order—whereas the BDB records appear to be generated using a random date order. We felt strongly that this would be closer to a real world use case, as a clickstream web log database be loaded in a natural date order as well.

These files were generated and copied up to an S3 bucket on AWS in the same region as the cluster environments.

Cluster Environments

Our benchmark included two different cluster environments—one for Actian Vector and the other for Amazon Redshift. The exact instance classes are not available for both AWS EC2 instances and Redshift. With EC2 instances, system administrators have a variety of processor, memory, and storage configuration options. It is up to the administrator to select the configuration best suited for their organization's requirements. On the other hand, Redshift has limited configuration options in terms of storage. To add storage to the cluster, you must introduce additional nodes. Therefore, the EC2 instance types for Vector were not necessarily optimized for Vector, but were selected to best match the Redshift configuration. The only difference was the disk sizes, as shown in the table below, and was due nature of the instance choices, and the excess for Vector was not material to the test.

The database management systems were each deployed on extra large 6-node clusters configured to run the benchmark queries using the MCG 1TB, 5TB, and 10TB data sets. Only 5 nodes in each of the clusters were used for processing. For Vector, the sixth node was the Hadoop namenode and was smaller than the five nodes on which Vector was installed. For Redshift, Amazon automatically includes an invisible leadernode that acts as an endpoint for the cluster, but is not involved in the storage or processing of data.

Platform	**Actian Vector**	**Amazon Redshift**
Version	5.0 (with the latest patch 53001 applied)	1.0.1583
Nodes	5	5
Instance Class	d2.8xlarge (dedicated)	ds2.8xlarge
Cluster vCPUs	180 (36 per node)	180 (36 per node)
Cluster RAM	1,220 GiB (244 GiB per node)	1,220 GiB (244 GiB per node)
Storage	240 TB HDD (24x 2000GB RAID 0 per node)	80 TB HDD

The cluster instances were created in the same AWS Region Northern Virginia (us-east-1) and put in the same placement group for maximum network performance between the cluster nodes. The default security groups recommended by the product vendors were also used.

Data Load Routines

The data was loaded into each cluster environment using the DBMS COPY function. Amazon Redshift had a native advantage of being able to access an S3 bucket within the COPY command syntax:

```
copy rankings from 's3://mcg-actian-benchmark/1TB/uservisits/' CREDENTIALS ''
delimiter ',';
```

With Actian Vector, we leveraged a third-party package called *s3fs-fuse* to mount the S3 bucket containing the benchmark data as a readable device directly on the Vector node leader. Then the contents of the data folder were loaded using the vwload utility[4] from the Linux command line:

```
vwload --verbose --fdelim "," --table uservisits mcg /s3mcg/1TB/uservisits/*
```

Once the data was loaded, in Vector, we generated statistics for the data using the following SQL command[5], which is consistent with the product documentation and best practices.

```
create statistics for all tables\g
```

The Redshift equivalent of this command is ANALYZE TABLE, but according to Redshift, it is not necessary to run this command separately. It is run automatically during table creation and data loading. It only needs to be run if a table is altered.

In Vector, the data was loaded in 90 partitions, according to the following Actian-specified best practices formula:

The number of CPU cores / 2

Also, 90 is divisible by the number of cluster nodes (5), so we know the partition count is acceptable.

For Redshift, the default round robin partition method was used.

Load times were not a part of this benchmark, due to the inability to create load processes that could be directly comparable with all other factors set equal.

Use Cases (Query Sets)

We sought to replicate the UC Berkeley AMPLab Big Data Benchmark queries in larger scale data volumes. Consistent with the original BDB methodology, each query's results were written to a table using a platform-dependent variant of CREATE TABLE AS SELECT (CTAS) to handle the large result sets. The most efficient means for handling the result set was desired. According to Amazon's documentation, the most efficient Redshift variant of the CTAS statement was written:

```
CREATE TABLE results AS %q;
```

Where %q was the query itself. According to Redshift documentation, Redshift "CTAS makes a best effort to choose the optimal distribution style and sort key based on the query plan."

According to Actian's documentation, the most efficient Vector variant of the CTAS statement was written:

```
CREATE TABLE results AS %q WITH PARTITION = (HASH ON %h 90 PARTITIONS), NOMINMAX;
```

[4] The Vector family of databases have several methods of loading external data, including a SQL COPY command—vwload was used, so that data could be loaded uninterrupted and unattended from the Linux command line using nohup

[5] SQL statements in the Ingres/Vector family of databases are terminated with \g

Where %q was the query itself and %h was the chosen field—which was pageURL for query set #1 and sourceIP for query set #2 (see below).

The NOMINMAX clause directs Vector not to rescan the results table after the CTAS had finished, which is an unnecessary step for a results table. Redshift does not perform this rescan either, but if desired, it is performed by a separate query with the EXPLAIN command.

BDB Use Case 1: Scan Query Set

Query set 1 primarily tested the throughput with which each database can read and write table data. Query set 1 had three variants:

Variant a	BI Use	Small result sets that could fit in memory and quickly displayed in a business intelligence tool (450 million rows @ 10TB)
Variant b	Intermediate Use	Result set likely too large to fit in memory of a single node 1.3 billion rows @ 10TB)
Variant c	ETL Use	Result sets are very large with result sets you might expect in a large ETL load (2.0 billion rows @ 10TB)

Query set 1 were exploratory SQL queries with potentially large result sets. The following table shows how the query was scaled:

1a	`select pageURL, pageRank from rankings where pageRank > 1000`
1b	`select pageURL, pageRank from rankings where pageRank > 100`
1c	`select pageURL, pageRank from rankings where pageRank > 10`

BDB Use Case 2: Sum Aggregation Query Set

Query set 2 applies string parsing to each input tuple then performs a high-cardinality aggregation. Query set 2 also had three variants:

Variant a	Smaller number (65,025) of aggregate groups
Variant b	Intermediate number (1.6 million) of aggregate groups
Variant c	Larger number (17 million) of aggregate groups

The following table shows how the query was scaled:

2a	`select substr(sourceIP, 1, 8), sum(adRevenue) from uservisits group by substr(sourceIP, 1, 8)`
2b	`select substr(sourceIP, 1, 10), sum(adRevenue) from uservisits group by substr(sourceIP, 1, 10)`

2c	select substr(sourceIP, 1, 12), sum(adRevenue) from uservisits group by substr(sourceIP, 1, 12)

BDB Use Case 3: Join Query Set

This query set joins a smaller table to a larger table then sorts the results. Query set 3 had a small result set with varying sizes of joins. The query set had three variants:

Variant a	Smaller JOIN within a date range of one month
Variant b	Medium JOIN within a date range of one year
Variant c	Larger JOIN within a date range of five years

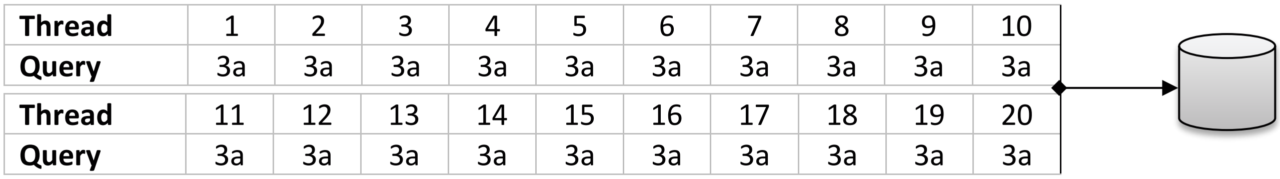
The time scanning the table and performing comparisons becomes a less significant fraction of the overall response time with the larger JOIN queries.

3a	select sourceIP, sum(adRevenue) as totalRevenue, avg(pageRank) as pageRank from rankings R join (select sourceIP, destURL, adRevenue from uservisits UV where UV.visitDate > "1970-01-01" and UV.visitDate < "1970-02-01") NUV on (R.pageURL = NUV.destURL) group by sourceIP order by totalRevenue desc limit 1;
3b	select sourceIP, sum(adRevenue) as totalRevenue, avg(pageRank) as pageRank from rankings R join (select sourceIP, destURL, adRevenue from uservisits UV where UV.visitDate > "1970-01-01" and UV.visitDate < "1971-01-01") NUV on (R.pageURL = NUV.destURL) group by sourceIP order by totalRevenue desc limit 1;
3c	select sourceIP, sum(adRevenue) as totalRevenue, avg(pageRank) as pageRank from rankings R join (select sourceIP, destURL, adRevenue from uservisits UV where UV.visitDate > "1970-01-01" and UV.visitDate < "1975-01-01") NUV on (R.pageURL = NUV.destURL) group by sourceIP order by totalRevenue desc limit 1;

Concurrency Test Harness

The final objective of the benchmark was to demonstrate Vector and Redshift performance at scale in terms of concurrent users as well. There are many ways and possible scenarios to test concurrency. To keep from introducing too much complexity into this benchmark, a simple case was used—the exact same query executed at the exact same time by 20 concurrent users.

For these tests, a concurrency test harness written in Java and using JDBC drivers was used that permitted the same query to be run in parallel and simulate multiple users using the platform at the same time. The query driver had parameters that we passed it to create multiple threads and execute the benchmark queries in parallel. For example, the following diagram demonstrates the query driver's parallel execution of the 3a query to simulate 20 concurrent users.



Thread	1	2	3	4	5	6	7	8	9	10
Query	3a	3a	3a	3a	3a	3a	3a	3a	3a	3a

Thread	11	12	13	14	15	16	17	18	19	20
Query	3a	3a	3a	3a	3a	3a	3a	3a	3a	3a

Threads 1-20 were released simultaneously.

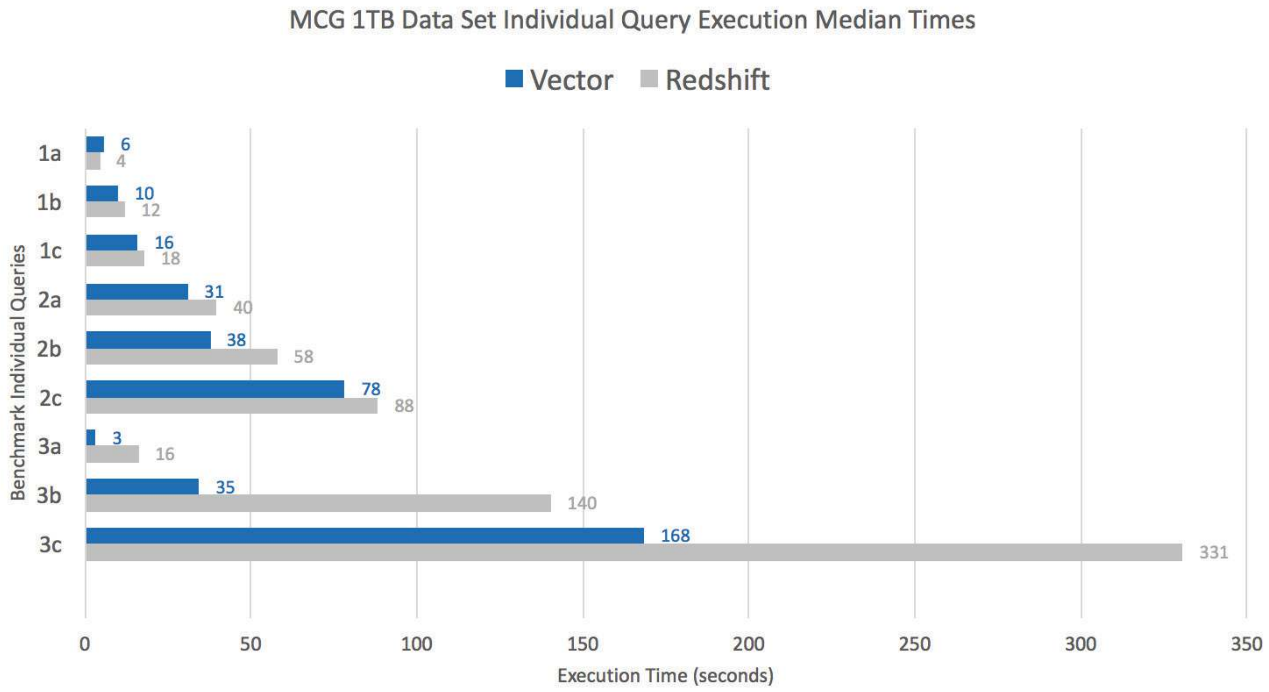
Benchmark Results

Extended Data Set – Extra Large 5-Node Cluster Results

The following tables display the individual query median and overall cumulative execution times (in seconds) for the benchmark queries using the extra-large 5-node clusters.

1TB Data Set

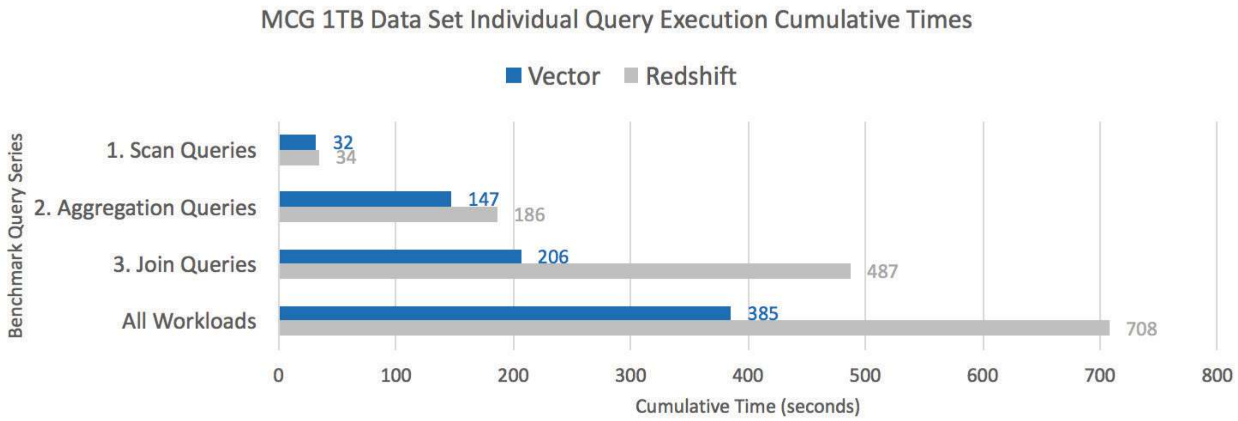
Below are the individual query results for the 1TB data set of Redshift and Vector median query execution times out of five trials.



**This graph measures time to execute queries. A shorter bar indicates a faster response time.*

In the case of the Extended 1TB data set on a 5-node clusters, Vector query response times were all faster than Redshift—with the exception of query 1a taking slightly over a second longer. On average, the Vector queries took 30% less time than Redshift. However, the biggest gap was seen during the Query #3 Join series. For Vector, Query 3b ran just over 4 times faster, and Query 3c ran about twice as fast.

Overall, the cumulative execution times (all median times added together) are presented in the following graph:

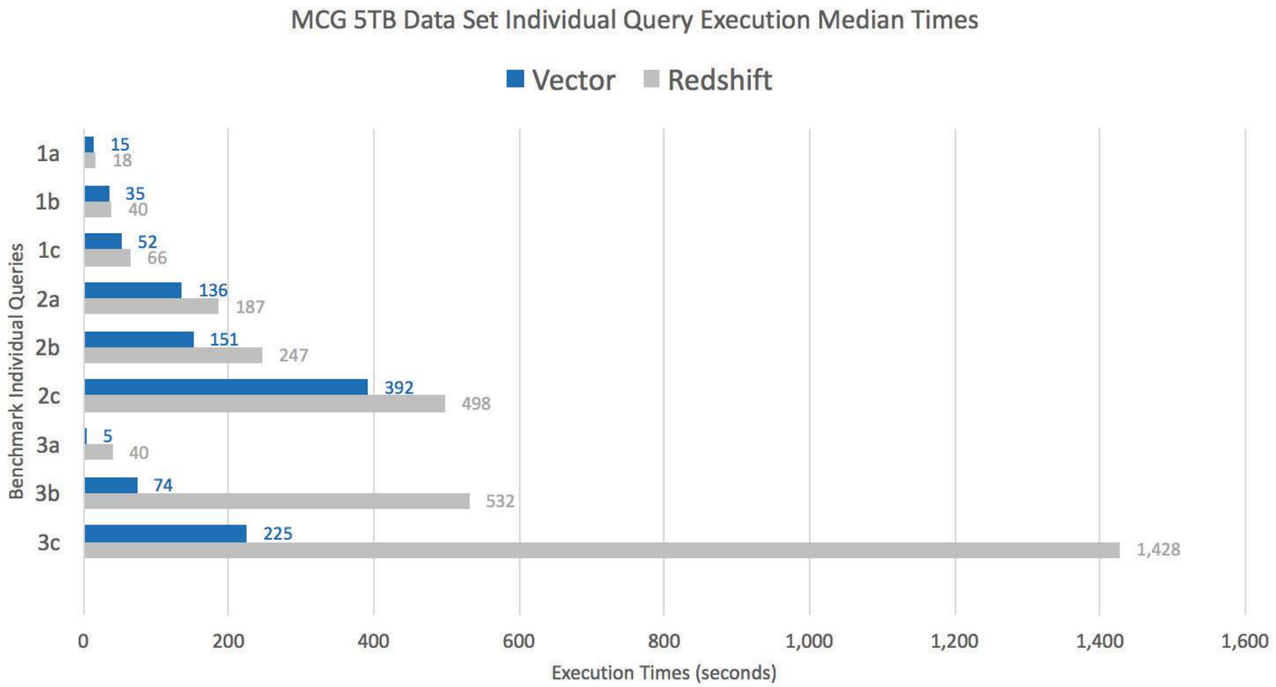


**This graph shows query execution times added together. A shorter bar indicates faster total response times across the workloads.*

Overall, Vector was twice as fast across all workloads with the clearest margin appearing in the execution of benchmark queries with JOIN clauses.

5TB Data Set

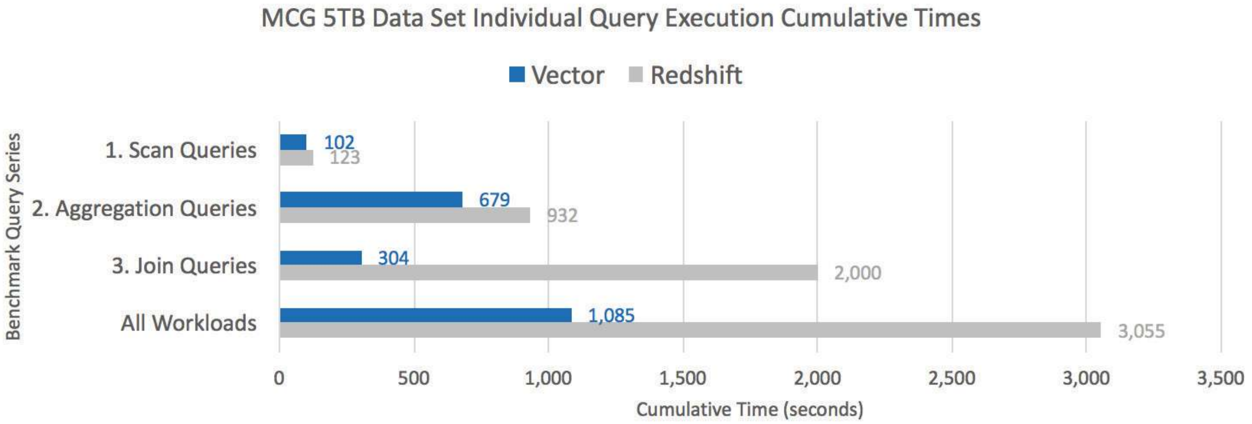
Next are the individual query results for the 5TB data set of Redshift and Vector median query execution times, again, out of five trials.



**This graph measures time to execute queries. A shorter bar indicates a faster response time.*

In the case of 5TB (i.e., 29 billion rows in the uservisits table) on the same 5-node clusters, Vector query response times were all faster than Redshift. On average, the Vector queries took 44% less time than Redshift. However, the biggest gap was seen during the Query #3 Join series. For Vector, Query 3b ran over 7 times faster, and Query 3c ran over 6 times as fast.

Overall, the cumulative 5TB execution times (all median times added together) are presented in the following graph:

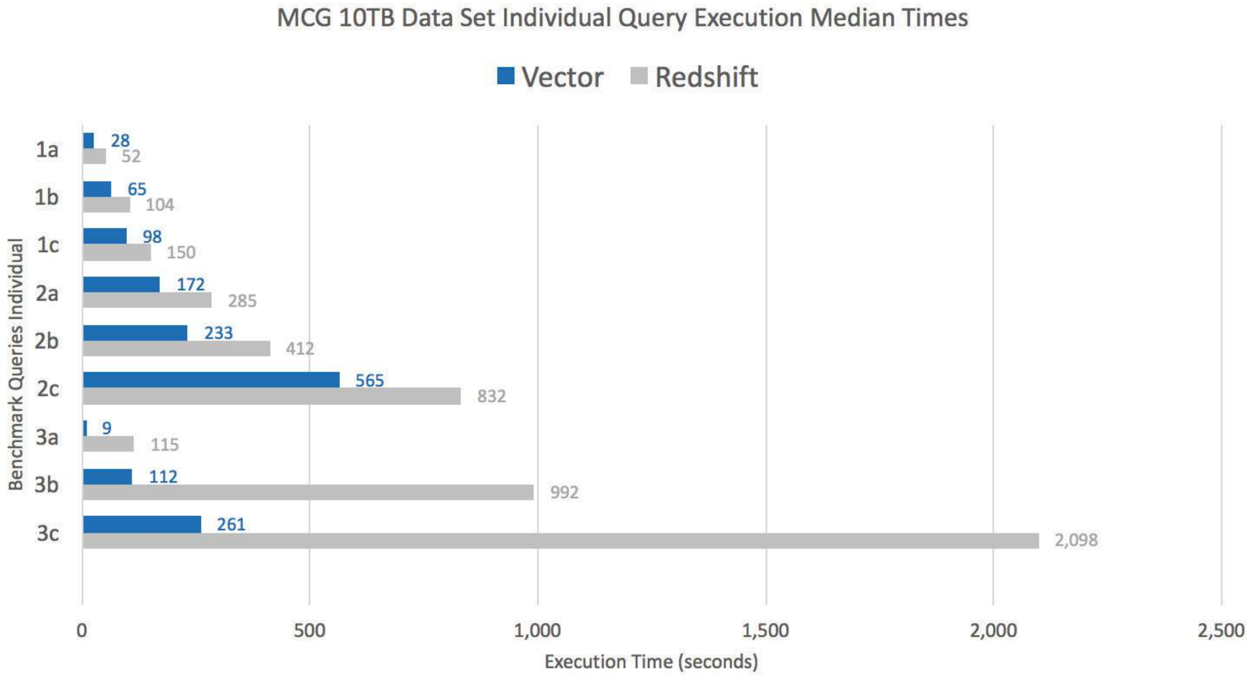


**This graph shows query execution times added together. A shorter bar indicates faster total response times across the workloads.*

Overall, Vector was nearly three times as fast across all workloads. Note the divergence in performance between the two platforms in the execution of benchmark queries with JOIN clauses.

10TB Data Set

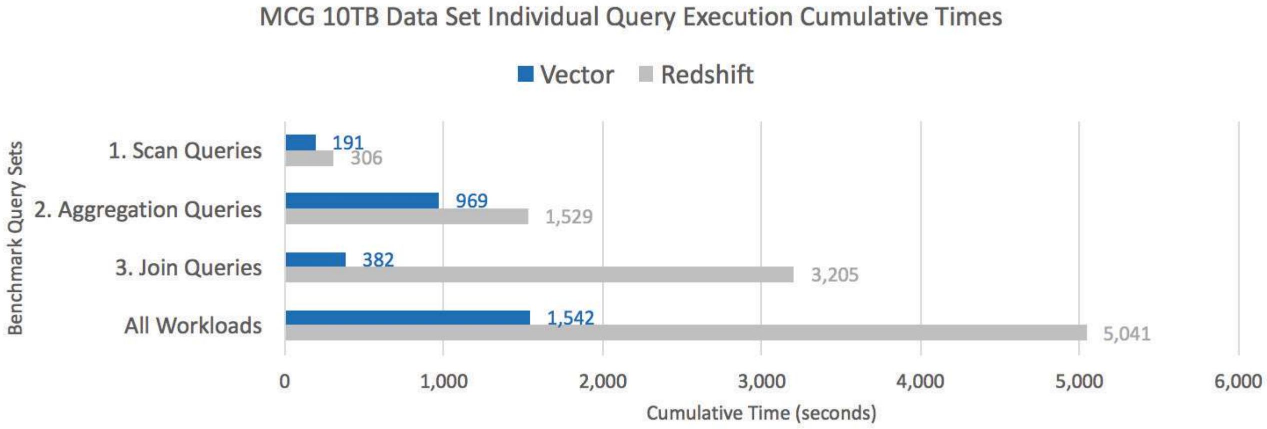
Finally, we have the individual query results for the 10TB data set (with 58 billion uservisits rows) of Redshift and Vector median query execution times, again, out of five trials.



**This graph measures time to execute queries. A shorter bar indicates a faster response time.*

In the case of 10TB on the same 5-node clusters, Vector query response times were all faster than Redshift. On the whole, the Vector queries took 56% less time than Redshift. Once again, the continued separation is seen with the Query #3 Join series. For Vector, Query 3a was an impressive 12 times faster; Query 3b finished over 9 times faster; and Query 3c ran over 8 times as fast.

Overall, the cumulative 10TB execution times (with all median times added together) are presented in the following graph:



**This graph shows query execution times added together. A shorter bar indicates faster total response times across the workloads.*

This time, Vector was over three times as fast across all workloads. Once again, notice the divergence in performance between the two platforms in the execution of benchmark queries with JOIN clauses of 2x, 7x, and 8x scaling from 1TB, 5TB, and 10TB, respectively.

Concurrency Tests

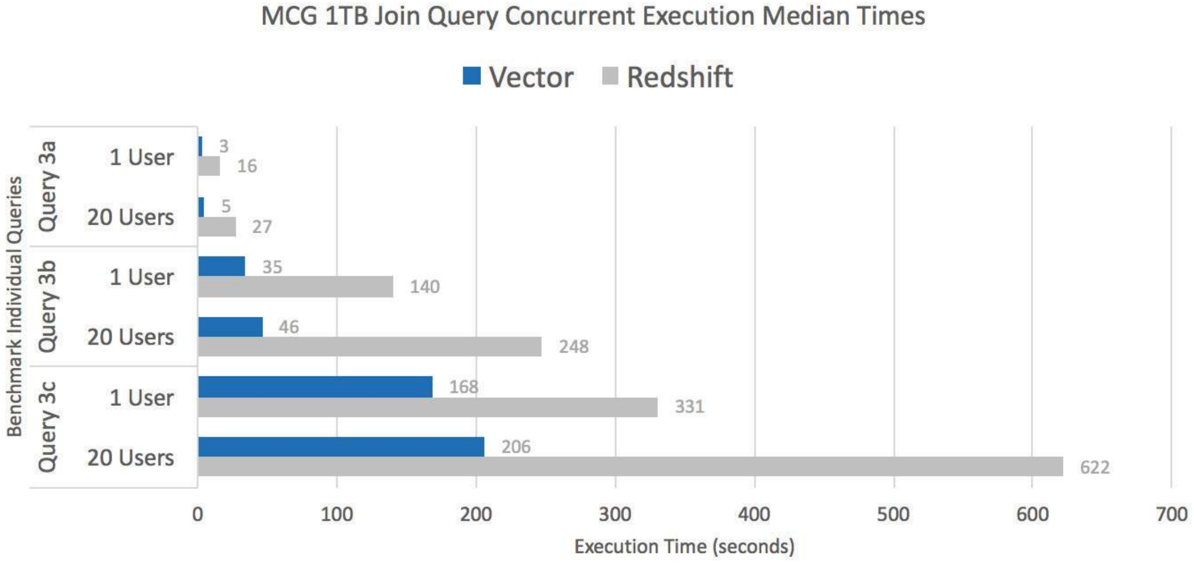
The benchmark concurrency tests were executed using the multi-thread JDBC query test harness. The queries were executed simulating 20 concurrent users.

Query #3c on Redshift did not complete at 10TB when scaled up to 20 concurrent users. Vector was able to complete all tests at all data scale and concurrency levels.

The following tables display the median execution times (in seconds) over five runs of the benchmark queries executed to simulate 20 concurrent users. The Scan Query (Query Set #1) results are omitted, because they were so short, they did not contribute to overall results.

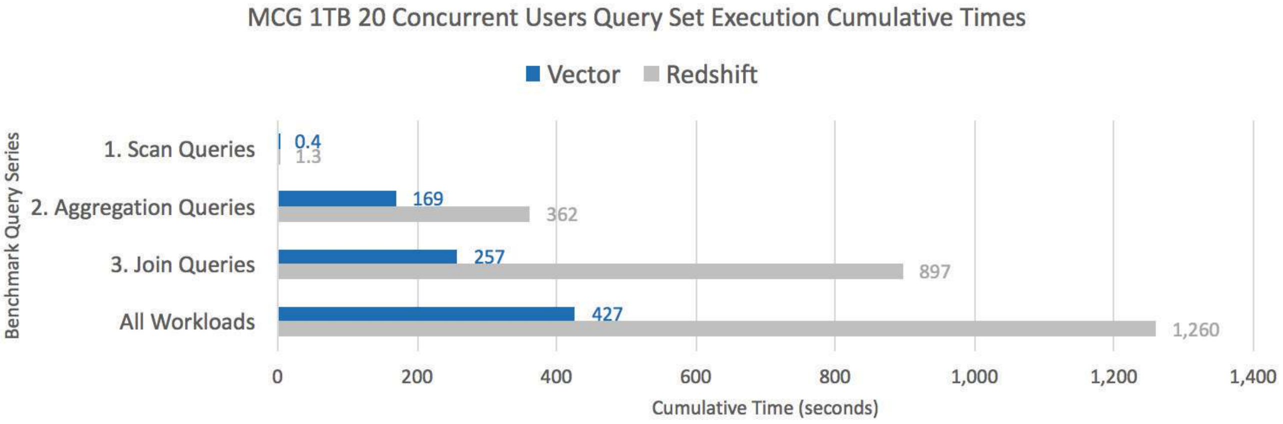
1TB Data Set with 20 Concurrent Users

In the case of 1TB on the 5-node clusters, Vector concurrency response times were all faster than Redshift. On the whole, the Vector queries took 66% less time than Redshift. Once again, the separation is seen with the Query #3 Join series. The first table shows Join Queries (Query Set #3) results—1 user versus 20 users at 1TB. For Vector at 20 users, Queries 3a, 3b, and 3c were between 3 and 6 times faster.



**This graph measures time to execute queries. A shorter bar indicates a faster response time.*

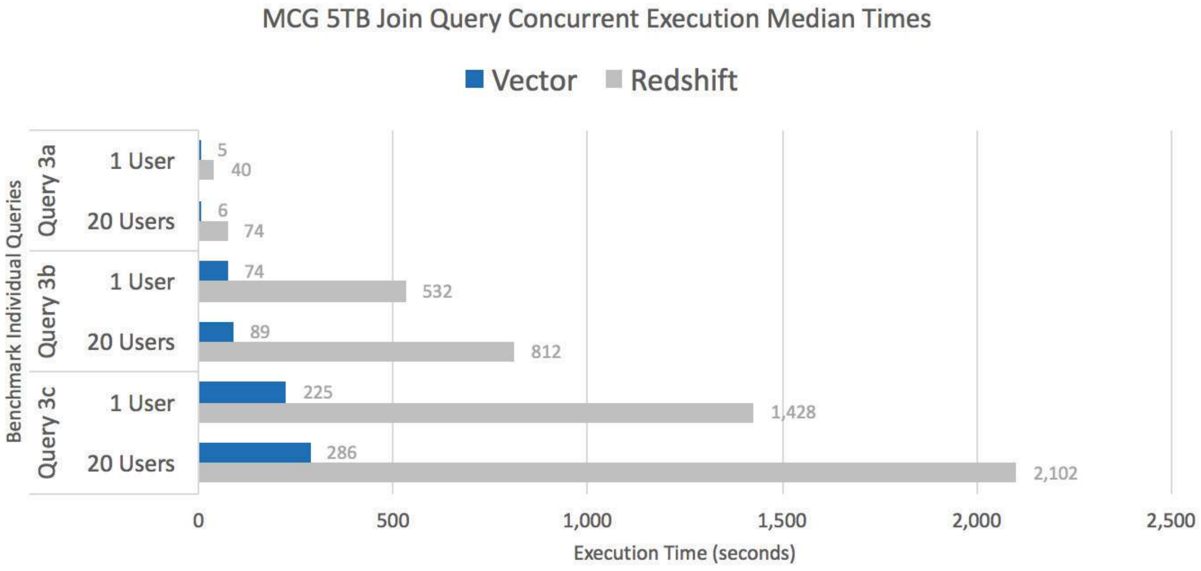
Overall, the cumulative 1TB execution times (with all median times added together) from 20 user concurrency test are presented in the following graph. Vector was 3 times faster across all workloads.



*This graph shows query execution times added together. A shorter bar indicates faster total response times across the workloads.

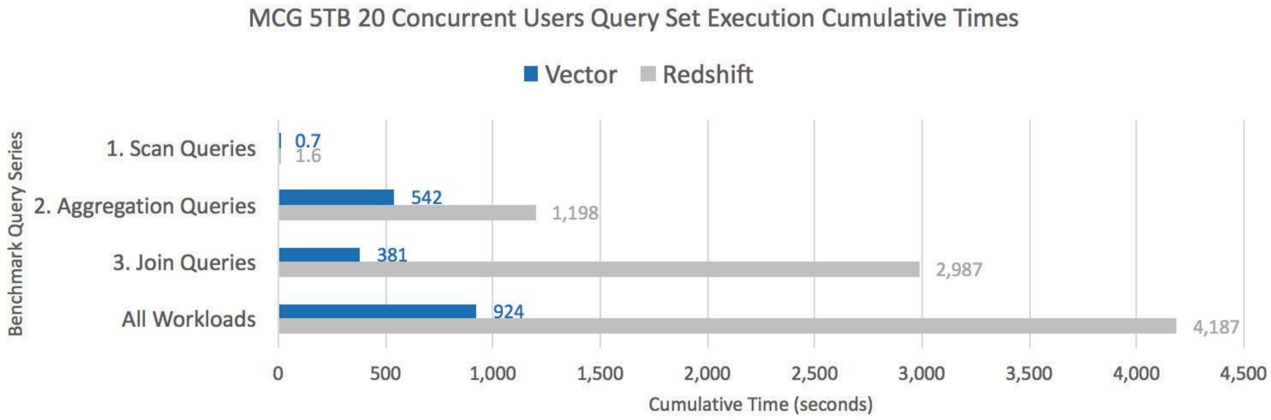
5TB Data Set with 20 Concurrent Users

In the case of 5TB, Vector query response times for 20 users were all faster than Redshift. On the whole, the Vector queries took 78% less time than Redshift. The most significant difference is seen with the Query #3 Join series. For Vector at 20 users, Queries 3a, 3b, and 3c were 12, 9, and 7 times faster, respectively. The following table shows Join Queries (Query Set #3) results using the 5TB data set:



*This graph measures time to execute queries. A shorter bar indicates a faster response time.

Overall, the cumulative 5TB execution times (with all median times added together) from 20 user concurrency test are presented in the following graph:

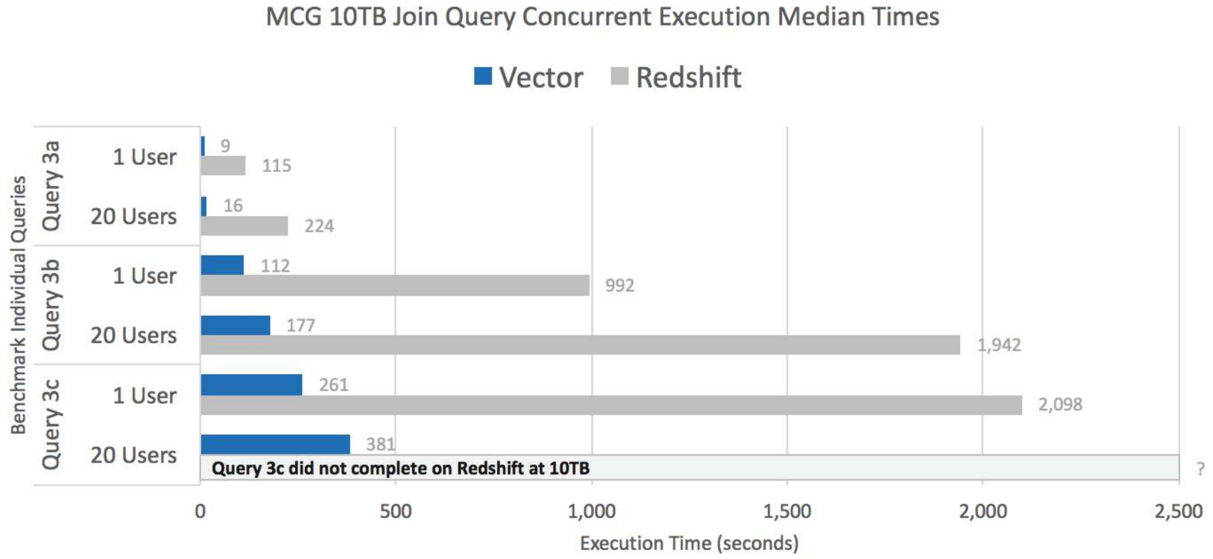


**This graph shows query execution times added together. A shorter bar indicates faster total response times across the workloads.*

Vector averaged 4.7 times faster across all workloads. Once again, notice the difference in performance between the two platforms in the execution of benchmark queries with joins—which were nearly 8 times faster on Vector.

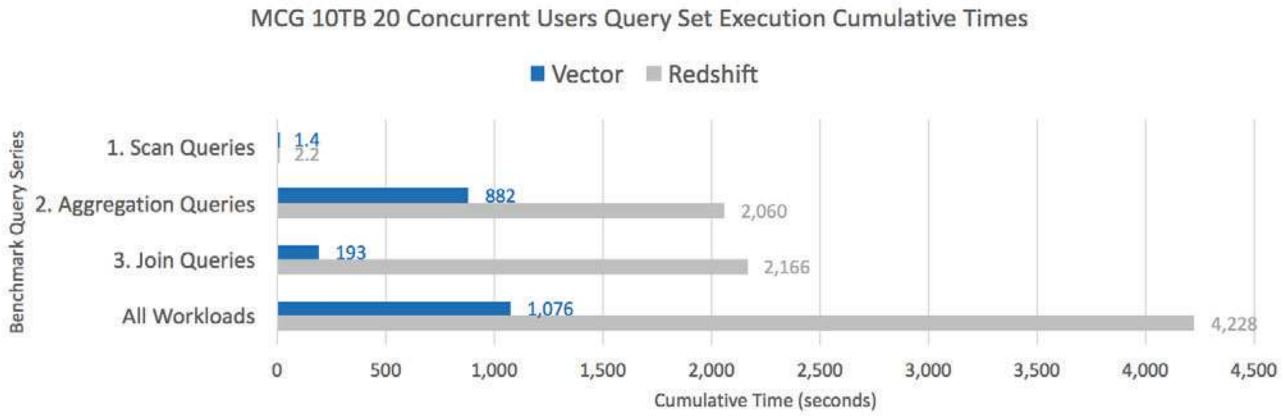
10TB Data Set with 20 Concurrent Users

In the case of 10TB, Vector query response times were all faster than Redshift. Vector queries took 75% less time than Redshift. For Join queries at 20 users, Query 3a was nearly 14 times faster on Vector. Query 3b was 11 times faster. Query 3c did not complete on Redshift at 20 users after 2 hours of waiting. Vector completed all query runs. The first table shows Join Queries (Query Set #3) results at 10TB.



**This graph measures time to execute queries. A shorter bar indicates a faster response time.*

Overall, the cumulative 10TB execution times (with all median times added together and excluding 3c) from 20 user concurrency test are presented in the following graph. Vector averaged nearly 5 times faster across all workloads.

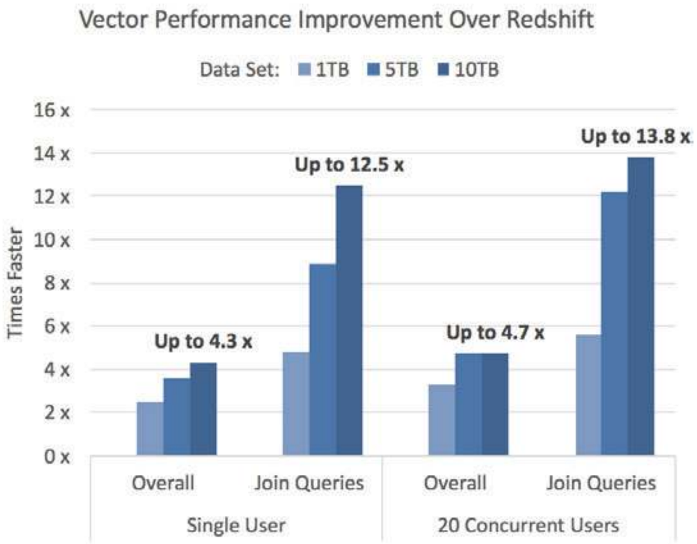


**This graph shows query execution times added together. A shorter bar indicates faster total response times across the workloads.*

Conclusion

Cloud databases, notably on Amazon Web Services, are a way to avoid large capital expenditures, provision quickly, and provide performance for advanced analytic queries in the enterprise. Relational databases with analytic capabilities continue to support the advanced analytic workloads of the organization with performance, scale and concurrency. In a representative set of corporate-complex queries, Actian Vector consistently outperformed Redshift.

Overall, the benchmark results were insightful in revealing the query execution performance of Actian Vector and Redshift scaling up by data volume and concurrency—revealing some of the differentiators in the two products. Overall average query response times showed Vector performing up to nearly 5 times faster than Redshift when 20 concurrent users are sending simultaneous requests. A noteworthy finding was executing the benchmark queries containing joins, with Actian Vector performing 12.5 (single user) to nearly 14 times (20 concurrent users) faster than Redshift.[6]



These results are most likely explained by the technology underlying Vector. The basic architecture of Actian Vector is the Actian patented X100 engine utilizes a concept known as "vectorized query execution" where processing of data is done in chunks of cache-fitting vectors. Vector performs “single instruction, multiple data” processes by leveraging the same operation on multiple data simultaneously and exploiting the parallelism capabilities of modern hardware. It reduces overhead found in conventional "one-row-at-a-time processing" found in other platforms. Additionally, the compressed column-oriented format uses a scan-optimized buffer manager.

Overall, Actian Vector on AWS or on-premise is an excellent choice for data-driven companies needing high performance and a scalable analytical database in the cloud or to augment their current, on-premises data warehouse with a hybrid architecture—at a reasonable cost.

[6] It should also be noted that in 2011, Vector set a new record in a TPC-H benchmark at scale factor 100, delivering 340% higher performance of the previous best record while improving price/performance by 25%. Today they still lead in the 3,000GB category according to the TPC.

About MCG Global Services

William McKnight is President of McKnight Consulting Group (MCG) Global Services (http://www.mcknightcg.com). He is an internationally recognized authority in information management. His consulting work has included many of the Global 2000 and numerous midmarket companies. His teams have won several best practice competitions for their implementations and many of his clients have gone public with their success stories. His strategies form the information management plan for leading companies in various industries.

Jake Dolezal has two decades of experience in the Information Management field with expertise in business intelligence, analytics, data warehousing, statistics, data modeling and integration, data visualization, master data management, and data quality. Jake has experience across a broad array of industries, including: healthcare, education, government, manufacturing, engineering, hospitality, and gaming. He has a doctorate in information management from Syracuse University.

MCG services span strategy, implementation, and training for turning information into the asset it needs to be for your organization. We strategize, design and deploy in the disciplines of Master Data Management, Big Data Strategy, Data Warehousing, Analytic Databases and Business Intelligence.

About Actian

Actian, the hybrid data management, analytics and integration company, delivers data as a competitive advantage to thousands of customers worldwide. Through the deployment of innovative hybrid data technologies and solutions Actian ensures that business critical systems can transact and integrate at their very best – on premise, in the cloud or both. Thousands of forward-thinking organizations around the globe trust Actian to help them solve the toughest data challenges to transform how they run their businesses, today and in the future. For more information about Actian Vector and the entire Actian portfolio of hybrid data management, analytics and integration solutions on-premise or in the cloud, visit https://www.actian.com.

Actian Vector for SMP systems is detailed here. Vector for Hadoop is described here. Downloads for Actian Vector on-premise here, on the Amazon Marketplace here and on Microsoft Azure here.