Cloud Analytics Database Performance Report

Actian Vector up to 20x faster than Snowflake

MCG Global Services Benchmark Results

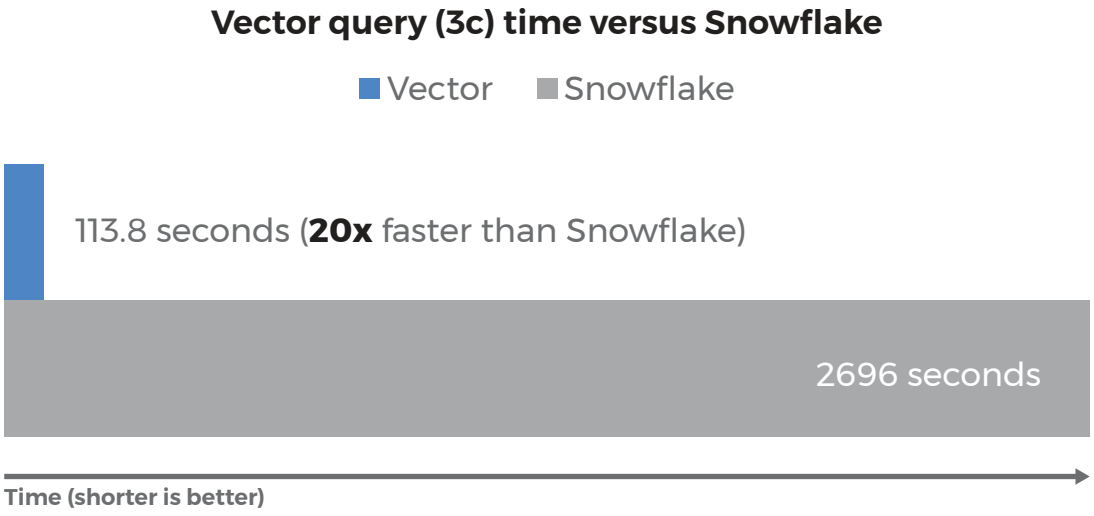
ACTIAN™

Key insights

- Independent benchmark performed by MCG Global Services
- Test based on Berkeley AMPlab Big Data benchmark workload
- Tested on a 16-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 20 seconds to run** while **Snowflake** took more than 5 minutes.

Vector up to 20X faster than Snowflake

On a 16 node AWS cluster on a 10 TB database



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

ACTIAN™

Cloud Database Performance Benchmark

Product Profile and Evaluation: Actian Vector and Snowflake

By McKnight Consulting Group Global Services

March 2018

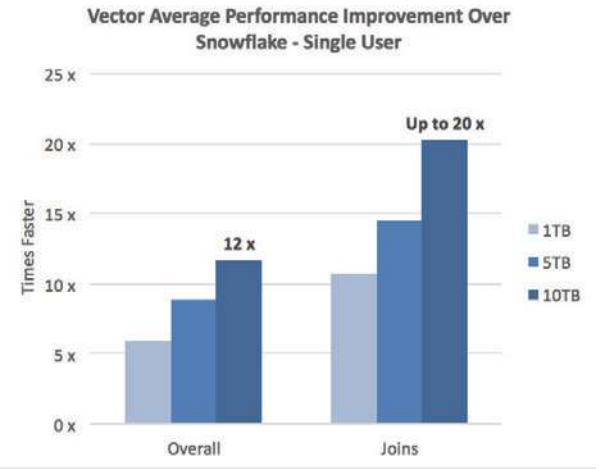
Sponsored by

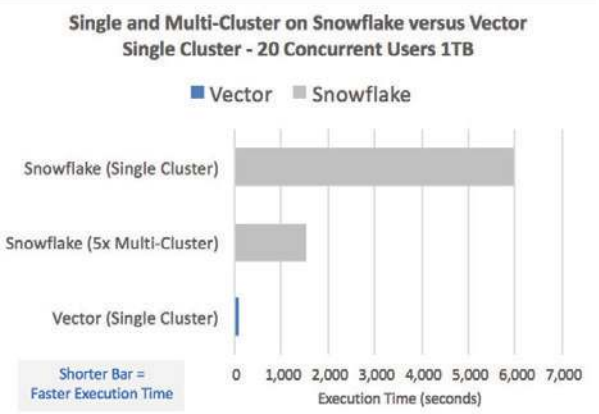
Executive Overview

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 organizations' 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, analytical-workload solutions on Actian Vector and Snowflake Computing. The benchmark is designed to simulate a set of real-world scenarios to answer fundamental business questions that an organization from nearly any industry sector might ask.

The benchmark tested the scalability of corporate-complex workloads. The tests were based on the industry standard UC Berkeley AMPLab Big Data Benchmark, with the data set sizes being extended to 1, 5, and 10 TB of data to simulate real-world Big Data demands. The testing was conducted using single clusters on Amazon Web Services (AWS) that were equivalent in cluster node counts and comparable in cost per hour to run.





Measuring execution performance of queries with increasing data volumes and concurrency, benchmark results for Actian Vector and Snowflake revealed some performance differentiators between the two products. Actian Vector performed 12 times faster overall and up to 20 times faster on queries with joins on single-user tests.

A revealing finding emerged when we stressed the workload by simulating 20 concurrent users. With Snowflake's multi-cluster option enabled and 5 Snowflake clusters versus a single Vector cluster, Vector was still 17 times faster than Snowflake overall.

[1] Since, unlike Actian Vector, Snowflake is limited to cloud-only deployment, to enable an equitable comparison, all tests were done in an Amazon Web Services instance.

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 such as machine, sensor, log, sentiment, clickstream, and geospatial data. Data-driven businesses leverage data for many use cases including 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 desired analytic insights to optimize their business performance.

This paper focuses on relational analytical databases in the cloud since cloud deployments are at an all-time high and poised to expand dramatically. The cloud offers opportunities to differentiate and innovate with these database systems more rapidly than ever before possible. Further, the cloud has been a disruptive technology, as cloud storage tends to cost less, enables rapid server deployment, and offers elastic scalability as compared with on-premise deployments. For these reasons many data-driven companies are increasingly migrating to the cloud.

This paper compares two popular cloud-based analytical databases: Actian Vector and Snowflake. Both relational analytical databases are based on massively parallel processing (MPP) and columnar-based database architectures that scale and provide high-speed analytics. Note that, while the benchmark measures the cloud-based performance of both offerings, unlike Snowflake, Vector is also available as an on-premise offering. In addition, Vector is available for developers as a free, downloadable on-premise community edition. The community edition is also available in the cloud at the AWS and Azure marketplaces for single-click deployment.

About the Platforms

	Actian Vector	*Snowflake*
Company	Actian	Snowflake
Released	2014	2015
Current Version	5.0	2.34
Storage	Hadoop HDFS	Amazon S3
SQL	ANSI SQL 2003	SnowSQL
Massive Parallel Processing (MPP)	✓	✓
Columnar	✓	✓
AWS Cloud	✓	✓
Azure Cloud	✓	
On-premise	✓	

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 data set.

AMPLab BDB Data Set

The pre-existing Big Data Benchmark (BDB) that we modeled our data sets after was provided by the UC Berkeley AMPLab. The data was sourced from the BDB S3 bucket publicly available at s3n://big-data-benchmark/pavlo/. For more about the AMPLab BDB data set, 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 identical schema as the original Berkeley BDB data set, which consists of two tables—rankings and uservisits[2]. The schemas of these two tables are detailed below.

Additionally, the extended data sets were scaled up to 10TB. A table below describes the sizes of these data sets.

[2] 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**

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 each, bringing the total to 12,000 files per TB. Each part of the uservisit data sets contains 982,000 rows. 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 pageURL fields—rankings has on average 1 row for every 24 rows of uservisits data. The serial number of the part files was padded to 6 digits (e.g., part-000023) to allow for the large number of part files.

The major difference between our generated data sets and the original Berkeley BDB data sets (other than volume) was that our sets were generated in natural date order, whereas the BDB records appeared 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 would be loaded in natural date order.

These files were generated and uploaded 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 Snowflake—using Amazon EC2. 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.

Although Snowflake uses Amazon EC2 instances for its compute layer, the instance classes are not documented; it is proprietary information that Snowflake does not disclose. Vector can run on any of the EC2 instance classes. Thus, we rightsized the EC2 instance type for Vector and sized the clusters to create the most "apples-to-apples" comparison as possible.

In this benchmark, several selection criteria needed to be compared when evaluating and selecting hardware configurations: number of cluster nodes, processing power (number of and type of CPU cores), memory, storage, disk I/O, and cost. With limited hardware configuration information about Snowflake's product, we selected based on knowns. The following is an explanation of each factor:

- **Number of cluster nodes** – Snowflake recommends their X-Large 16-node cluster for general use and data warehousing applications. Vector recommends at least 3 nodes, so we chose 16 nodes for our clusters.
- **Processing power** – Snowflake does not disclose this information. We chose the EC2 r4 family to use with Vector for its High Frequency Intel Xeon E5-2686 v4 (Broadwell) processors—good for general use and typical among many of the EC2 instance classes.
- **Memory** – Likewise, Snowflake does not disclose this information. For Vector, we chose EC2 r4 for its DDR4 memory—also common among most of the EC2 instance types.
- **Storage and disk I/O** – Snowflake is architected to use S3 as its storage layer. Clearly, this is not a configuarable option out-of-the-box for EC2. We believed the closest equivalent to this for Vector was Amazon's Elastic Band Storage (EBS), rather than dedicated drives. Regardless, the benchmark queries were written and clusters sized to minimize the amount of disk I/O. When profiling query runs, we wanted to see disk operations consume 5% or less of the overall query processing time.
- **Cost** – This was a difficult criterion to consider, because even though the bundled cost of Snowflake is known, it's difficult to ascertain how much we are actually paying for in computing hardware. We chose the $3.00 per node per hour Snowflake Enterprise version. For Vector, we chose the r4.8xlarge instance type with an On Demand price of $2.128. Storage costs were not considered, again, because they are not comparable (S3 versus EBS).

In summary, the following table compares all factors considered.

Platform	Actian Vector	Snowflake
Version	5.0 (with the latest patch 53001 applied)	1.0.1583
Instance Class	r4.8xlarge (dedicated, no shared tenancy)	X-Large
Nodes	16	16
Cluster vCPUs	512 (32 per node)	Unknown
Cluster RAM	3,904 GiB (244 GiB per node)	Unknown
Storage	16TB EBS (1TB per node)	S3
Computing Cost	$34.05 per hour ($2.128 per node)	$48 per hour ($3.00 per node)

The database management systems were each deployed on extra-large 16-node clusters configured to run the benchmark queries using the MCG 1TB, 5TB, and 10TB data sets. Only 16 nodes in each of the clusters were used for processing. For Vector, a 17th node was the Hadoop namenode, which was smaller than the other nodes on which Vector was installed.

The Vector 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. We also used the default security groups recommended by the product vendors.

Data Load Routines

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

```
copy into rankings from s3://mcg-actian-benchmark/1TB/rankings/
credentials=(aws_key_id='XXXXX' aws_secret_key='XXXXXXX')
file_format = (format_name = BENCHMARKCSV);
```

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 we loaded the contents of the data folder using the vwload utility[3] 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[4], which is consistent with the product documentation.

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

According to Snowflake:

Snowflake manages all aspects of how this data is stored in S3—the organization, file size, structure, compression, metadata, statistics, and other aspects of data storage are handled by Snowflake. The data objects stored by Snowflake in S3 are not directly visible nor accessible by customers; they are only accessible through SQL query operations run using Snowflake.

There is no operation in Snowflake for collecting database statistics. It is handled by the engine.

In Vector, the data was loaded in 256 partitions, according to the following Actian-specified best practices formula:

> The number of CPU cores / 2

Also, 256 is divisible by the number of cluster nodes (16), so we knew the partition count was acceptable.

For Snowflake, partitioning is handled internally—see the statement above.

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

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

Load times were not part of this benchmark because of the inability to create load processes that were comparable with all other factors set equal. We found both times, with the methods chosen, to be within the bounds of acceptability for an enterprise.

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 that had each query's results written to a table using a platform-dependent variant of CREATE TABLE AS SELECT (CTAS). Because we do not know the difference in disk-write latency between EBS (Vector) and S3 (Snowflake), we wanted I/O to impact the benchmark results as little as possible. Additionally, Snowflake has no documented way of sending results to a NULL device from a command line prompt, which is easy, based on a Linux-based platform, so that method was not considered.

We decided to change from CTAS to SELECT COUNT(*) FROM as a method of handling the large result sets because we wanted to use the most efficient means for handling the result sets. 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 that they did not contribute significantly to the overall benchmark.

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 be 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 as you might expect in a large ETL load (2.0 billion rows @ 10TB)

Query set 1 contained 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 applied string parsing to each input tuple, then performed a high-cardinality aggregation. Query set 2 also had 3 variants:

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

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 joined a smaller table to a larger table, then sorted the results. Query set 3 had a small result set with varying sizes of joins. The query set had 3 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 to scan the table and perform comparisons became 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;`

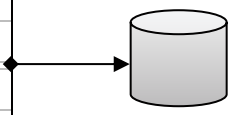
3c	`select sourceIP, sum(adRevenue) as totalRevenue, avg(pageRank) as pageRank from rankings R`
	`join (select sourceIP, destURL, adRevenue from uservisits UV where UV.visitDate > "1970-01-01" and UV.visitDate < "1975-01-01") NUV on (R.pageURL = NUV.destURL)`
	`group by sourceIP order by totalRevenue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

Concurrency Test Harness

The final objective of the benchmark was to demonstrate Vector and Snowflake performance at scale in terms of concurrent users. There are many ways and possible scenarios to test concurrency. We employed a use case where the identical query was executed at the exact same time by 20 concurrent users.

For these tests, we created a concurrency test harness written in Java using JDBC drivers. This approach permitted the same query to be run in parallel and simulate multiple users accessing the platform at the same time. The query driver had parameters that we passed to 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



Although threads 1–20 were released simultaneously, the two platforms behaved very differently.

When we ran benchmark query sets 1 and 2, Snowflake executed 8 of the queries simultaneously. The remaining 12 were queued, and Snowflake waited for 1 of the first 8 to complete before releasing the next query for processing. It continued this way until all 20 queries completed. For query set 3, Snowflake ran 6 at a time.

Snowflake has a session variable, max_concurrency_level, that allows the user to specify the maximum number of queries the cluster may run concurrently. However, this is an upper boundary and not an absolute limit. Query complexity and available resources appear to affect it, and this probably explains why query set 3 allowed only 6 queries to run—due to its JOIN statement. Snowflake also recommends NOT changing this value without testing. We did some testing, and although it's not documented, we found the ceiling value to be around 10 for our configuration. Throttling this value had little impact on overall results.

Snowflake documents that a best practice to get their product to handle concurrency is to use their multi-cluster processing option.[5] Snowflake multi-cluster acts by automatically introducing

[5] The Snowflake multi-cluster feature is not enabled by default—it must be approved and enabled by Snowflake.

additional clusters (of the same size) to handle concurrent query requests and then automatically suspending them once the workloads are complete.

With Snowflake multi-cluster enabled, we explored the performance effects by using a multi-cluster configuration, which we discuss later in this report.

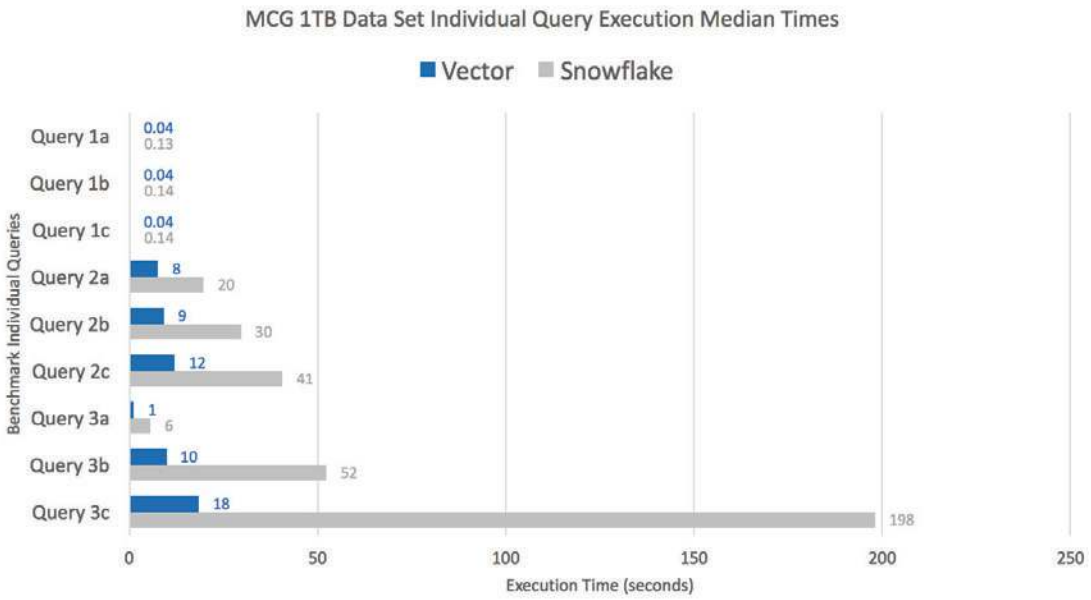
Benchmark Results

Single User Extra Large 16-node Cluster Results

The following tables display the individual query median and overall cumulative execution times (in seconds) for the benchmark queries using the 16-node clusters.

1TB Data Set

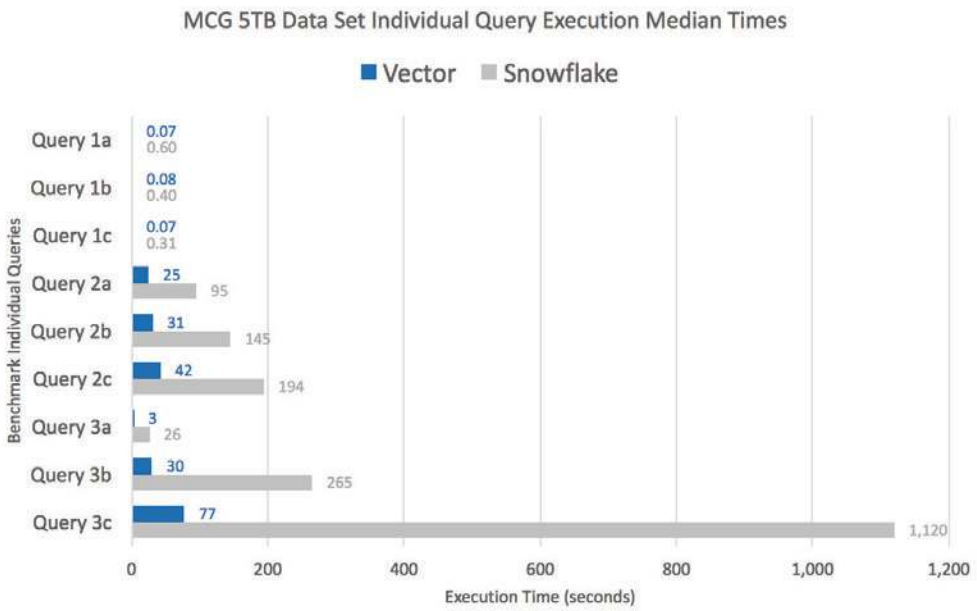
In the case of the extended 1TB data set on a 16-node cluster, Vector query response times were all faster than Snowflake. Overall, Vector was 6 times faster than Snowflake. However, the biggest gap appeared during the Query 3 Join series. For Vector, Query 3c ran almost 11 times faster. Below are the individual query results for the 1TB data set of Snowflake and Vector median query execution times out of 5 trials.



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

5TB Data Set

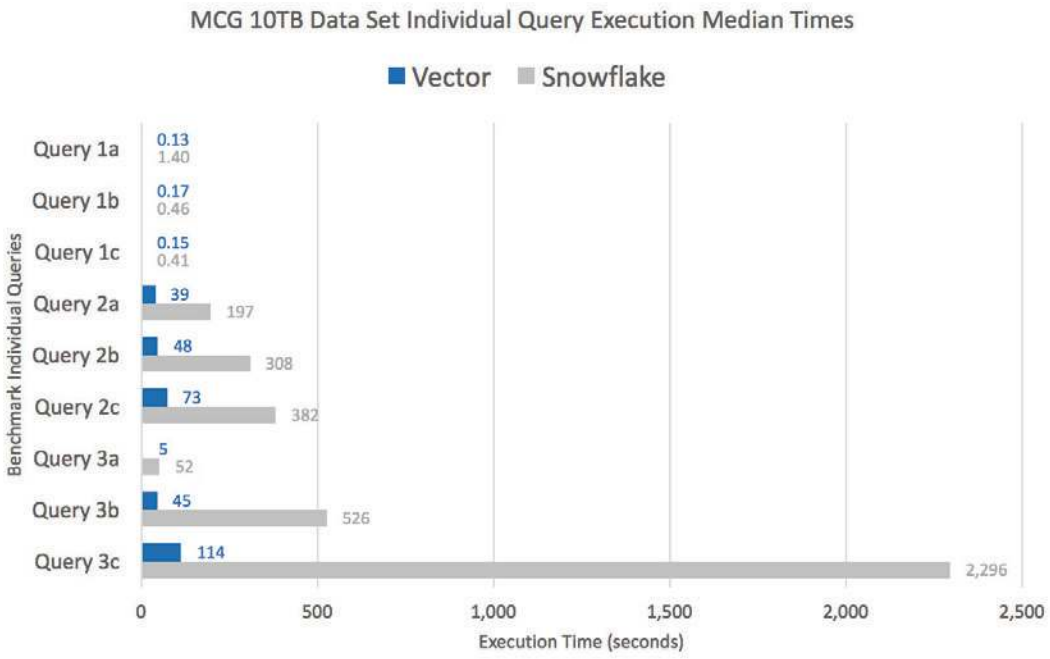
In the case of 5TB (i.e., 29 billion rows in the uservisits table) on the same 16-node cluster, Vector query response times were all faster than Snowflake. Overall, Vector was 9 times faster than Snowflake. However, the biggest gap was noticed during the Query 3 Join series. For Vector, Query 3c ran over 14 times faster. Below are the individual query results for the 5TB data set of Snowflake and Vector median query execution times, again, out of 5 trials.



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

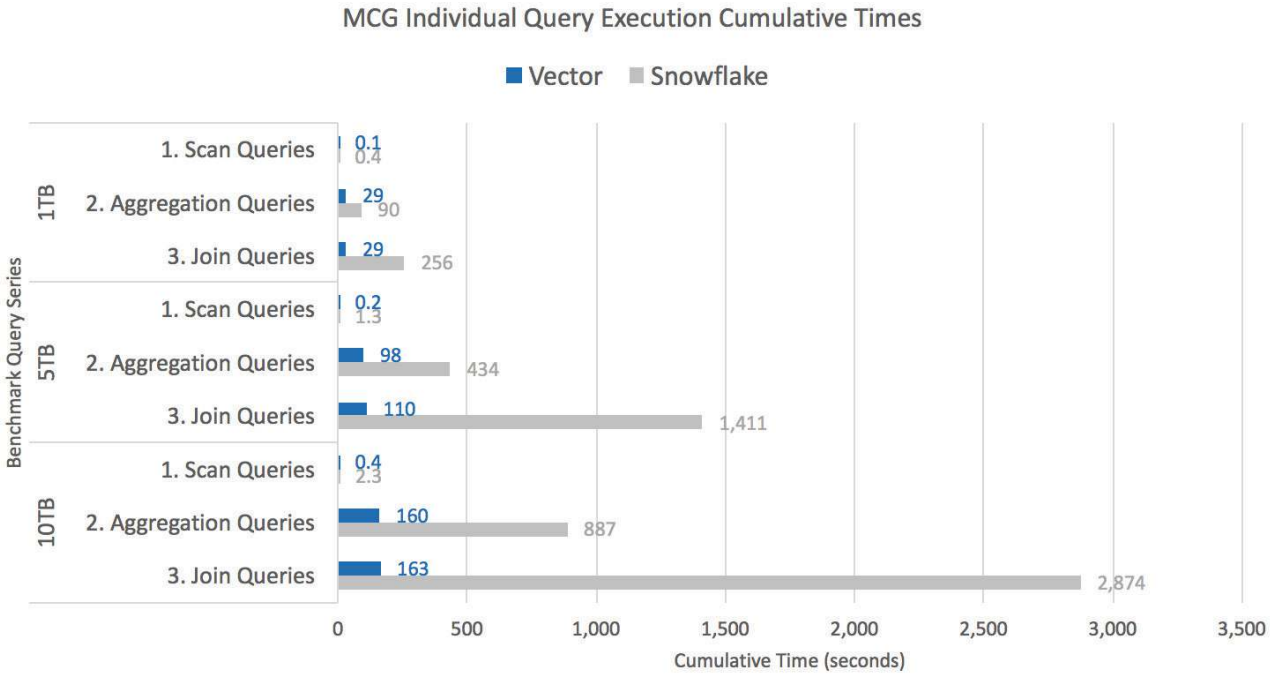
10TB Data Set

In the case of 10TB on the same 16-node cluster, Vector query response times were all faster than Snowflake. On the whole, Vector was nearly 12 times faster than Snowflake. Once again, the continued separation is seen with the Query 3 Join series. For Vector, Query 3a was 11 times faster; Query 3b finished 12 times faster; and Query 3c ran over 20 times faster.



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

Overall, the cumulative 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.

Across all data sizes and workloads, Vector was over 10 times faster than Snowflake. Once again, notice the divergence in performance between the two platforms in the execution of benchmark queries with JOIN clauses: 9x, 13x, and 18x, scaling from 1TB, 5TB, and 10TB, respectively.

Concurrency Results

Single Cluster Concurrency Tests

We also conducted the benchmark concurrency tests using a single Snowflake cluster to understand how the platforms performed under concurrency using the closest hardware configuration we feasibly could create.

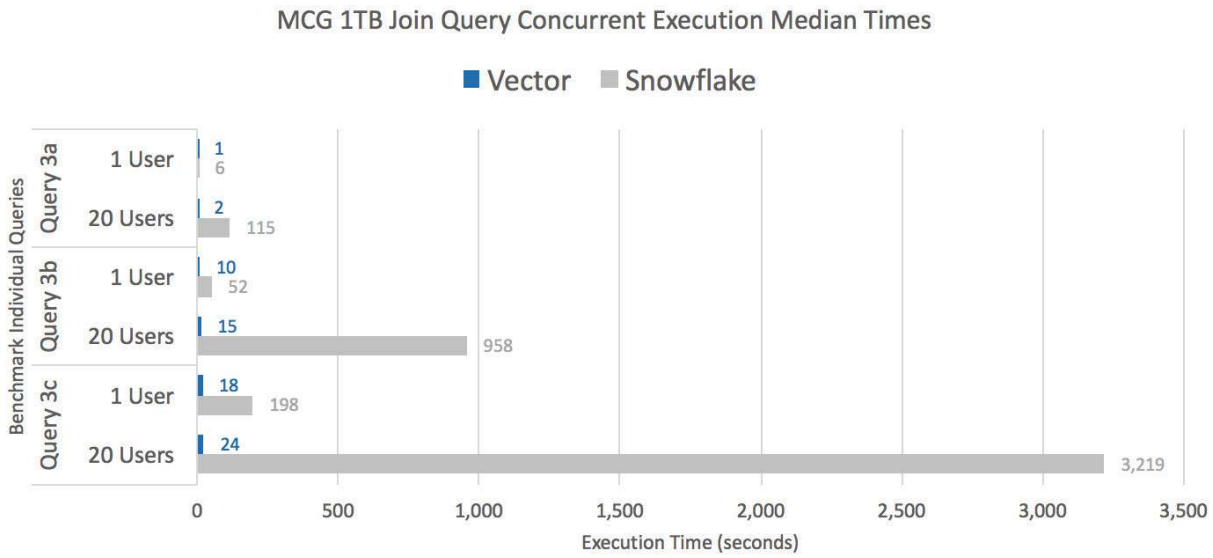
The tests were executed using a multi-threaded JDBC query test harness. The queries were executed simulating 20 concurrent users.

Both platforms were able to complete all tests at all data scales and concurrency levels.

The following tables display the median execution times (in seconds) over 5 runs of the benchmark queries executed to simulate 20 concurrent users.

1TB Data Set with 20 Concurrent Users

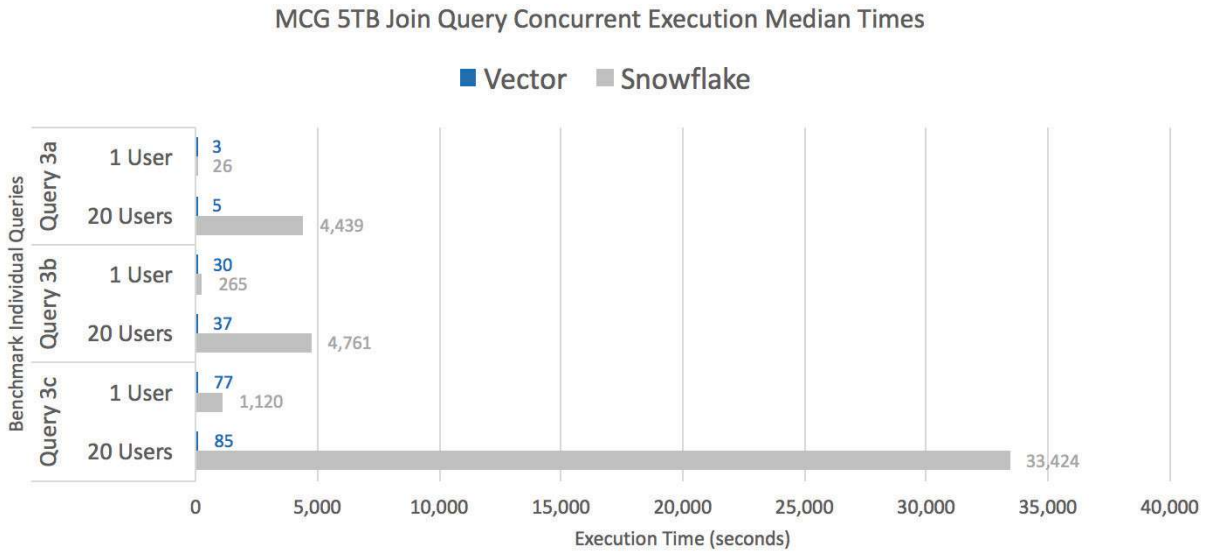
In the case of 1TB on the 16-node clusters, Vector concurrency response times were all faster than Snowflake. On the whole, Vector was 66 times faster than Snowflake. Once again, the separation is seen with the Query 3 Join series. The table below shows Join Queries (Query Set 3) results—1 user versus 20 users at 1TB. For Vector at 20 users, Queries 3a, 3b, and 3c were between 59 and 137 times faster.



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

5TB Data Set with 20 Concurrent Users

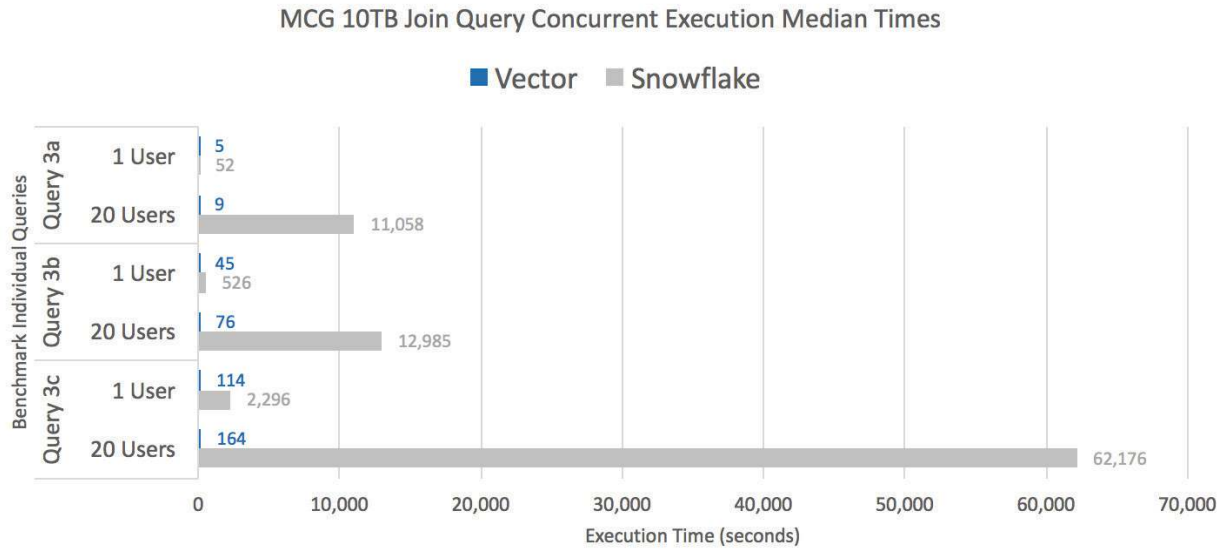
In the case of 5TB, Vector query response times for 20 users were faster than Snowflake. On the whole, the Vector queries ran many times faster than Snowflake with concurrency. The most significant difference was seen with the Query #3 Join series. For Vector at 20 users, Queries 3a, 3b, and 3c were 931, 129, and 394 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.*

10TB Data Set with 20 Concurrent Users

In the case of 10TB, Vector query response times were faster than Snowflake. For Join queries at 20 users, Query 3a was significantly faster on Vector. Query 3b was 130 times faster. Query 3c took 9 hours for Snowflake to complete at 20 users. (Vector was over 1,200 times faster.) The first table shows Join Queries (Query Set 3) results at 10TB.

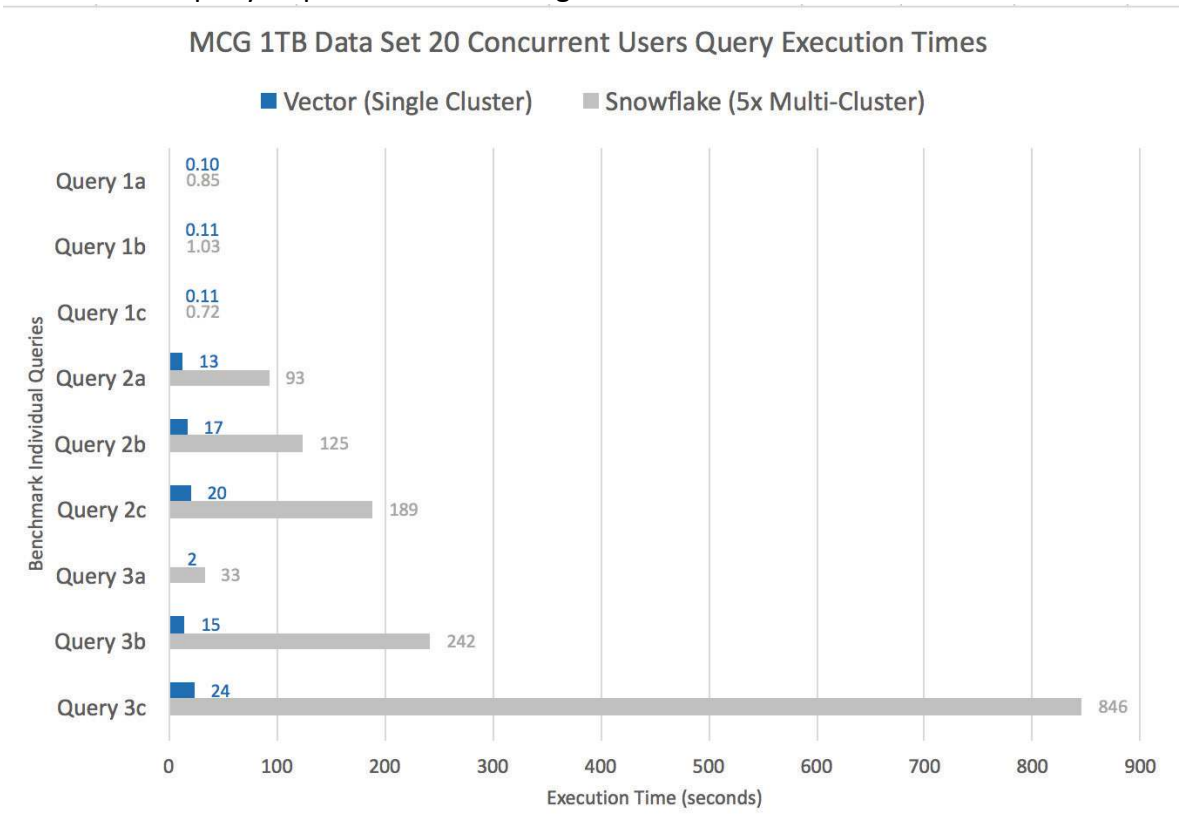


**This graph measures time to execute queries. A shorter bar indicates a faster response time.*

Snowflake Multi-Cluster

As previously noted, we conducted benchmark tests using Snowflake multi-cluster to see how it would perform. Snowflake, as a single cluster, did not run all 20 queries at the same time, so its response times were very high compared to what they are with multi-cluster enabled.

We ran the 1TB data set with a multi-cluster of 5 X-Large Snowflake warehouses and sent the 20 simultaneous query requests. The following chart shows the result:



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

In the case of 1TB, the query response times on a Vector single cluster were all faster than the Snowflake 5x multi-cluster. On the whole, Vector was 17 times faster than Snowflake. Similar to the single-user results, the continued separation is seen with the Query 3 Join series. For Vector, Query 3a and 3b were 17 times faster, and Query 3c ran 36 times faster than the Snowflake 5x multi-cluster.

The use of multi-cluster with Snowflake was not an apples-to-apples hardware configuration in terms of computing nodes or cost. In this case, 5 Snowflake clusters versus 1 Vector cluster is not equivalent. The multi-cluster of 5 X-Large Snowflake clusters cost $240 per hour to run compared to the $34.05 per hour it cost to run Vector.

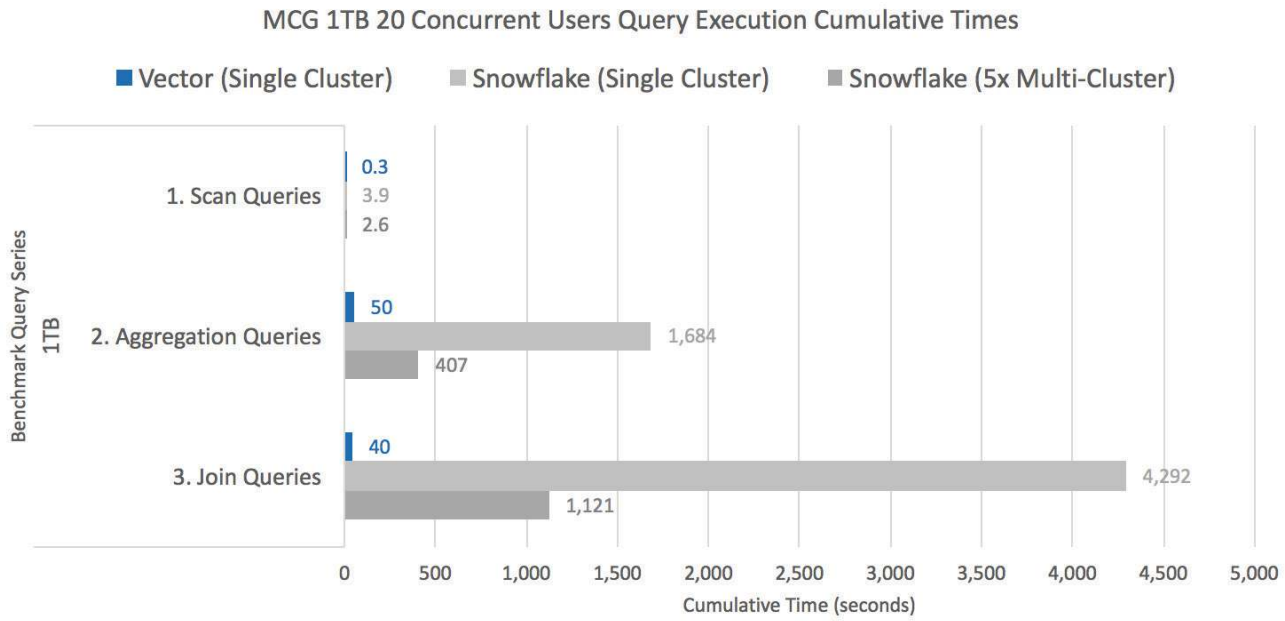
Commentary on Concurrency and Snowflake

From our testing, it appears that in a production environment with more than a handful of users, Snowflake's multi-cluster option would be necessary due to Snowflake's concurrency performance limitations of a single-cluster configuration. The activation of multi-cluster could significantly increase aggregate computing costs for an organization and should be taken into consideration. Since our goal was to design and run equitable benchmarks to simulate real-world scenarios, our concurrency results are limited in offering a full performance profile of Snowflake at scale with concurrency.

Snowflake multi-cluster acts by automatically spawning additional clusters (of the same size) in an attempt to handle concurrent query requests, and then automatically suspending them once the workloads are complete.

On the upside, you could say Snowflake automatically scales itself up when user demands are high. However, organizations would need to understand that additional resource activation comes at an additional cost. Our 16-node cluster cost $48 per hour to run on Snowflake. Upscaling our cluster to 5 clusters to run the 20 concurrent benchmark queries, cost $240 per hour—5 times more.

Snowflake did perform better in the concurrency tests with multi-clusters enabled. However, at that point, we would need an equal number of Vector clusters to make a fair comparison for benchmark purposes. Thus, to create an "apples-to-apples" comparison, we ran the tests on single clusters. Regardless, the following chart compares the single cluster configurations of both platforms with the multi-cluster configuration of Snowflake for 1TB and 20 concurrent users.

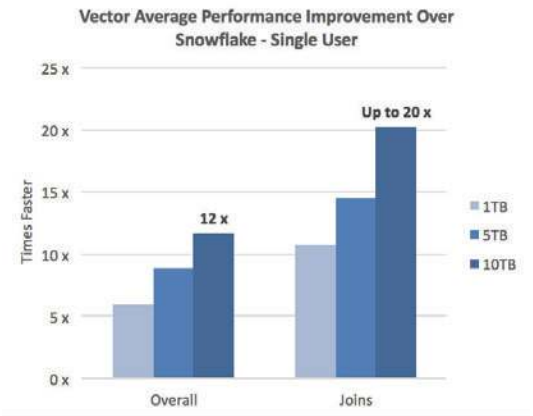


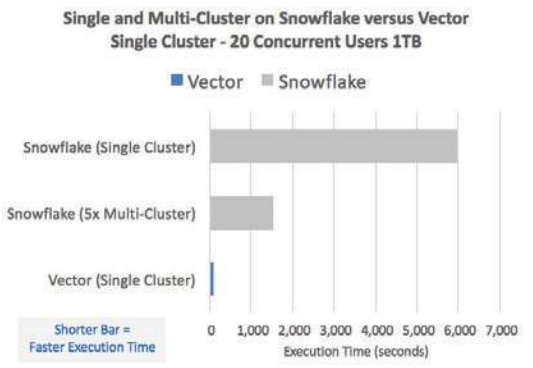
**This graph shows query execution times added together.*
A shorter bar indicates faster total response times across the workloads.

Conclusion

Cloud databases, notably on Amazon Web Services, are a way to avoid upfront 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 outperformed Snowflake when scale, and especially joins, were introduced.

Measuring execution performance of queries with increasing data volumes and concurrency, benchmark results for Actian Vector and Snowflake revealed some performance differentiators between the two products. Actian Vector performed 12 times faster overall and up to 20 times faster on queries with joins on our single-user tests. A revealing finding was with concurrency. With Snowflake's multi-cluster option enabled and 5 Snowflake clusters versus a single Vector cluster, Vector was still 17 times faster than Snowflake overall.[6]





These performance results are most likely explained by the technology underlying Vector. The basic architecture of Actian Vector is the Actian patented X100 engine, which 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-premises 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.

For more information about Actian Vector including how to get a free download, go to https://www.actian.com/analytic-database/vector-smp-analytic-database/.

[6] In 2011, Vector set a new record in a TPC-H benchmark at scale factor 100, delivering 340% higher performance than the previous best record while improving price/performance by 25%. Today Vector still leads 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 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.

More information:

- Actian Vector for SMP systems
- Actian Vector for Hadoop
- Download Actian Vector on-premise
- Actian Vector in the Amazon Marketplace
- Actian Vector in Microsoft Azure