Cloud Database Performance Benchmark

Product Profile and Evaluation:
Actian Vector and Microsoft SQL Server

By William McKnight, Jake Dolezal and Roger Barker
McKnight Consulting Group Global Services
February 2018

Sponsored by

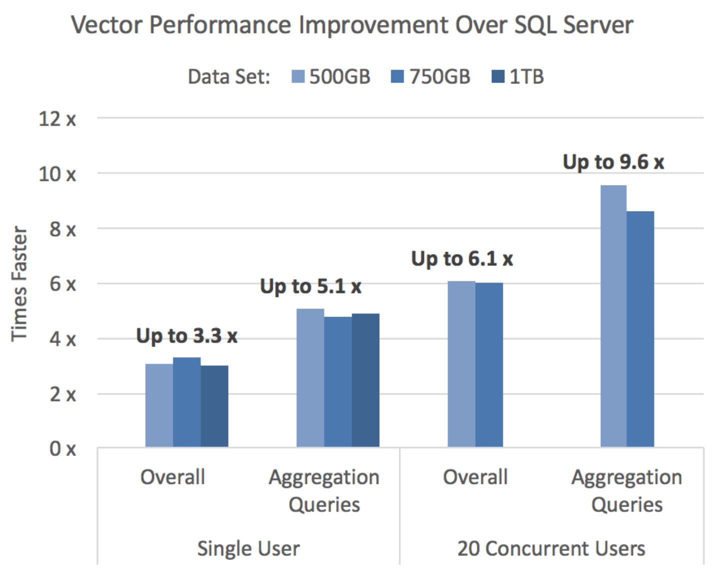
Executive Overview

Data-driven organizations rely on database platforms to analyze volumes of data at high speed to derive timely insights. Data volumes within a modern organization's information ecosystem are rapidly expanding—placing significant performance demands on legacy architectures. Today, to fully harness data to gain a competitive advantage, businesses need high levels of performance and reliability to provide timely analytical insights. Relying on legacy platforms or maintaining status quo in the IT shop, because "that's what we know," can prove to be an Achilles' heel.

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 Microsoft SQL Server. 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 500 GB, 750 GB, and 1 TB of data to simulate real world demands. Also, multiple simultaneous user concurrency was tested. The testing was conducted using the same size and class of Amazon Web Services (AWS) EC2 Instances.

Overall, the benchmark results were insightful in revealing the query execution performance of Actian Vector and SQL Server at scale—both in terms of data volume and concurrent users—revealing some of the differentiators in the two products. Overall average query response times showed Vector performing between 3 and 6 times faster than SQL Server. The most noteworthy finding was specifically results of executing the benchmark queries containing an aggregation, with Actian Vector performing 5 to nearly 10 times faster than SQL Server[2].



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. We did not compare the on-premise versions of Actian Vector or SQL Server. Both are available in the cloud on AWS and Azure, but only AWS was used for the benchmark.
[2] Some of the more complex queries at 750GB and 1TB did not complete on SQL Server.

Analytics Platform Offerings

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 and comes from a variety of sources. These analytics-driven businesses leverage this data, for example, for performing 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 Microsoft SQL Server—the former with Vector processing and columnar based database architecture that scale and provide high-speed analytics, and the latter a versatile, multi-purpose database platform for many organizations and their applications. In addition, Vector is available for developers as a free on-premise community edition and as a download in the AWS marketplace with single-click support. SQL Server Express is offered free on the AWS marketplace. Other versions, like Standard and Enterprise, have their license fee baked into the hourly rate for on-demand usage.

About the Platforms

	Actian Vector	*SQL Server*
Company	Actian	Microsoft
Released	2014	1989
Current Version	5.0	2017
Storage	Hadoop HDFS	Conventional
SQL	ANSI SQL 2003	Transact-SQL
Massive Parallel Processing (MPP)	✓	
Columnar	✓	[3]
Cloud	✓	✓
On-premise	✓	✓

[3] SQL Server is row-oriented by default. Users must create and maintain Clustered ColumnStore Indexes for tables in order to gain the performance benefits of column-orientation.

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[4]. The schema of these two tables are detailed below.

Additionally the extended data sets were scaled up to 1TB. A table describing the sizes of these data sets appears below as well.

[4] 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 500GB	123 million	8GB	2.9 billion	492GB	**500GB**
MCG 750GB	184 million	12GB	4.4 billion	738GB	**750GB**
MCG 1TB	245 million	16GB	5.8 billion	1,008GB	**1TB**

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 for 1TB. Each part of the uservisit data sets contain 945,000 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 8 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 instance environments.

Instances

Our benchmark included two different single node environments—one for Actian Vector and the other for Microsoft SQL Server. The exact instance classes were available for running both Vector and SQL Server.

The database management systems were each deployed on extra large single-node instances configured to run the benchmark queries using the MCG 500GB, 750GB, and 1TB data sets.

Platform	**Actian Vector**	**Microsoft SQL Server**
Version	5.0 (with the latest patch 53001 applied)	2017 Enterprise (RTM-CU1) 14.0.3006.16
Nodes	1	1
Instance Class	d2.8xlarge (dedicated)	d2.8xlarge (dedicated)
Instance vCPUs	36	36
Instance RAM	244 GiB	244 GiB
Storage	48 TB HDD (24x 2000GB RAID 0)	48 TB HDD (24x 2000GB RAID 0)

The EC2 instances were created in the same AWS Region Northern Virginia (us-east-1) and put in the same placement. The default security groups recommended by the product vendors were also used.

Data Load Routines

The data was loaded into each instance. In order to load the data, one option was to use SQL Server Integration Services (SSIS). However, SSIS does not have an S3 connector. There are some third party alternatives. Ultimately, the data was downloaded from S3 and loaded into SQL Server as a batch process.

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[5] from the Linux command line:

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

Although load times were not benchmarked, it took about 7 hours for Vector to load the 1TB dataset, while it took 16 hours for SQL Server.

Once the data was loaded, in Vector, we generated statistics for the data using the following SQL command[6], which is consistent with the product documentation and best practices.

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

After the SQL Server data was loaded, Clustered ColumnStore Indexes were created using the following command.

[5] 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

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

```
CREATE CLUSTERED COLUMNSTORE INDEX cl_rankings ON [dbo].[rankings]
CREATE CLUSTERED COLUMNSTORE INDEX cl_uservisits ON [dbo].[uservisits]
```

Otherwise, all other default, out-of-the-box configurations for both Vector and SQL Server were used. We wanted to show as-is, default performance configured for general use, without trying to tailor the database engine specifically for these benchmark workloads.

Use Cases (Query Sets)

We sought to replicate the UC Berkeley AMPLab Big Data Benchmark queries in larger scale data volumes with a few exceptions.

First, we deviated from the original BDB methodology which had each query's results written to a table using a platform-dependent variant of CREATE TABLE AS SELECT (CTAS). During preliminary runs of the SQL Server and Vector benchmarks, we discovered a significant difference in CTAS behavior on the two platforms—Vector writes its result set in its native columnar format, while SQL Server creates a row store. We experimented with forcing Vector to write a row-oriented table and applying a ColumnStore index to the results table on SQL Server during creation. Both introduced unwanted complexity and overhead that could have potentially skewed the benchmark. Additionally, SQL Server does not have an efficient or documented way of sending results to a NULL device from a command line prompt, which is easy on a Linux based platform, so that method was not considered.

We came to the conclusion to change from CTAS to SELECT COUNT(*) FROM as a method of handling the large result sets. The most efficient means for handling the result set was desired. Thus, Query Sets #1 and #2 (see below) were encapsulated with the following:

```
SELECT COUNT(*) FROM (%q);
```

Where %q was the query itself.

The only negative to this method was Query Set #1 execution times became so fast, they did not contribute to the overall benchmark in a significant way.

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 (45 million rows @ 1TB)
Variant b	Intermediate Use	Result set likely too large to fit in memory of a single node 125 million rows @ 1TB)

Variant c	ETL Use	Result sets are very large with result sets you might expect in a large ETL load (208 million rows @ 1TB)

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`

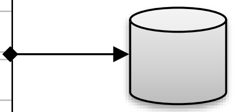
	`join (select sourceIP, destURL, adRevenue from uservisits UV where UV.visitDate > "1970-01-01" and UV.visitDate < "1970-02-01") NUV on (R.pageURL = NUV.destURL)` `group by sourceIP order by totalRevenue desc limit 1;`
3b	`select sourceIP, sum(adRevenue) as totalRevenue, avg(pageRank) as pageRank from 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 SQL Server performance at scale, not only of larger data volumes, but also considering 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.

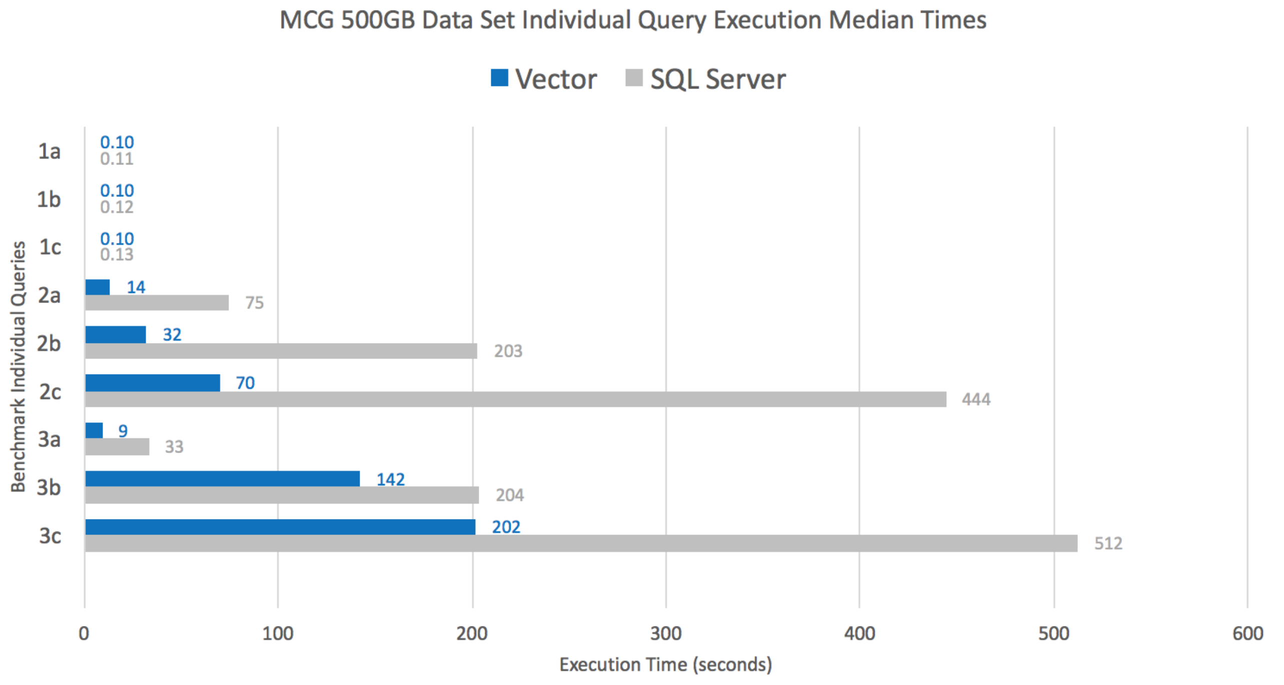
Benchmark Results

Extended Data Set – Extra Large Single-Node Instance Results

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

500GB Data Set

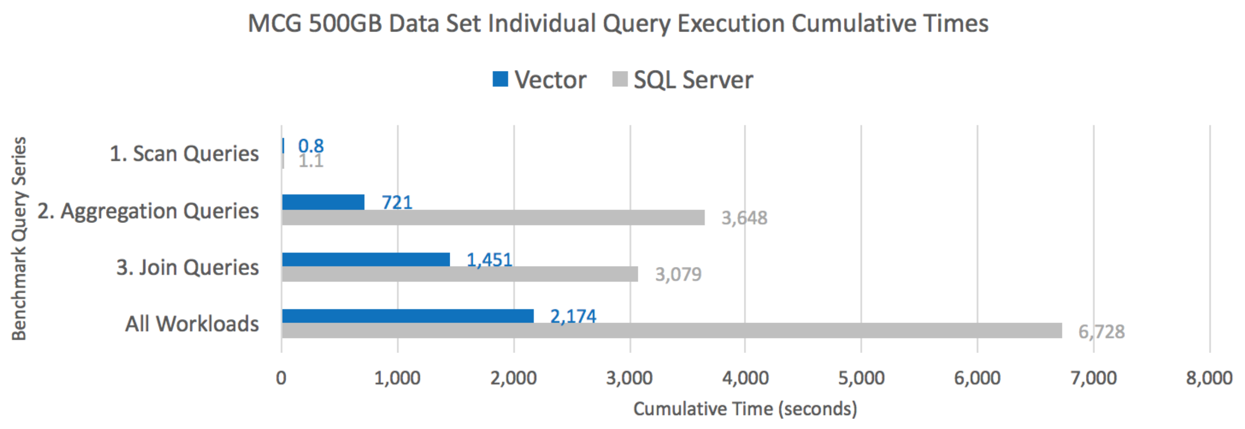
Below are the individual query results for the 500GB data set of SQL Server and Vector median query execution times out of five trials.



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

In the case of the Extended 500GB data set on a single-node instances, Vector query response times were all faster than SQL Server. On average, the Vector queries took 68% less time than SQL Server. However, the biggest gap was seen during the Query #2 Aggregation series. For Vector, queries 2b and 2c ran over 6 times faster. Another result of note in the JOIN queries: Query 3c on Vector actually ran faster than Query 3b on SQL Server—that is, Vector ran the JOIN on a date range of five years in less time than it took SQL Server to run the same JOIN on a range of one year.

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

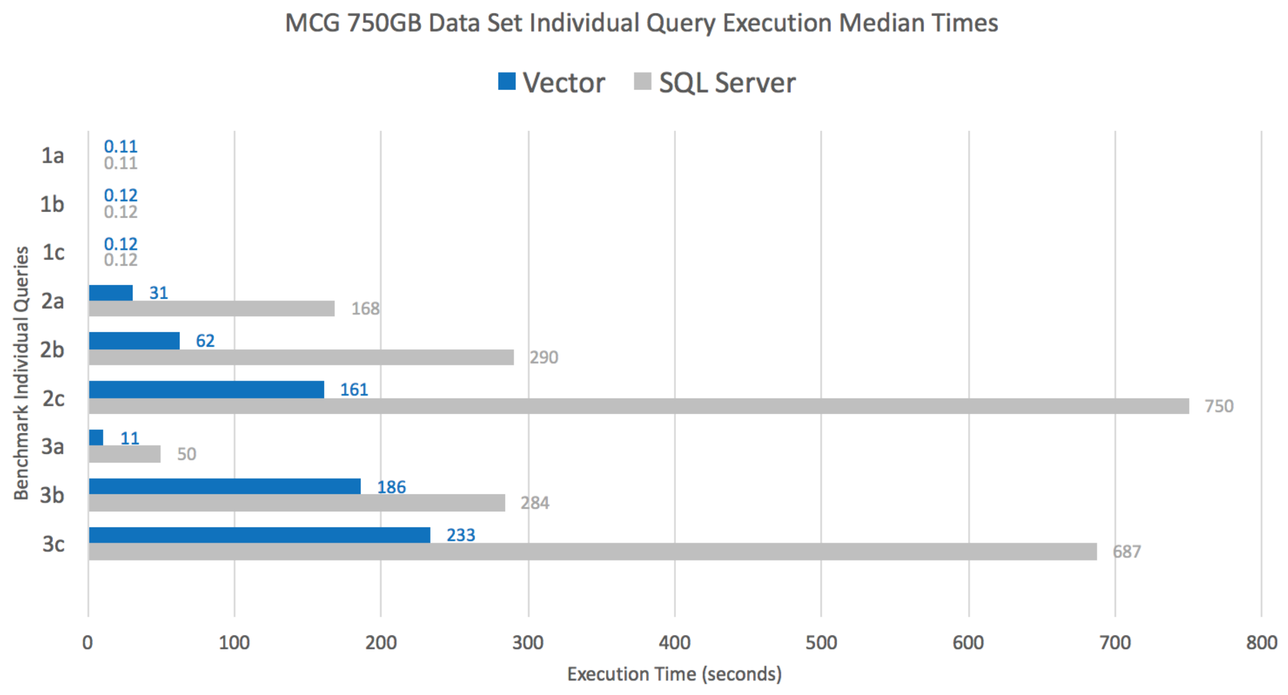


This graph shows query execution times added together. A shorter bar indicates faster total response times across the workloads.

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

750GB Data Set

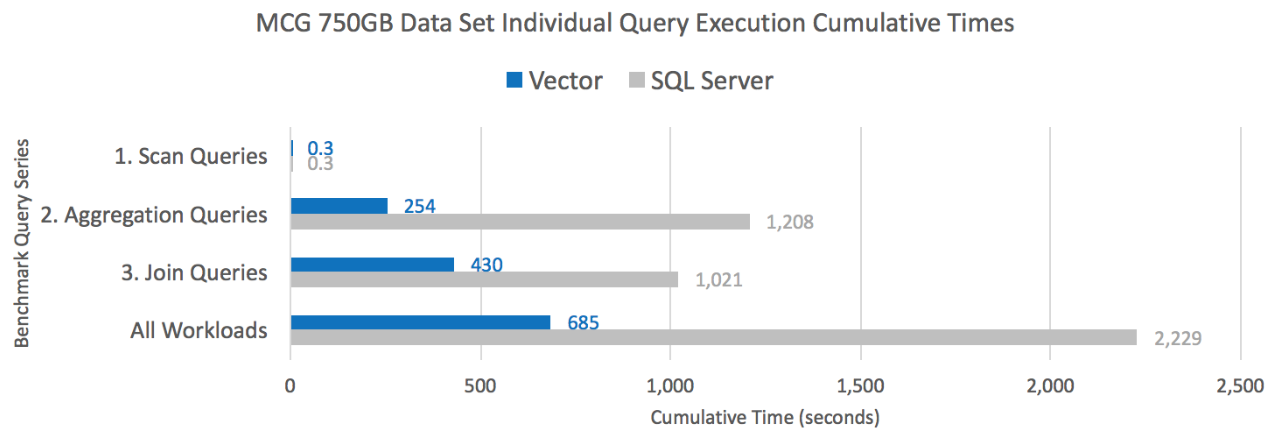
Next are the individual query results for the 750GB data set of SQL Server and Vector median query execution times, again, out of five trials.



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

In the case of 750GB (i.e., over 4 billion rows in the uservisits table) on the same single-node instances, Vector query response times were all faster than SQL Server. On average, the Vector queries took 69% less time than SQL Server. However, the biggest gap was seen during the Query #2 aggregation series. These queries ran nearly 5 times as fast across the board.

Overall, the cumulative 750GB execution times (all median times added together) are presented in the following graph:

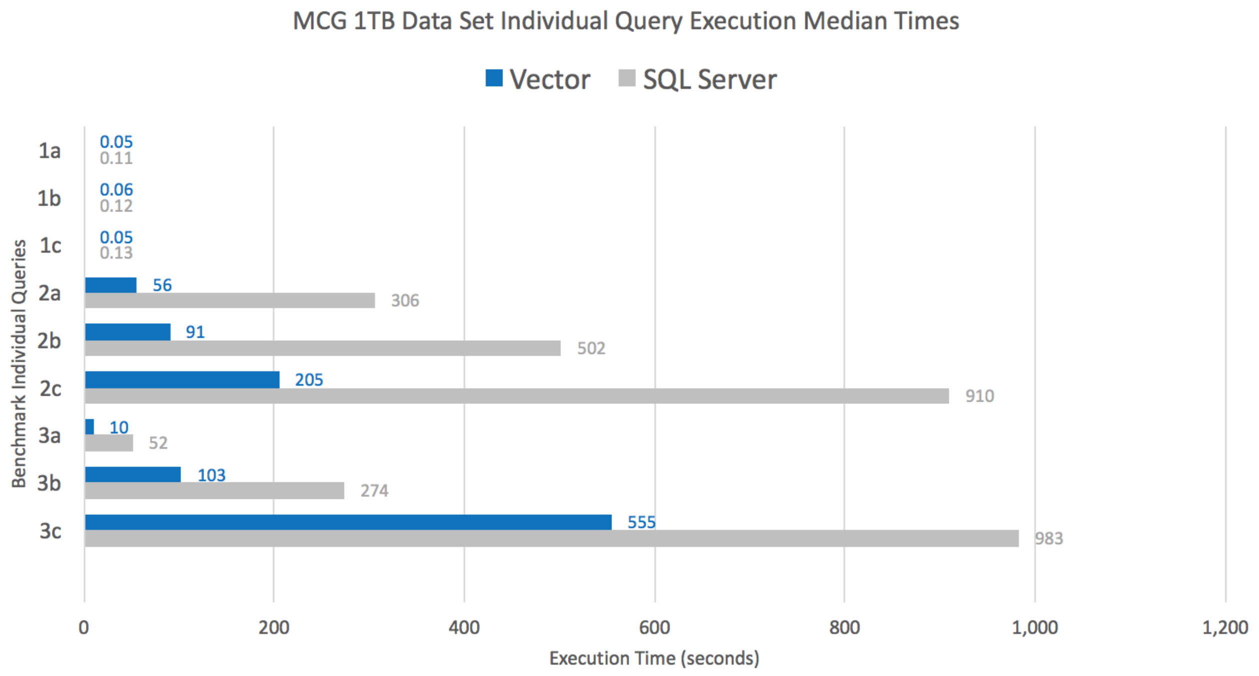


**This graph shows query execution times added together. A shorter bar indicates faster total response times across the workloads.*

Overall, Vector was over three times faster across all workloads. Note the divergence in performance between the two platforms in the execution of benchmark queries with aggregations.

1TB Data Set

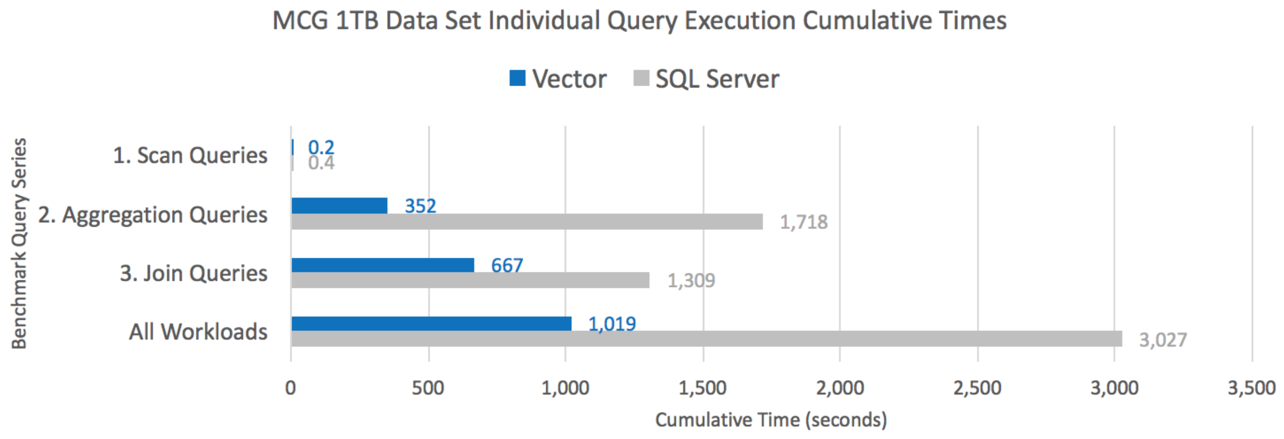
Finally, we have the individual query results for the 1TB data set (with nearly 6 billion uservisits rows) of SQL Server and Vector median query execution times, again, out of five trials.



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

In the case of 1TB on the same single-node instances, Vector query response times were all faster than SQL Server. On the whole, the Vector queries took 66% less time than SQL Server. Once again, the continued separation is seen with the Query #2 Aggregation series. For Vector, Queries 2a and 2b were 5.5 times faster

Overall, the cumulative 1TB 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 three times faster across all workloads. Once again, notice the difference in performance between the two platforms in the execution of benchmark queries with aggregations—which were 5 times faster on Vector.

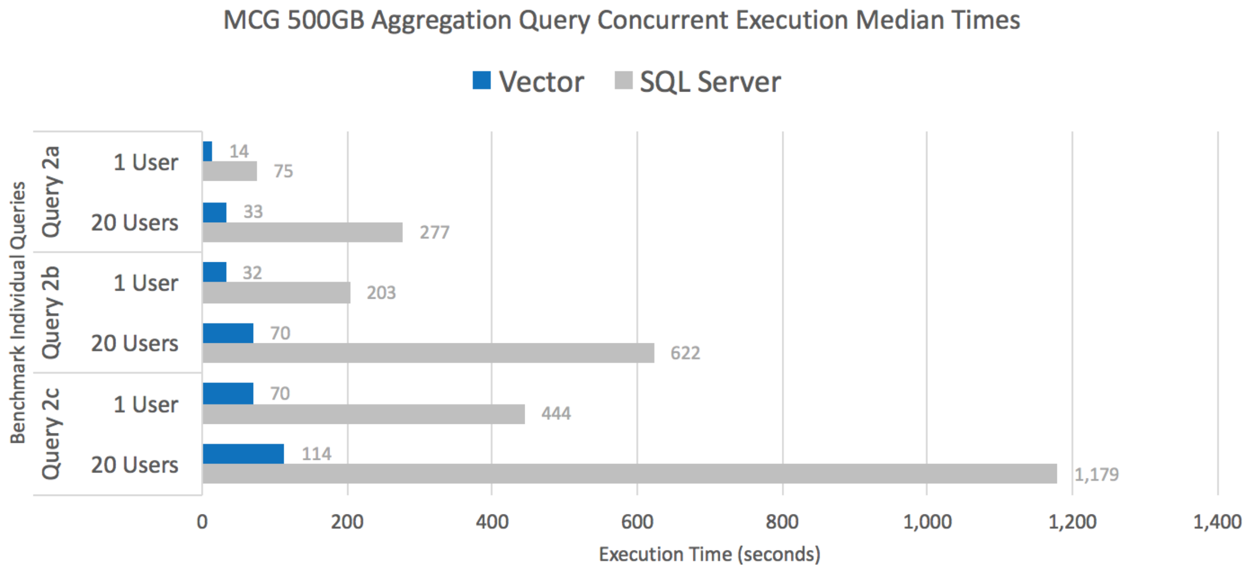
Concurrency Tests

The benchmark concurrency tests were executed using the multi-thread JDBC query test harness. The queries were executed simulating 20 concurrent users. Only the 500GB and 750GB results sets are presented, because Query Sets #2 or #3 on SQL Server did not complete at 1TB. Also, individual queries 2c and 3c would not complete on SQL Server at 750GB 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.

500GB Data Set with 20 Concurrent Users

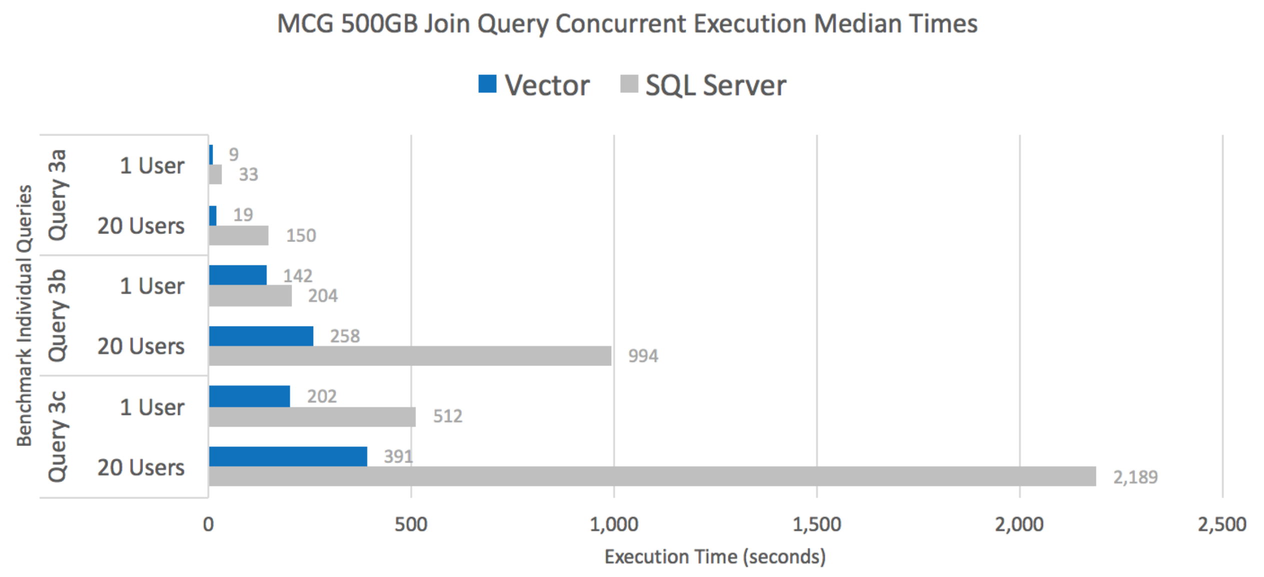
The first table shows Aggregation Queries (Query Set #2) results—1 user versus 20 users at 500GB:



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

In the case of 500GB on the same single-node instances, Vector query response times were all faster than SQL Server. On the whole, the Vector queries took 84% less time than SQL Server. Once again, the separation is seen with the Query #2 Aggregation series. For Vector at 20 users, Queries 2a, 2b, and 2c were 8 times, 9 times, and over 10 times faster, respectively.

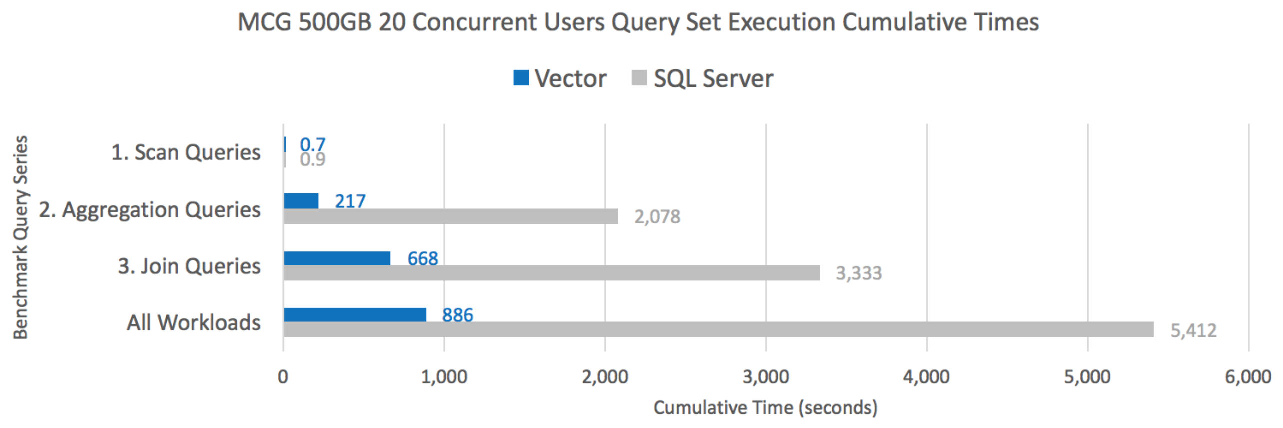
The next table shows Join Queries (Query Set #3) results—1 user versus 20 users using the 500GB data set:



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

For the Join Query Set at 500GB with 20 users, Queries 3a was 8 times faster. Queries 3b and 3c were 4 and 6 times faster respectively.

Overall, the cumulative 500GB execution times (with all median times added together) from 20 user concurrency test are presented in the following graph:

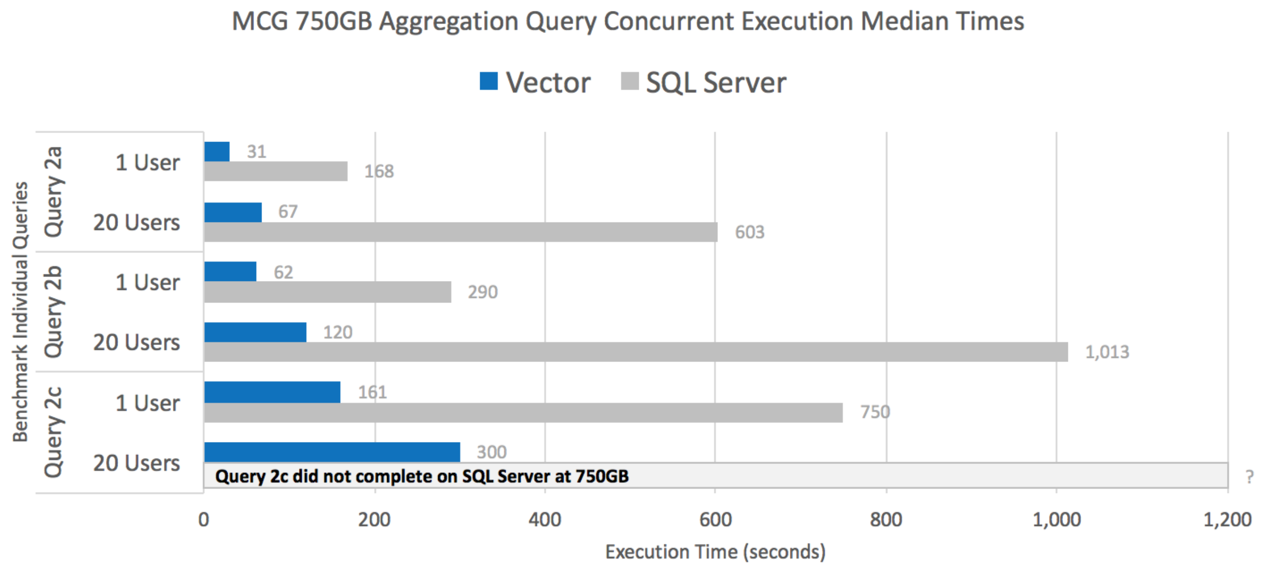


**This graph shows query execution times added together. A shorter bar indicates faster total response times across the workloads.*

Vector was 6 times faster across all workloads. Once again, notice the difference in performance between the two platforms in the execution of benchmark queries with aggregations—which were nearly 10 times faster on Vector.

750GB Data Set with 20 Concurrent Users

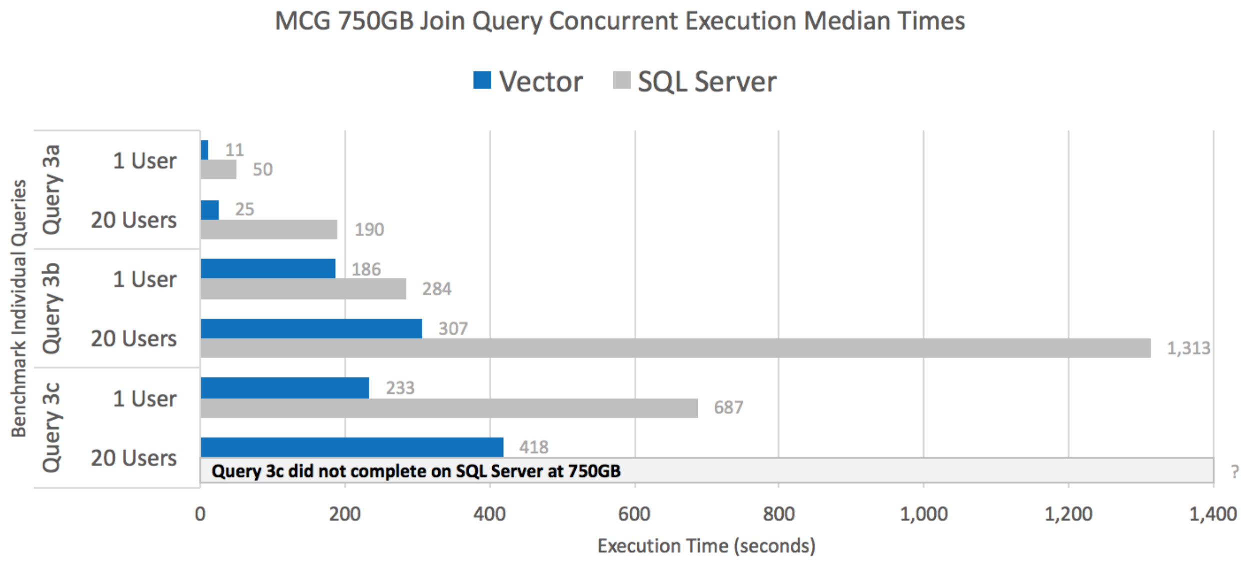
The following table shows Aggregation Queries (Query Set #2) results using the 750GB data set:



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

In the case of 750GB, Vector query response times were all faster than SQL Server. Query 2c did not complete on SQL Server and the test harness request was cancelled after 2 hours of no response. On the whole (excluding 2c), the Vector queries took 83% less time than SQL Server. The most significant difference is seen with the Query #2 Aggregation series. For Vector at 20 users, Queries 2a and 2b were both 9 times faster.

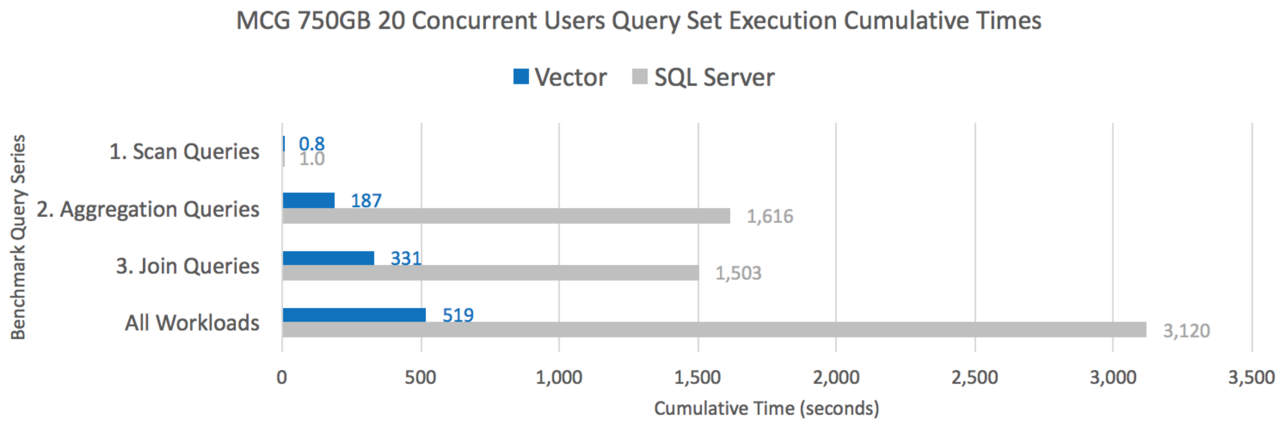
The next table shows Join Queries (Query Set #3) results—1 user versus 20 users using the 750GB data set:



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

Again, the most complex query (3c) did not complete on SQL Server after 2 hours of waiting. For the Join Query Set with 20 users, Query 3a was 8 times faster. Query 3b was over 4 times faster.

Overall, the cumulative 750GB execution times (with all median times added together and excluding Queries 2c and 3c) from 20 user concurrency test are presented in the following graph:



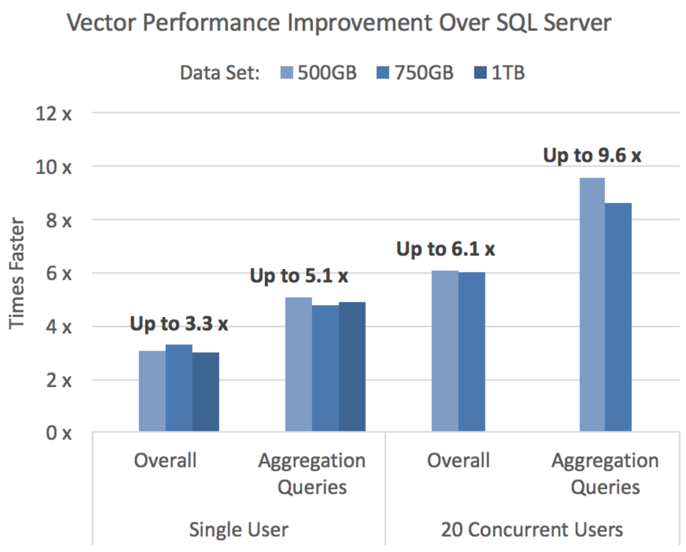
**This graph shows query execution times added together. A shorter bar indicates faster total response times across the workloads.*

Vector was 6 times faster across all workloads. Once again, notice the difference in performance between the two platforms in the execution of benchmark queries with aggregations—which were nearly 9 times faster on Vector.

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 SQL Server.

Overall, the benchmark results were insightful in revealing the query execution performance of Actian Vector and SQL Server at scale—revealing some of the differentiators in the two products. Overall average query response times showed Vector performing between 3 times faster on single user tests and over 6 times faster than SQL Server when 20 concurrent users are sending simultaneous requests. A noteworthy finding was executing the benchmark queries containing aggregations, with Actian Vector performing 5 (single user) to nearly 10 times (20 concurrent users) faster than SQL Server.[7]



These results are most likely explained by the technology underlying Vector. The architecture of Vector leverages the Actian patented X100 engine that 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, Azure, 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.

[7] 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. 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.

Find out more about Actian Vector for single servers and for Hadoop clusters, or get links to downloads for on-premise deployment or cloud instances.