The Business Intelligence for Hadoop Benchmark

Q4 2016

atscale

Table of Contents

atscale

Hadoop as an Analytics Platform

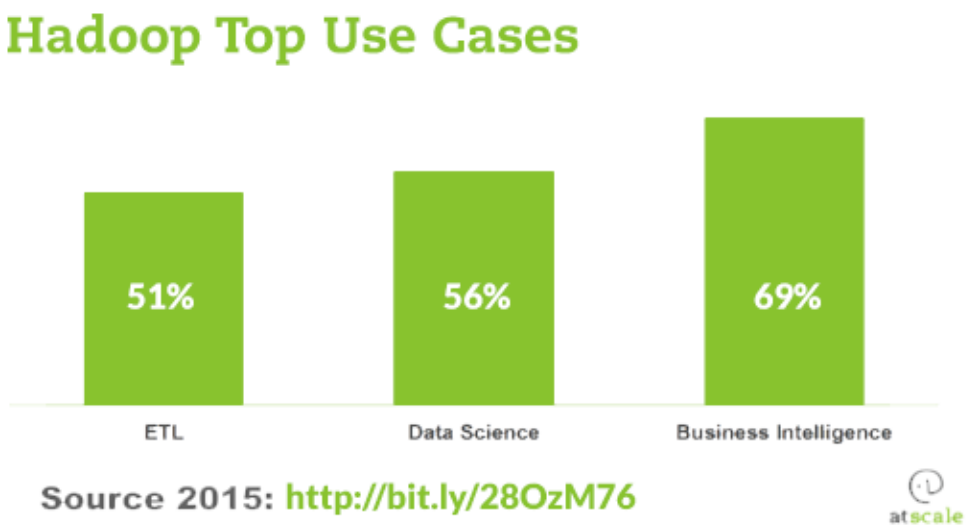
The past decade has brought significant change to the way organizations store and process data. Much of this change is due to the innovation, adoption, and commercialization of Hadoop. The limitations of traditional data infrastructures have led many organizations to move to Hadoop for not only new data use cases, but for day-to-day operational workloads as well.

As Hadoop matures, enterprises are starting to use this powerful platform to serve more diverse workloads. Hadoop is no longer just a batch-processing platform for data science and machine learning use cases – it has evolved into a multi-purpose data platform for operational reporting, exploratory analysis, and real-time decision support.

With the ongoing innovation of the SQL-on-Hadoop and in-memory data processing engines, Hadoop is now able to serve business-critical workloads in production. Hadoop is now ready to be the data source for business intelligence (BI) and online analytical processing (OLAP) workloads. According to the 2015 Hadoop Maturity Survey conducted by AtScale, Cloudera, Hortonworks, MapR, and Tableau, Business Intelligence (BI) is the top use case enterprises plan to migrate to Hadoop.

As we learned from the First Edition of our BI on Hadoop Benchmarks, the performance and maturity of the top SQL-on-Hadoop engines - Impala, Hive, and SparkSQL - make them all suitable candidates to support business intelligence workloads. When combined with the continued emergence of BI as the top workload for Hadoop adopters, it becomes clear that the adoption use of SQL-on-Hadoop engines to enable BI is a trend that will only continue to grow.

Forrester predicts that 100% of enterprises will adopt Hadoop in the next 24 months.



In fact, according to the 2015 Hadoop Maturity Survey conducted by AtScale, Cloudera, Hortonworks, Mapr, and Tableau, the top Hadoop use case for enterprises is **Business Intelligence.**

Based on this trend, coupled with the positive community reception to our first round of benchmarks, we conducted a second round of benchmark testing with the purpose of evaluating the progress of existing engines as well as the evaluation of new entrants in the SQL-on-Hadoop space. This document provides a deep look into both the methodology and results of the Second Edition of the BI on Hadoop Performance Benchmarks.

Executive Summary: Key Findings

The goal of this benchmark is to help technology evaluators select the best SQL-on-Hadoop technology for their use cases. Key findings include:

- **SQL-on-Hadoop engines are well suited for Business Intelligence (BI):** All tested engines - Hive, Impala, Presto, and Spark SQL - successfully executed all of the queries in our benchmark suite and are stable enough to support business intelligence workloads.

- **There is no single "best engine":** We continue to see different engines shine in different areas. Depending on raw data size, query complexity, and the target number of end-users enterprises will find that each engine has its own 'sweet spot'.

- **Version-to-version improvements are significant:** The open source community continues to drive significant and rapid improvements across the board. All engines tested showed between 2x and 4x performance gains in the six months between the first and second edition of the benchmarks. This is great news for those enterprises deploying BI workloads to Hadoop.

- **Small vs. Big Data:** Impala and Spark SQL continue to shine for small data queries (queries against small data sets). New in this edition, the latest release of Hive LLAP (Live Long and Process) shows suitable "small data" query response times. Presto also shows promise on small, interactive queries.

- **Few vs. Many Users:** Impala continues to shine in terms of concurrent query performance, and at the same time Hive and SparkSQL show significant improvements in this category. Presto, new to this edition of the benchmarks, shows the best results in concurrency testing.

The majority of this document describes the details of the testing methodology, datasets, and queries used to complete this benchmark. A more technical summary of the benchmark results as well configuration details can be found at the end of this paper be found at the end of this paper.

The BI Benchmark Evaluation Framework

The AtScale BI on Hadoop Benchmarks consider the following core elements when evaluating how well SQL-on-Hadoop engines satisfy Business Intelligence workloads:

- **Performs on Big Data:** the SQL-on-Hadoop engine must be able to consistently analyze billions or trillions of rows of data without generating errors and with response times on the order of 10s or 100s of seconds.

- **Fast on Small Data:** the engine needs to deliver interactive performance on known query patterns return results in no greater than several seconds on small data sets (on the order of thousands or millions of rows).

- **Stable for many Users:** Enterprise BI user bases consist of hundreds or thousands of data workers, and as a result the underlying SQL-on-Hadoop engine must perform reliably under highly concurrent analysis workloads.

We believe that the above three criteria are representative of the primary requirements that enterprises across industries including financial services, healthcare, retail, telecommunication and healthcare will have to meet in order to successfully execute BI workloads on Hadoop.

To be clear, there are other aspects of SQL-on-Hadoop engines that can and have been evaluated in other studies. Breadth of SQL syntax support and engine performance for Data Science queries remain relevant evaluation criteria for other SQL-on-Hadoop workloads. That said, the AtScale Business Intelligence benchmark focuses on traditional OLAP-style (On-Line Analytical Processing) queries that make extensive use of aggregation functions, GROUP BYs and WHERE clauses.

Benchmark Data Set

AtScale used the Star Schema Benchmark (SSB) data set to perform tests in this benchmark study. This data set is described in greater detail here: http://www.cs.umb.edu/~poneil/StarSchemaB.PDF.

This benchmark data set is based on the widely-used TPCH data set and has been modified to more accurately represent a data layout (in the form of a star schema) that is common for business intelligence workloads.

We used a large scale version of the SSB data set, with a focus on testing queries across large tables: the LINEORDER table contains close to 6 billion rows. Additionally, because big data sets on Hadoop often include dimension tables with very high cardinality (an architecturally weak spot for traditional Multidimensional-OLAP - or MOLAP - solutions) we also expanded the size of the CUSTOMER table to over 1 billion rows. The row counts for each of the tables used in the BI benchmarks are shown in Table 1 below.

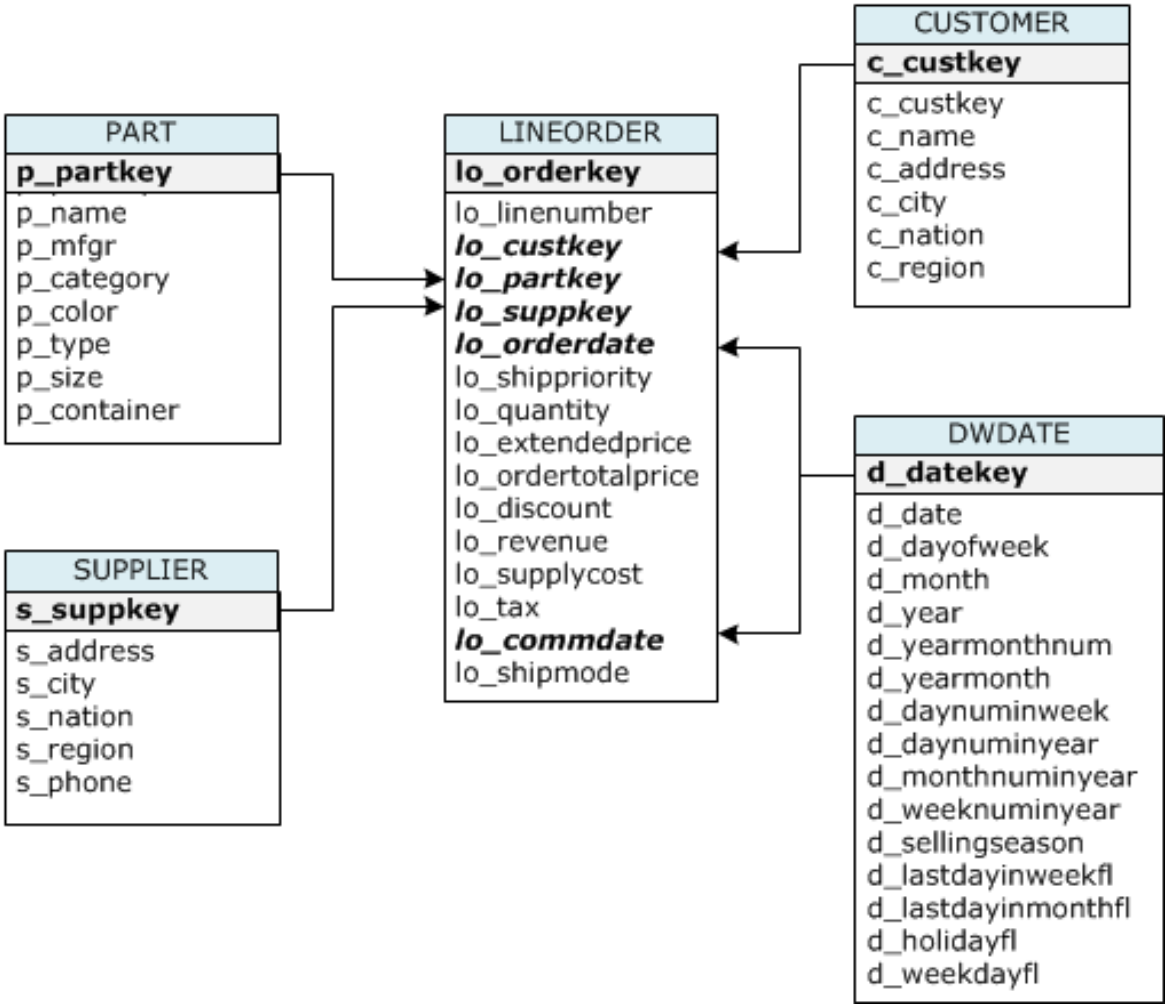
Table 1: Star Schema Benchmark (SSB) Table Details

Table Name	Number of Rows	Notes
CUSTOMER	1,050,000,000	Expanded customer dimension to test large joins
LINEORDER	5,999,989,709	
SUPPLIER	2,000,000	
PART	2,000,000	
DWDATE	16,799	

The data layout for the SSB schema and the relationships between the key tables from the benchmark are shown in Figure 1 on the following page.

Benchmark Data Set (continued)

Figure 1: Star Schema Benchmark (SSB) Table Details



Business Intelligence Query Types

In order to truly simulate a Business Intelligence enterprise environment, 13 queries were tested. The queries used for this benchmark can be summarized into a several higher level query patterns, spanning from small dataset complexity to high data volume and sophistication:

- **Q1.1 - Q1.3 - "Quick Metric" queries,** which compute a particular metric value for a period of time. These queries have a small number of JOIN and minimal/no GROUP BY operations.
- **Q2.1 - Q2.3 - "Product Insight" queries,** which compute a metric (or several metrics) aggregated against a set of product and date dimension attributes. These queries include medium-sized JOIN operations and a small number of GROUP BY operations.
- **Q3.1 - Q4.3 - "Customer Insight" queries,** which compute a metric (or several metrics) aggregated against a set of product, customer, and date based dimensions. These queries include medium and large sized JOIN operations as well as many GROUP BY operations.

During benchmark testing all 13 queries were executed twice; once each in two modes. Mode 1 was against the large data set and Mode 2 against the tables comprising AtScale's Adaptive Cache™ (aggregated data generated by the AtScale Engine based on the set of 13 queries issued). This benchmark refers to the AtScale's Adaptive Cache tables as the aggregated data. All queries used in the benchmark were produced by the AtScale Engine based on Tableau Queries. The only differences between the queries between engines were semantic. To find out more about the AtScale semantic layer and query engine, go here.

Table 2: Benchmark Query Characteristics

Query ID	Number of Joins	Largest Join Table	Number of Group Bys	Number of Filters	Comments
Q1.1	1	16,799	0	3	1 range condition, 1 comparative filter condition directly on LINEORDER table
Q1.2	1	16,799	0	3	2 range filter conditions directly on LINEORDER table
Q1.3	1	16,799	0	4	2 range filter conditions directly on LINEORDER table, 2 conditions on joined table
Q2.1	3	2,000,000	2	2	filter on p_category (less selective)
Q2.2	3	2,000,000	2	2	filter on p_brand, 2 values (more selective)
Q2.3	3	2,000,000	2	2	filter on p_brand, 1 value (most selective)
Q3.1	3	1,050,000,000	3	3	filter on region (less selective)
Q3.2	3	1,050,000,000	3	3	filter on nation (more selective)
Q3.3	3	1,050,000,000	3	3	filter on city (most selective)
Q3.4	3	1,050,000,000	3	3	filter on city (most selective) and month (vs. year)
Q4.1	4	1,050,000,000	2	2	
Q4.2	4	1,050,000,000	3	3	includes filter on year (more selective)
Q4.3	4	1,050,000,000	3	3	includes filter on year and nation (most selective)

Overall Engine Improvement

The AtScale architecture is rooted in a belief that the relational query engines that have been under development for many years - including Impala, Hive, Spark SQL, and Presto - will continue to make improvements at a pace that will outperform what an individual proprietary approach could achieve. We are pleased to find validation of this philosophy in this Second Edition of our BI on Hadoop Benchmarks.

As shown in Figure 2, all three SQL engines that were evaluated in both the First and Second Editions of the BI on Hadoop Benchmarks, within a period of 6 months, showed significant performance gains.

Figure 2: Improvements Between SQL-on-Hadoop Engine Versions

From Hive 1.2 to Hive 2.1

- Performance on large queries improved, on average, more than 340%. Of the engines tested, the version-to-version improvements we saw with Hive 2.1 with LLAP were the most significant.
- Small query performance improved over 2X, making Hive with LLAP a much more attractive engine for both large and small query patterns.
- The LLAP persistent daemon was the cause of the improvement for Hive. The ability for Hive to offload some of the query processing, cache tables and metadata and to off-load specific I/O tasks to a persistent daemon all contributed to the performance improvements.

Overall Engine Improvement (continued)

From Spark 1.6 to Spark 2.0

- Large query performance improved, on average, by a factor of 2.4X. These gains were due to the whole stage codegen and parquet reader improvements implemented in Spark 2.0.
- Small query performance remained relatively the same, consistent with previously tested excellent performance on small tables.
- Concurrency also was helped by the whole stage codegen and parquet reader improvements.

From Impala 2.3 to Impala 2.6

- Large query performance improved, on average, by a factor of 2.8X. These gains were due to implementation of run time filtering and improvements in the parquet reader.
- Small query performance remained relatively the same, consistent with previously tested excellent performance on small tables.
- Concurrency continued to be a strong point for Impala, consistent with previous evaluations.

Large Data Set Results and Observations

The table and chart below show the relative performance of Impala, Spark SQL, and Hive for our 13 benchmark queries against the 6 Billion row LINEORDERS table.

Table 3: Benchmark Query Results for Large Tables

Query	Query Execution Time (in seconds)			
	Impala 2.6	Spark 2.0	Hive 2.1 (LLAP)	Presto 0.152
Q1.1	5.6	4.8	10.5	10.3
Q1.2	5.0	3.8	9.1	8.7
Q1.3	5.6	3.3	9.3	7.9
Q2.1	8.0	11.8	10.0	12.4
Q2.2	6.2	11.0	9.3	10.4
Q2.3	6.0	10.6	9.3	9.8
Q3.1	12.6	15.5	37.5	35.5
Q3.2	15.5	15.6	38.1	39.9
Q3.3	8.0	9.2	28.1	24.9
Q3.4	7.4	6.8	28.7	15.7
Q4.1	97.3	64.9	137.6	134.4
Q4.2	49.1	30.3	131.9	123.8
Q4.3	26.8	13.2	137.9	135.5

**Note: fastest query time for each row is highlighted in green*

Based on the results of the Large Table Benchmarks shown in Figure 3, there are several key observations. Figure 3 represents the results graphically for visual analysis.

Large Data Set Results and Observations (continued)

- **No single SQL-on-Hadoop engine is best for ALL queries.** Consistent with the findings in the First Edition of the BI on Hadoop Benchmarks, Spark SQL and Impala tend to be faster than Hive. For many queries, the performance difference between Impala and Spark SQL is relatively small. Presto 0.152, a newcomer to the Second Edition of the BI on Hadoop Benchmarks, shows a performance profile that is very similar to that of Hive 2.1.

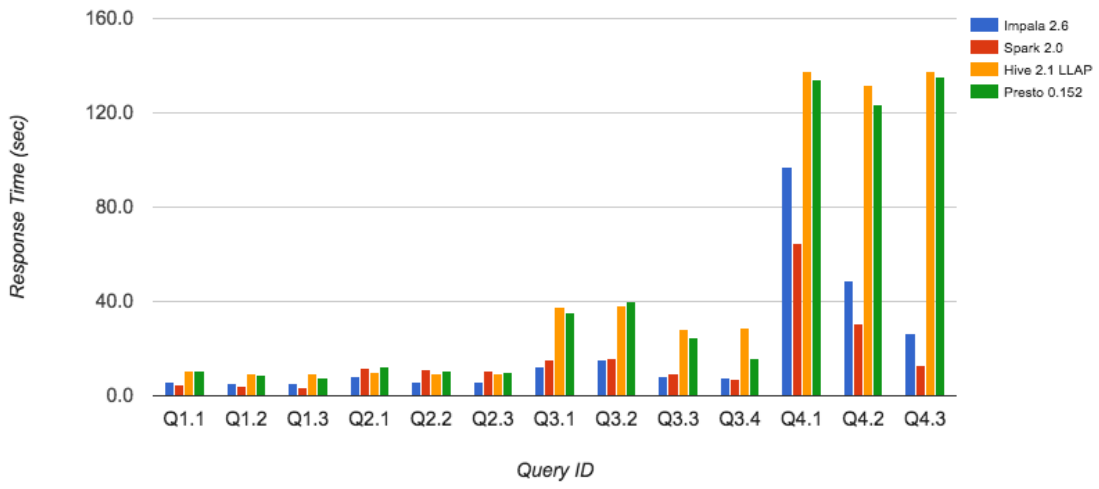
- **Increasing the number of joins generally increases query processing time.** Going from "Quick Metric" to "Product Insight" queries (where the number of joins goes from 1 to 3) had the largest query time difference on Hive and Presto. For very large joins Spark and Impala also showed increases in query response time.

- **Increased query selectivity resulted in reduced query processing time** (which may reduce the total amount of data that needs to be included in the query processing steps). Both Impala and Spark show performance improvements as query selectivity increases. The sensitivity to selectivity is consistent with earlier versions of Impala, while Spark shows improvements vs. earlier versions. Hive and Presto are much less sensitive to query selectivity.

- **JOIN operations between very large tables increased query processing time for all engines.** As expected, joining two large tables was an expensive operation for all engines. This was particularly true when involving the 1 Billion-row CUSTOMERS table.

- **As the number of joins increases, Impala and Spark SQL are more likely to perform best.** Consistent with our earlier benchmarks, Impala and Spark SQL tend to outperform Presto and Hive for queries involving larger joins.

Figure 3: Benchmark Query Results for Large Tables



Small Data Set: Characteristics & Results

Business Intelligence architectures rely on caching mechanisms in the form of in-memory stores, pre-materialized data structures, and aggregate tables. As such, a complete set of Business Intelligence benchmarks needs to evaluate the performance of the execution engines against such acceleration mechanisms.

In order to evaluate each of the engines for the benchmarks the team re-executed the Large Table Queries after the tables comprising the AtScale Adaptive Cache were generated. These tables contain aggregated measures for the attributes included in the original raw data queries, but (due to these aggregations) contain fewer rows of data. *Note - AtScale Adaptive Cache tables are stored on the Hadoop cluster in the form of Parquet or ORC tables, and can be queried by any SQL on Hadoop engine.*

AtScale Adaptive Cache™ technology provides two key functions:

- **Cache creation:** after the first queries have been executed, the AtScale engine uses heuristic functions to determine if it is optimal to store related query result sets to accelerate the processing of future queries. If so, one more aggregate tables may be created in the Adaptive Cache.

- **Cache management:** after the first caching results have been stored, the AtScale engine manages the Adaptive Cache result sets to optimize for new queries, incorporate new data entering the Hadoop cluster and to remove low-value aggregate tables.

The Table 4 shows the performance of these engines against such aggregate tables.

Small Data Set: Characteristics & Results (continued)

Table 4: Benchmark Query Results for Cache Tables

Query	Query Execution Time (in seconds)			
	Impala 2.6	Spark 2.0	Hive 2.0 LLAP	Presto 0.152
Q1.1	0.2674	0.4795	2.0752	0.3595
Q1.2	0.2983	0.4798	2.0772	0.3841
Q1.3	0.6424	0.6978	2.1079	0.4856
Q2.1	3.137	3.3371	4.3056	3.9627
Q2.2	2.2093	2.279	4.2811	3.2376
Q2.3	2.3541	2.2519	4.2756	2.7475
Q3.1	1.6381	1.1865	3.2781	2.3797
Q3.2	2.5693	1.642	8.0862	1.9072
Q3.3	2.1285	1.2819	2.2755	1.9804
Q3.4	2.4062	2.5785	5.471	3.7276
Q4.1	3.4521	5.577	6.9383	5.9396
Q4.2	4.0494	5.3763	7.6193	6.501
Q4.3	4.3873	3.582	4.5264	3.4225

**Note: fastest query time for each row is highlighted in green*

AtScale Adaptive Cache: Summary of Results

Our benchmark results indicate that both Impala, Spark SQL and Presto perform best on the AtScale Adaptive Cache tables, effectively returning query results on our 6 Billion row data set with query response times ranging from from under 300 milliseconds to several seconds.

With Hive 2.1 and the inclusion of LLAP, we also see a significant improvement in Hive query performance against the smaller tables, with an average speedup of ~2X. With this latest round of benchmark testing we feel that all four SQL-on-Hadoop engines can be legitimate options for *both* the large and small table query engines.

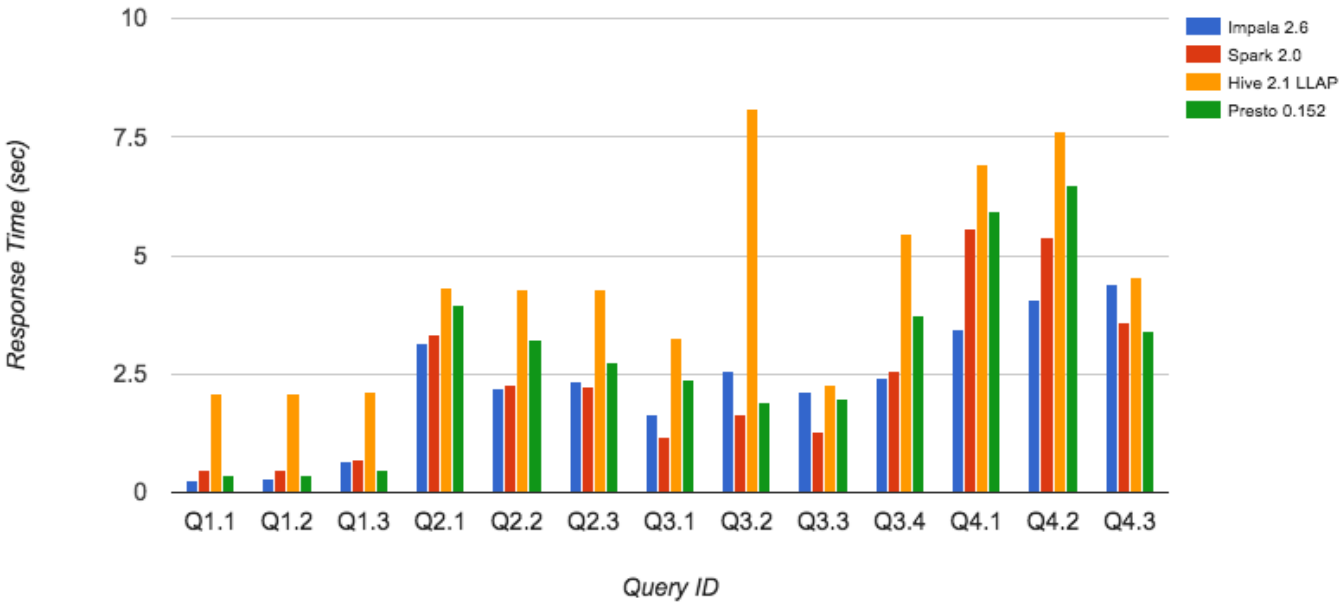
Figure 4 showcases the roughly similar query performance profiles of all 4 engines when benchmarked against the aggregate cache tables.

Small Data Set: Characteristics & Results (continued)

Figure 4 showcases the roughly similar query performance profiles of all 4 engines when benchmarked against the aggregate cache tables

.

Figure 4: Benchmark Query Results for Cache Tables



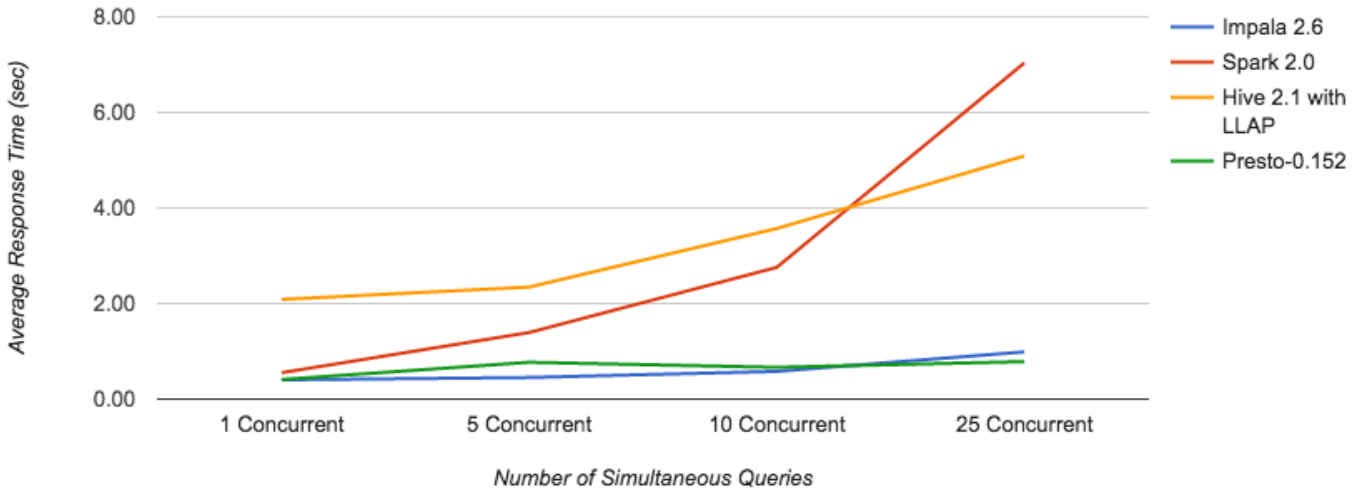
Our benchmark results also illustrate significant performance gains that can be realized from AtScale's Adaptive Cache™ technology, with query performance for the same query improving by as much as 50X. In real-life deployments on larger data sets (on the order of 100s of Billions of rows) AtScale customers have seen performance gains over 100-200X as a result of this technology.

Concurrent Query Results

Business Intelligence (BI) workloads often involve many data workers issuing queries at the same time. Concurrency is therefore a key factor when evaluating performance of a BI platform. And with an increasing number of enterprises deploying more and more production BI workloads against Hadoop platforms, there is an increasing focus on how SQL-on-Hadoop engines support concurrent query workloads.

To properly assess engine performance against concurrent queries, AtScale tested each benchmark query with a range concurrent users. A test was executed for each of the 13 queries with each test simultaneously submitting 1, 5, 10, or 25 queries simultaneously. Since production BI workloads consistently execute against aggregate tables, the results published below are for queries against the AtScale Adaptive Cache aggregate tables (the "small data set" results discussed above).

Figure 5: Concurrency Profile for Quick Metric Queries



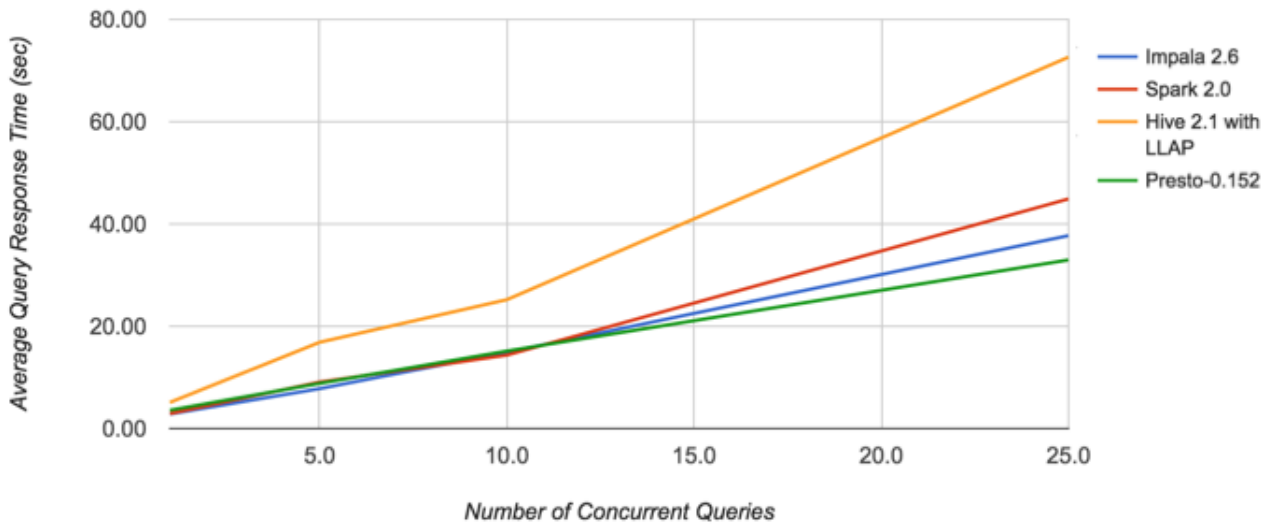
The chart above shows how the 4 tested engines perform for the "Quick Metric" queries (Q1.x quereis from Table 1) as simultaneous concurrent query volumes increase. As in the First Edition of the benchmarks,

Concurrent Query Results (continued)

While all engines were able to scale to support up to 25+ concurrent users, we did observe different behaviors from the various engines tested. Impala's performance scaled better than Spark SQL or Hive as the number of concurrent users increased these results are consistent with AtScale's experience using Impala in real-world customer deployments. Presto's MPP architecture also delivers excellent concurrent query scaling results, showing virtually no degradation with up to 25 concurrent Quick Metric queries. It is worth noting that both Presto and Impala were able to satisfy the 25 concurrent user workload and maintain an average query response time of under 1 second.

In addition to looking at dashboard style queries represented by the Quick Metric benchmarks, we also looked at the concurrency profile for more complex analytic queries (represented by queries Q2.x through Q4.x in Table 1). The results are shown in Figure 6.

Figure 6: Concurrency Profile for Quick Metric Queries



For these query patterns two characteristics are important; 1) do queries complete without failure, even under concurrent workloads; and, 2) does query performance degrade in a linear fashion.

The results are as follows:

- All engines demonstrate consistent query performance degradation under concurrent workload.
- There were no query failures for any of the engines tested, up to 25 concurrent queries.
- While all engines can effectively handle a large number of concurrent analytic queries, Spark, Impala, and Presto maintain a better average performance profile vs. Hive as concurrent query workload increases.

Cluster Environment & Engine Configurations

In order to evaluate the base performance of the SQL-on-Hadoop engines we compared the performance of Cloudera Impala, Hive-on-Tez, and Spark SQL, and Presto running against a retail data set as defined in the initial section of this document.

The test environment was an isolated 12 node cluster with 1 master node, 1 gateway node, and 10 data nodes.

Benchmark Cluster Configuration Details

RAM per node	128G
CPU specs for data (worker) nodes	32 CPU cores
Storage specs for data (worker) nodes	2x 512mb SSD

The SQL-on-Hadoop engine benchmarks were performed by running each engine individually against the same based Hadoop cluster configuration and data set. Details about engine-specific configurations and settings are below:

Impala Configuration	
Impala Version	2.6
File Format	Parquet
Workers	10
Memory per worker	110G
--num_hdfs_worker_threads	32
--max_cached_file_handles	3000

Cluster Environment & Engine Configurations (continued)

SparkSQL Configuration	
Spark Version	2.0.1-SNAPSHOT
File Format	Parquet
Workers	70
Memory per worker	16G
Cores per worker	4
Spark.kryoserializer.buffer.max	512

Hive LLAP Configuration	
Tez Version	0.8.5
Hive Version	2.1
File Format	ORC
LLAP Settings	--instances 10 --cache 50000m --xmx 55000m --size 110000m --executors 32
hive.execution.mode	llap
hive.llap.execution.mod	all
hive.llap.io.enabled	true

Presto Configuration	
Presto Version	0.152
File Format	ORC
Nodes	1 coordinator 10 worker
Memory per node	110G
sink.max-buffer-size	10GB
hive.force-local-scheduling	true
distributed-joins-enabled	false

NOTE: AtScale used default out-of-the box configuration settings and virtual machine instances to execute this benchmark. These results should be considered an assessment of relative engine performance and not a rigorous or complete performance evaluation of any single engine. While we did attempt to properly tune and configure each engine for optimal performance using publicly available best-practices and documentation, AtScale received no assistance from any other vendors.

Business Intelligence Benchmarks Summary

In this Second Edition of the Business Intelligence on Hadoop Benchmarks we discovered valuable insights and in some cases surprising results as we assessed the relative performance of several top SQL-on-Hadoop engines. As enterprises look to deliver a modern BI-on-Hadoop platform as part of their overall data modernization and consolidation initiatives, we hope that these insights help to inform critical decisions related to the design and deployment of a winning BI-on-Big-Data strategy.

At a high level, this study revealed the following about the current state of SQL-on-Hadoop engines:

- **In the SQL-on-Hadoop wars, everyone wins:** We saw significant improvements between the First and Second Editions of the benchmark, on the order of 2x to 4x, in the six months between each round of testing. This is great news for those enterprises deploying BI workloads to Hadoop. We believe that a best-of-breed strategy - best engine, best semantic BI layer, best visualization tool - will lead enterprises down the most successful path to BI-on-Hadoop success.

- **There is no single "best engine":** Presto, Hive, Impala, and Spark SQL were all able to effectively complete a range of queries on over 6 Billion rows of data. The "winning" engine for each of our benchmark queries was dependent on the query characteristics (join size, selectivity, group-bys).

- **Small vs. Big Data differences are diminishing:** With the release of Hive's LLAP technology the differences between Hive and other persistent query engines (Spark SQL, Impala, Presto) has decreased between the First and Second Edition Benchmark tests. While Impala and Spark SQL continue to shine for small data queries (queries against the AtScale Adaptive Cache), Hive now delivers suitable "small data" query response times as well. Presto also shows promise on small, interactive queries.

- **Presto and Impala scale better than Hive and Spark for concurrent dashboard queries:** Production enterprise BI user-bases may be on the order of 100s or 1,000s of users. As such, support for concurrent query workloads is critical. Our benchmarks showed that Presto and Impala performed best - that is, showed the least query degradation - as concurrent "Quick Metric" query workload increased.

- **All engines are suited to handle large user bases that generate mixed workloads:** All engines showed linear, non-breaking performance degradation for larger analytic-style query workloads. While Spark SQL, Presto, and Impala slightly outperformed Hive, all 4 engines are viable selections to handle analysis-heavy BI workloads.

- **A successful BI on Hadoop architecture will likely require more than one SQL on Hadoop engine:** Each engine has its strengths: Presto's and Impala's concurrency scaling support for quick metric queries, Spark SQL's handling of large joins, Hive's and Impala's consistency across multiple query types. Enterprises might consider leveraging different engines for different query patterns, and providing an abstraction-based query layer, such as AtScale, to automatically route queries to the best query engine for the job without additional work required or noticed on the client application side.