Cloud Analytics Database Performance Report

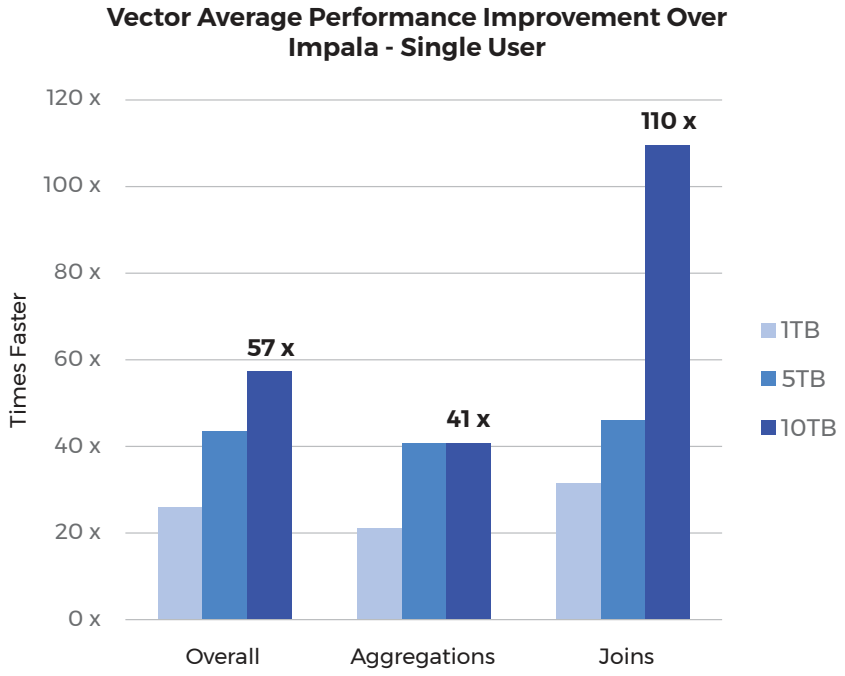
Actian Vector up to 100x faster than Apache Impala

MCG Global Services Benchmark Results

April 2018

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 full table scan queries that measure raw throughput
- Cloudera Impala could not complete aggregation and join queries with 20 simulated users when the database size was scaled to 5 and 10 TB, These queries were abandoned after taking more than 2 hours.



Check out 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 Apache Impala on Cloudera CDH

By McKnight Consulting Group Global Services

April 2018

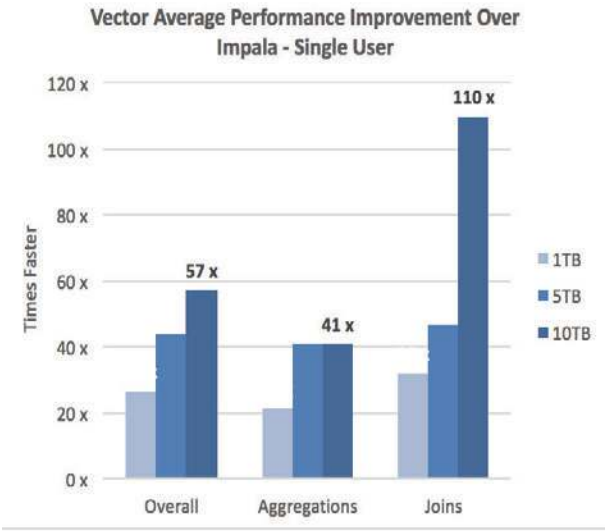
Sponsored by

Executive Overview

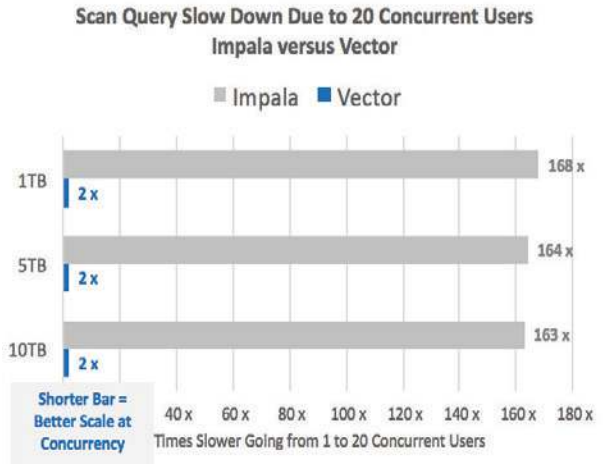
Data-driven organizations rely on analytic databases to load, store, and analyze volumes of data at high speed to derive timely insights. Data volumes within modern organizations' information ecosystems are rapidly expanding—placing significant performance demands on legacy architectures. Today, to fully 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, enterprise-ready, analytical-workload solutions on Actian Vector and Apache Impala supported by Cloudera Enterprise. 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 dataset sizes being extended to 1, 5, and 10 TB of data to simulate real-world Big Data demands. The testing was conducted using 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 Impala revealed some significant performance differentiators between the two products. Actian Vector performed up to 60 times faster overall, over 100 times faster on queries with joins, and up to 41 times faster on queries with aggregations on single-user tests.



A revealing finding emerged when we stressed the workload by simulating 20 concurrent users. On simple scan queries, Impala ran 165 times slower when 20 users ran the same query simultaneously—while Vector experienced only a 2x slow down as compared to a single user query. Also, Impala completed only 68% of the benchmark tests while Actian Vector was able to complete all portions of the benchmark suite.

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 enterprises 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 cloud-enabled relational analytical databases 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. Many companies are leveraging the cloud for portions of their information ecosystem and creating hybrid architectures of cloud and on-premises assets. 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 Apache Impala. Both relational analytical databases are based on massively parallel processing (MPP) that scale and provide high-speed analytics. Impala is available as a component of Cloudera CDH (Cloudera Distribution including Apache Hadoop) commercial stack and can be used for free as Cloudera Express. Actian 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.

	Actian Vector	*Impala*
Company	Actian	Cloudera
Released	2014	2013
Current Version	5.0	5.14
Storage	Hadoop HDFS	Hadoop HDFS
SQL	ANSI SQL 2003	Impala SQL
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.

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.[1]

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.

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.*

[1] 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 about the AMPLab BDB Data Set, see https://amplab.cs.berkeley.edu/benchmark/. 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.

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 uservisits data sets contain 982,000 rows per part. The uservisits 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 appear to be generated using a random date order. The rationale of conducting the benchmark tests employing natural date order was that this would be closer to a real world use case, as a clickstream web log database would typically 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 cluster environments—one for Actian Vector and the other for Impala—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.

Both Impala and Vector can run on any of the EC2 instance classes. Thus, we used identical EC2 instance types (for equal processor and memory capacity) and the exact same storage congfiguration to create an "apples-to-apples" comparison.

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, and disk I/O. The following is an explanation of each factor:

- **Number of cluster nodes** – According to Cloudera's sizing guide, they recommend roughly between 15 and 20 nodes[2] for our data size and concurrent user requirements. Vector recommends at least 3 nodes, so we chose 16 nodes for our clusters.
- **Processing power** – We chose the EC2 r4 family to use 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** – We also chose EC2 r4 for its DDR4 memory—common among most of EC2 the instance types.
- **Storage and disk I/O** – Since mathematically the benchmark queries "fit in memory" and the select count(*) from method of result set handling (see the next section) reduces disk I/O to a minimum, we used 1TB of Elastic Band Storage that comes with the EC2 r4 family.

In summary, the following table compares all factors considered.

Platform	Actian Vector	Impala
Version	5.0 (with the latest patch 53001 applied)	CDH 5.14
Instance Class	r4.8xlarge (dedicated, no shared tenancy)	r4.8xlarge (dedicated, no shared tenancy)
Nodes	16	16
Cluster vCPUs	512 (32 per node)	512 (32 per node)
Cluster RAM	3,904 GiB (244 GiB per node)	3,904 GiB (244 GiB per node)
Storage	16TB EBS (1TB per node)	16TB EBS (1TB per node)
Computing Cost	$34.05 per hour ($2.128 per node)	$34.05 per hour ($2.128 per node)

The database management systems were each deployed on extra-large AWS 16 compute 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 different methods. For Impala, the best way was to create an external table using the S3 data:

[2] Referencing the cluster size estimation table at the above hyperlink, we used Cloudera's recommendations of 1TB with 100 users (15 nodes) and 15TB with 10 users (20 nodes) to arrive at this node count for our benchmark.

```
CREATE EXTERNAL TABLE s3rankings (pageURL varchar(300), pageRank int,
avgDuration int)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION 's3a://mcg-actian-benchmark/1TB/rankings/';
```

Then we inserted the data from the external tables into physical tables on Impala:

```
insert into rankings select pageURL, pageRank, avgDuration from s3rankings;
```

Once the data was loaded, in Impala, we generated statistics for the data using the following SQL command, which is consistent with the product documentation:

```
compute stats %t;
```

where *%t* is the name of the table.

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[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
```

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 Impala, partitioning is used only when certain criteria are met. According to Cloudera documentation, Impala data should be partitioned when the following conditions are true:

- Tables are very large (TRUE for our benchmark)
- Tables are often/always queried with conditions on certain columns (TRUE)
- Columns have reasonable cardinality (FALSE, because pageURL and destinationURL have over 250 million unique values at 1TB)
- Data are loaded from an ETL pipeline (NOT APPLICABLE)

Thus, we did not partition the Impala data due to the high cardinality of the URL data.[5]

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.

[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.

[5] We attempted to load the data into Impala with partitions anyway, and the load operations failed on Impala.

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). We wanted I/O to impact the benchmark results as little as possible.

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 set. Thus, Query Sets 1 and 2 (see below) were encapsulated with the following:

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

where *%q* was the query itself.

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 3 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 scanning the table and performing 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;`

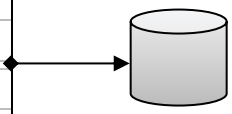
Concurrency Test Harness

The final objective of the benchmark was to demonstrate Vector and Impala 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.

Impala has a feature known as admission control to impose limits on concurrent queries. The idea is to avoid resource usage overruns and out-of-memory conditions on busy clusters. Admission control acts by holding certain query requests in a queue until there are enough resources to run them. For the purposes of our benchmark, we wanted to avoid admission control, because queued queries would negatively skew the execution results and not demonstrate the true ability of the platform to handle concurrency.

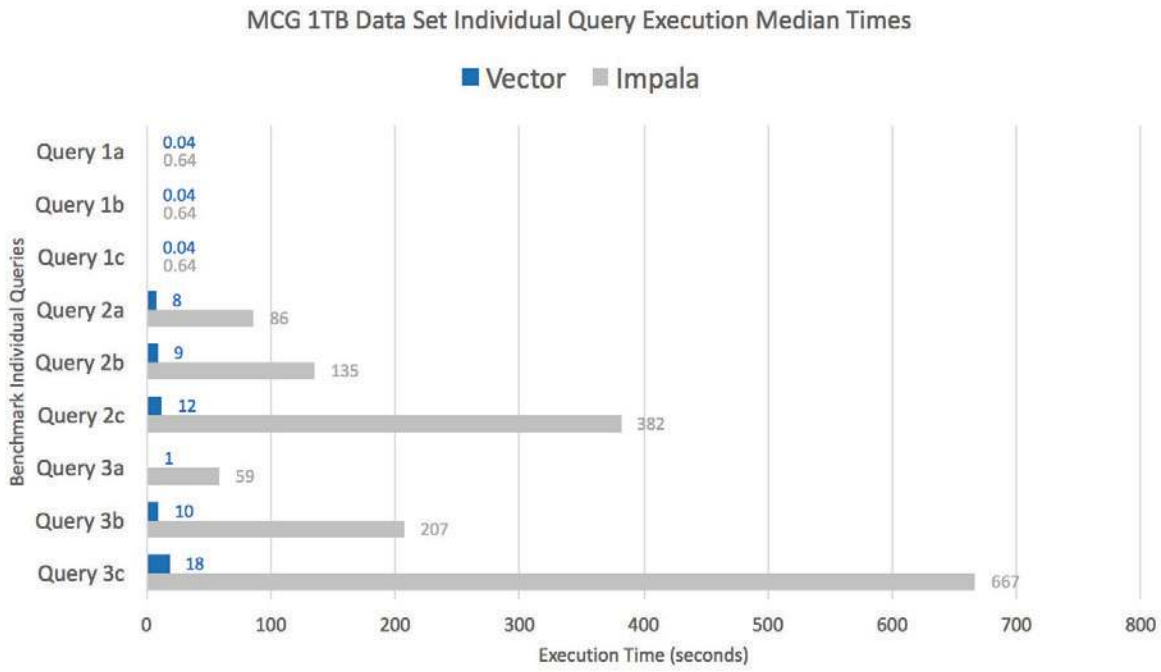
Benchmark Results

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

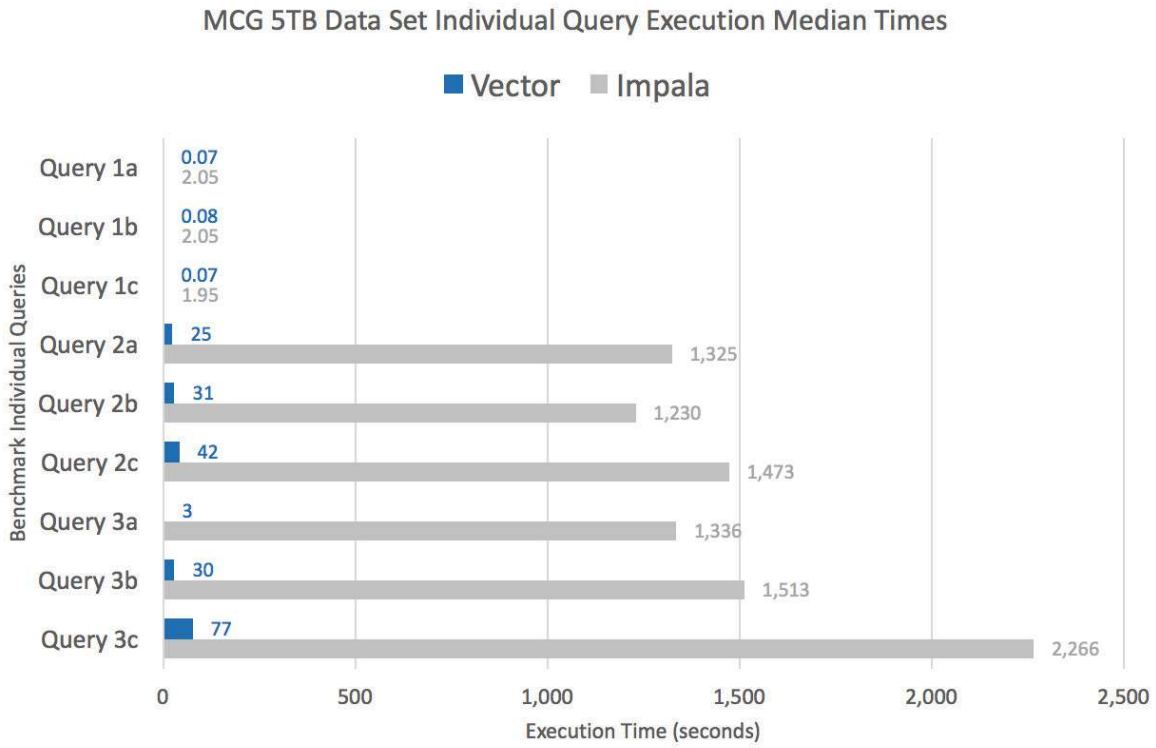
In the case of the extended 1TB data set on a 16-node cluster, Vector query response times were all faster than Impala. Overall, Vector was 27 times faster than Impala. However, the biggest gap appeared during the Query 3 Join series. For Vector, Query 3a ran 50 times faster—followed by 3b and 3c running 21 and 36 times faster, respectively. Below are the individual query results for the 1TB data set of Impala 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

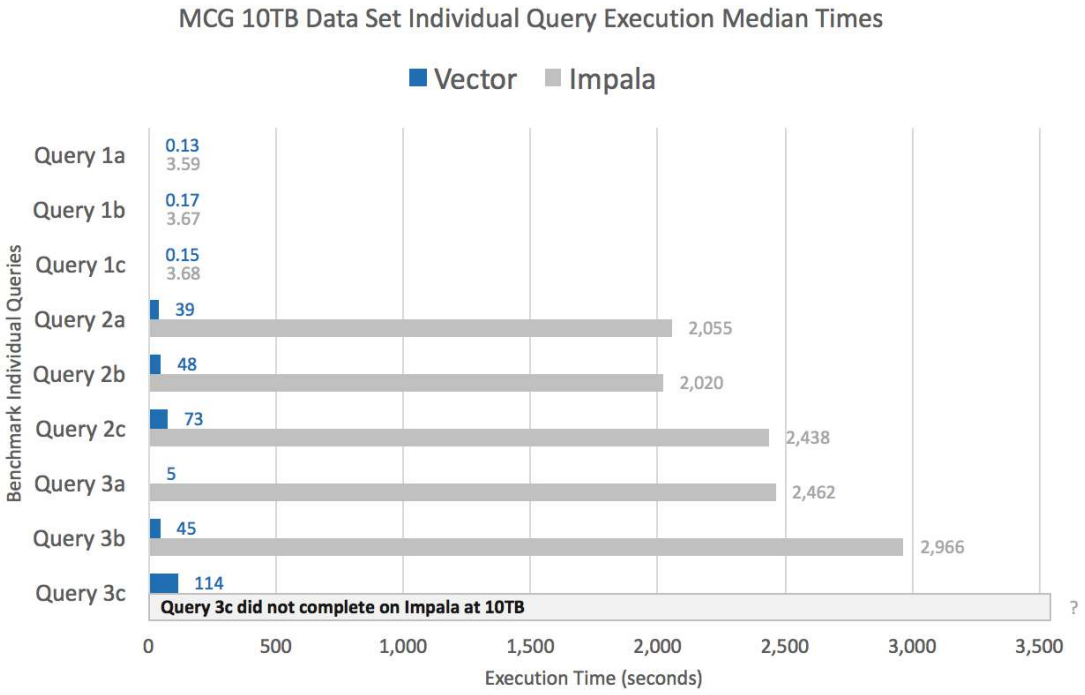
In the case of 5TB (i.e., 29 billion rows in the uservisits table) on the same 16-node cluster, Vector query response times were all faster than Impala. Overall, Vector was 44 times faster than Impala. Again, the biggest gap was noticed during the Query 3 Join series. For Vector, Query 3a ran over 500 times faster. Below are the individual query results for the 5TB data set of Impala 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.*

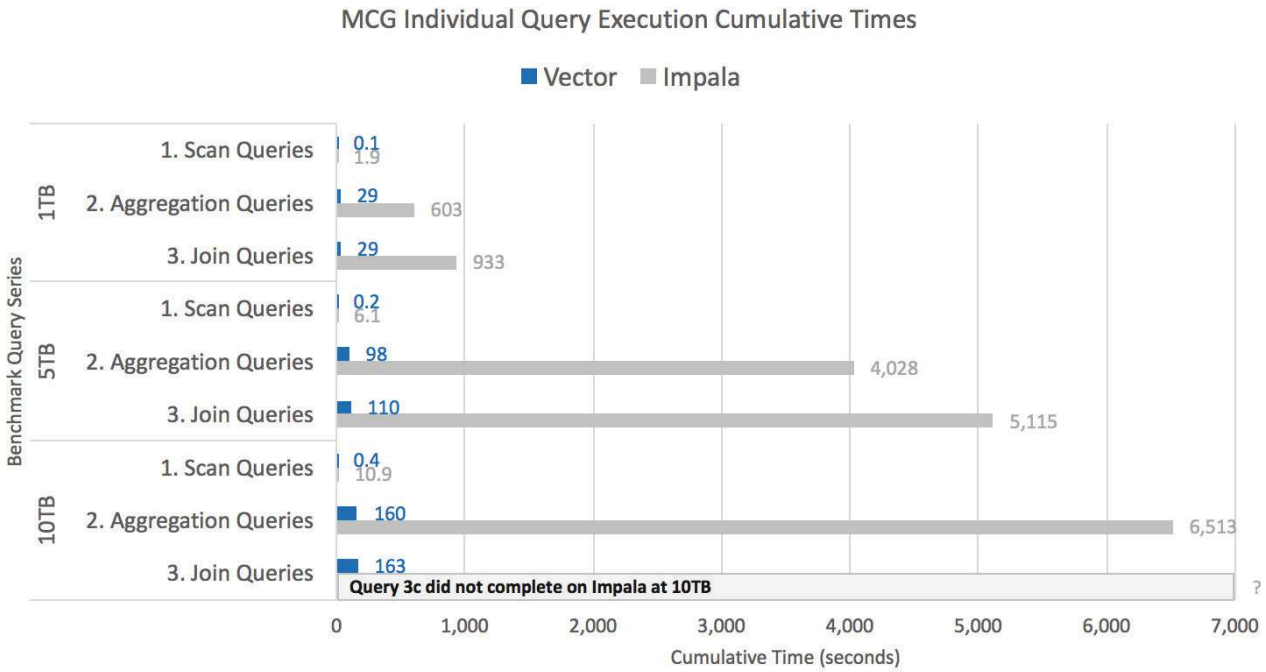
10TB Data Set

In the case of 10TB on the same 16-node cluster, Vector query response times were all faster than Impala. On the whole, Vector was 37 times faster than Impala. Once again, the continued separation is seen with the Query 2 Aggregation and Query 3 Join series. For Vector, queries 2a, 2b, and 2c were 53, 42, and 33 times faster, respectively. Also for Vector, Query 3a was 500 times faster, and Query 3b finished 66 times faster. Query 3c did not complete on Impala at 10TB.



**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 for the single-user tests, Vector was over 39 times faster than Impala.

Concurrency Results

As previously noted, we conducted the benchmark tests using a driver to simulate concurrency of 20 users to see how both platforms would perform. Impala struggled with concurrency, and did not complete many of the concurrency tests. We discuss the behavior we experienced after the results.

Vector was able to complete all tests at all data scale and concurrency levels.

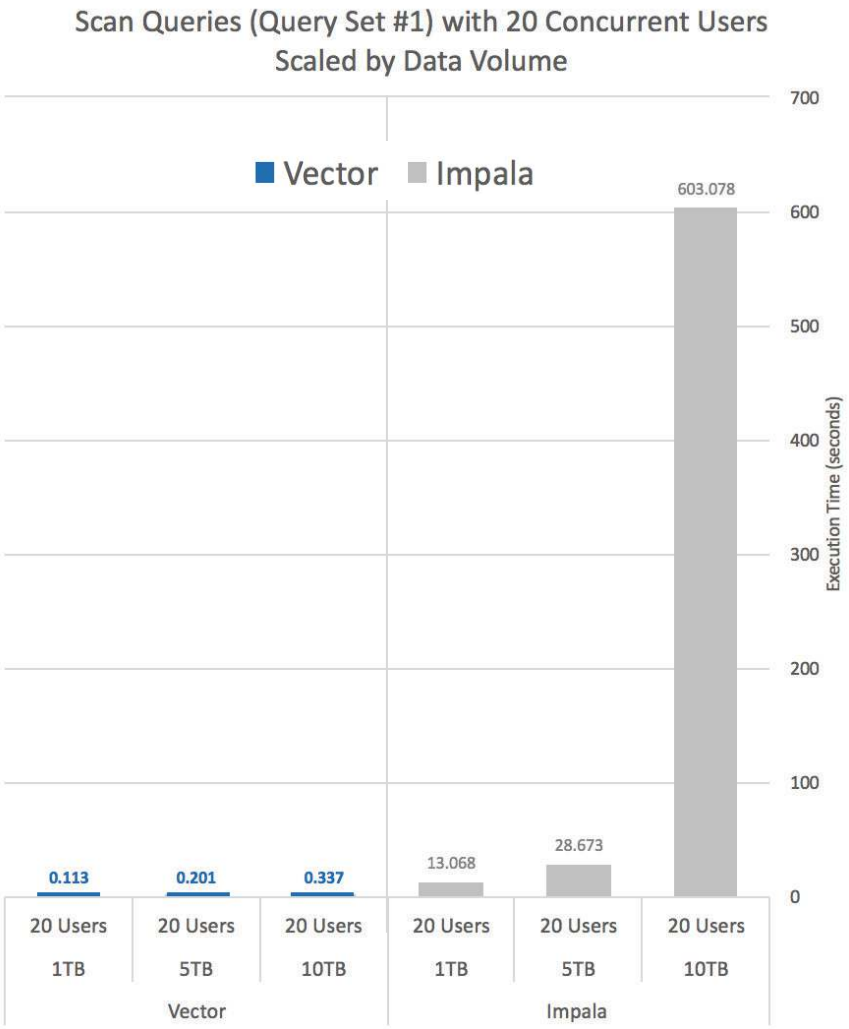
The following tables display the median execution times (in seconds) over 5 runs of the benchmark queries executed to simulate 20 concurrent users.

Scan Query Set 1 with 20 Concurrent Users

In the case of the Scan Query Series 1 on the 16-node clusters with 20 users, Vector concurrency response times were all faster than Impala.

Vector completed all the queries in less than a second. Impala execution times became exponentially longer with larger data volumes.

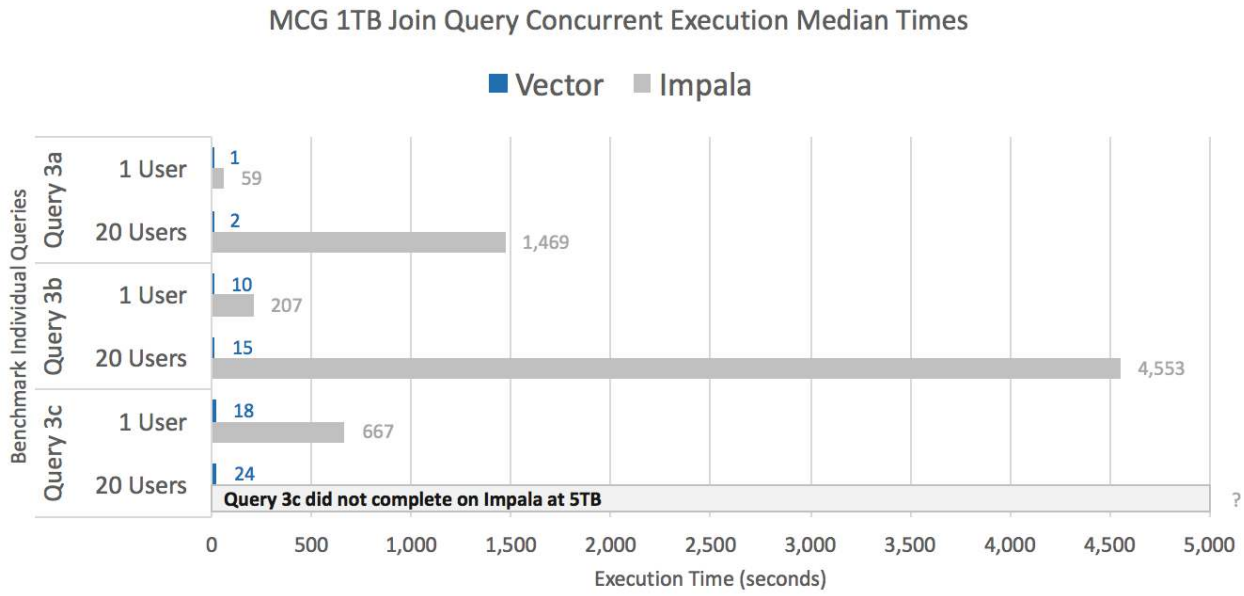
NOTE: The 1TB and 5TB concurrency tests were executed on Impala to simulate 4 queries sent to 5 nodes simultaneously (20 total) to take scratch space pressure off a single node. For the 10TB test, we utilized the 16TB scratch drive for a 20 queries sent to 1 node test.



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

Join Query Set 3 with 20 Concurrent Users

In the case of Join queries, Vector query response times for 20 users were faster than Impala. For Vector at 20 users, Queries 3a and 3b were over 700 and 300 times faster, respectively. Query 3c did not complete on Impala at 1TB. Also, Impala did not complete any of Query Set 3 at 5TB or 10TB. The following table shows Join Queries (Query Set 3) results using the 1TB data set:



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

Queries Not Completing on Impala

Many of the concurrent queries benchmark tests did not complete on Impala. None of the Aggregation (Query Set 2) queries completed at any data size on Impala.

Observing the nature of these incomplete query runs revealed a confluence of issues. First, disk spilling contributed to the underlying problem. For instance, the concurrent tests of aggregation (Query Set 2) saw disk spilling as high as 65GB per query. With 20 concurrent queries running, the total disk spillage was 1.3TB. This put enormous pressure on Impala.

Second, Impala uses temporary disk space during intensive query runs called scratch space. According to Cloudera, the scratch space must not be part of the HDFS but rather the local filesystem. Since the scratch space is not distributed throughout the cluster, the node to which we sent the query would have the entire scratch directory. During concurrent query execution, the scratch space would reach the capacity of the local file system, and the queries would fail.

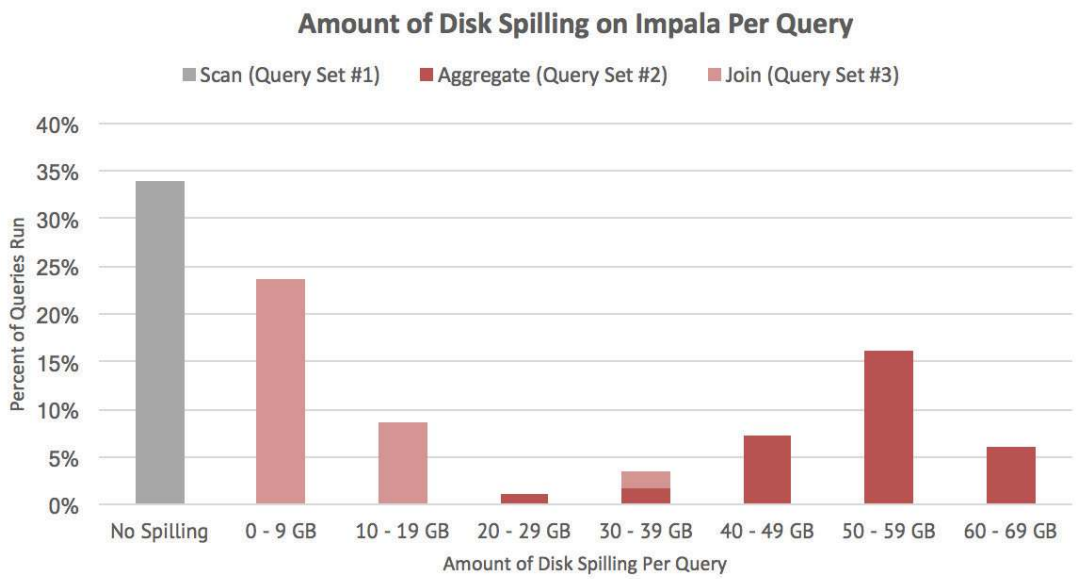
We compensated for this by attaching a 16TB disk to the node receiving the query requests to use it as scratch space. However, many of the concurrent queries would not complete within 2 hours, which was our allowance.

The table below shows which queries would not complete on Impala due to query failure or the execution time exceeding 2 hours. Impala completed 68% of the benchmark tests. All benchmark tests completed on Vector.

		1 User		20 Users	
Data Size	**Query**	Vector	Impala	Vector	Impala
1TB	**Query 1a**	0.04	0.64	0.10	3.88
	Query 1b	0.04	0.64	0.11	2.94
	Query 1c	0.04	0.64	0.11	13.07
	Query 2a	7.58	86.22	12.88	*Did Not Complete*
	Query 2b	9.11	135.28	16.99	*Did Not Complete*
	Query 2c	12.01	381.61	20.22	*Did Not Complete*
	Query 3a	1.16	58.71	1.94	1,468.71
	Query 3b	9.77	207.39	14.56	4,553.28
	Query 3c	18.33	666.95	23.51	*Did Not Complete*
5TB	**Query 1a**	0.07	2.05	0.15	26.15
	Query 1b	0.08	2.05	0.20	27.03
	Query 1c	0.07	1.95	0.19	28.67
	Query 2a	24.81	1,324.88	35.05	*Did Not Complete*
	Query 2b	30.85	1,229.76	49.42	*Did Not Complete*
	Query 2c	42.29	1,473.06	56.94	*Did Not Complete*
	Query 3a	3.38	1,335.95	4.77	*Did Not Complete*
	Query 3b	29.85	1,513.15	36.86	*Did Not Complete*
	Query 3c	76.98	2,265.95	84.90	*Did Not Complete*
10TB	**Query 1a**	0.13	3.59	0.24	603.08
	Query 1b	0.17	3.67	0.34	602.16
	Query 1c	0.15	3.68	0.31	599.29
	Query 2a	38.86	2,055.06	59.16	*Did Not Complete*
	Query 2b	47.90	2,020.01	77.92	*Did Not Complete*
	Query 2c	72.99	2,437.77	120.59	*Did Not Complete*
	Query 3a	4.85	2,461.89	9.02	*Did Not Complete*
	Query 3b	44.67	2,966.18	75.61	*Did Not Complete*
	Query 3c	113.77	*Did Not Complete*	164.33	*Did Not Complete*

Impala Disk Spilling

The factor that contributed the most for the long execution times on Impala (and the failure of the concurrency jobs to fully complete) were due to disk spilling that occurred—especially in the larger data sets and more complex queries (sets2 and 3). For example, we saw Impala spill as much as 65GB to disk per query for the Aggregation queries (query set 2). The following chart details the disk spilling we experienced on the Impala cluster.



**This graph reflects only Impala. Vector did not experience disk spilling on any of the benchmark tests.*

Reviewing Cloudera's documentation about disk spilling revealed the factors that caused Impala to spill to disk. The following table lists Impala's disk spilling triggers and identifies the queries that met those conditions:

Impala may spill to disk when…	Scan (Query Set 1)	Aggregation (Query Set 2)	Join (Query Set 3)
A query uses a GROUP BY clause for columns with millions of distinct values		✓	
Large tables are joined together			✓
A large result set is sorted by the ORDER BY clause			✓
The DISTINCT and UNION operators build in-memory structures for unique values			

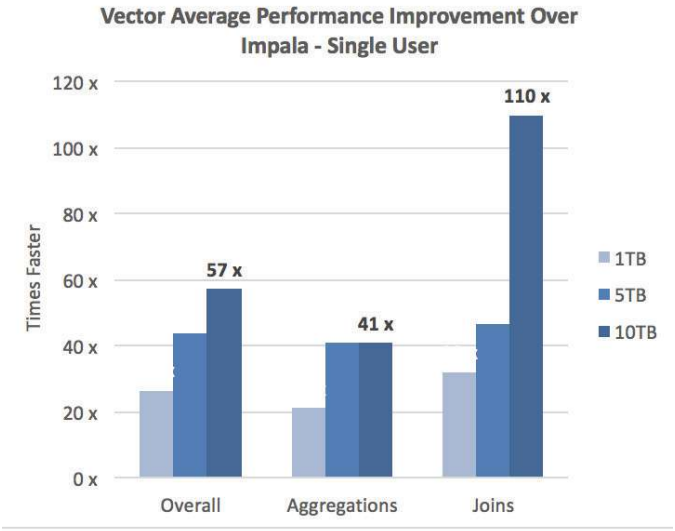
These are the most likely causes for the extensive disk spilling we experienced with Impala.

Vector experienced no disk spilling on any of the benchmark tests.

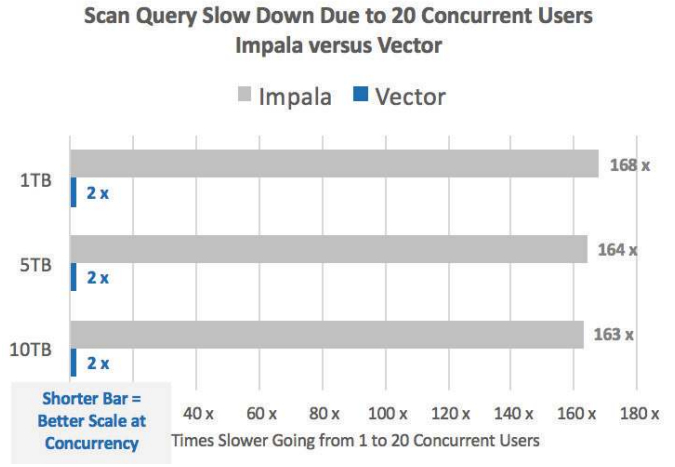
Conclusion

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 significantly outperformed Impala when scale, and especially joins, were introduced.

Measuring execution performance of queries with increasing data volumes and concurrency, benchmark results for Actian Vector and Impala revealed some performance differentiators between the two products. **Actian Vector performed up to 60 times faster overall, over 100 times faster on queries with joins, and up to 41 times faster on queries with aggregations on single user tests.**



A revealing finding was observed when we stressed the workload by simulating 20 concurrent users. On simple scan queries, Impala ran 165 times slower when 20 users ran the same query simultaneously—while Vector only experienced a 2x slow down.[6] Additionally, Impala fully completed only 68% of the benchmark tests.



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 data processing 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 from conventional "one-row-at-a-time

[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.

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/.

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.

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