
An Independent Comparison of Open Source SQL-on-Hadoop

Greg Rahn | [@GregRahn](#)
#Strataconf + #HadoopWorld
17 October 2014

SELECT about FROM speaker;

- ❖ Spent the past decade as a database performance engineer
- ❖ 8 years at Oracle running competitive customer RDBMS benchmarks
- ❖ 18 months at Cloudera working on Impala performance
- ❖ I <3 SQL, SQL engines, and benchmarking

Today's Menu

- ❖ SQL + Hadoop - the past 2 years
- ❖ Project comparison
- ❖ Technical analysis of some published benchmarks
- ❖ Benchmarking thoughts

SQL ALL THE HADOOPS



It all started 2 years ago...

- ❖ October 2012: Impala (beta) announced at Strata + Hadoop World
- ❖ February 2013: Hortonworks announces “Stinger” initiative for Hive
- ❖ May 2013: Impala 1.0
- ❖ June 2013: Facebook reveals Presto at Analytics @Scale
- ❖ November 2013: Facebook open sources Presto
- ❖ April 2014: Hive “Stinger” delivered (Hive 0.11, 0.12, 0.13)
- ❖ September 2014: Hortonworks announces Hive “Stinger.next”
- ❖ October 2014: Impala 2.0

Features Comparison

Hive

- ❖ Originally developed by Facebook
- ❖ SQL to MapReduce
- ❖ Has been notoriously slow
- ❖ Hortonworks currently leading development effort (Stinger)

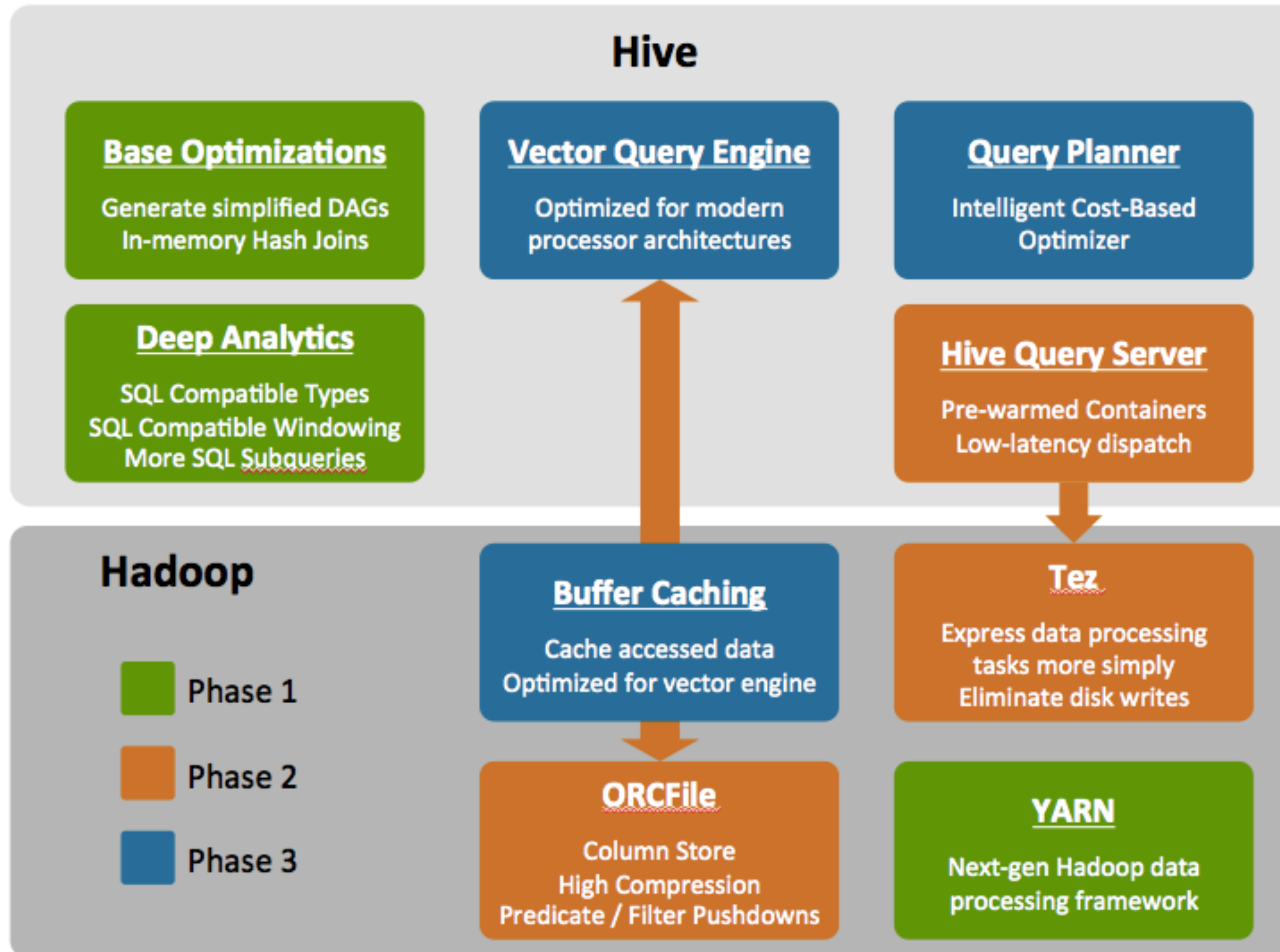


Project Stinger

- ❖ Move from MapReduce to Tez
- ❖ ORC file format & Vectorization
- ❖ In-memory hash joins (broadcast join)
- ❖ Window functions
- ❖ Decimal, Varchar, Date
- ❖ Limited subquery support
- ❖ No anti-join support



The Stinger Initiative: Making Apache Hive 100x Faster



Major Technical Advancements in Apache Hive

Yin Huai¹ Ashutosh Chauhan² Alan Gates² Gunther Hagleitner² Eric N. Hanson³
Owen O'Malley² Jitendra Pandey² Yuan Yuan¹ Rubao Lee¹ Xiaodong Zhang¹

¹The Ohio State University ²Hortonworks Inc. ³Microsoft

¹{huai, yuanyu, liru, zhang}@cse.ohio-state.edu

²{ashutosh, gates, ghagleitner, owen, jitendra}@hortonworks.com

³ehans@microsoft.com

ABSTRACT

Apache Hive is a widely used data warehouse system for Apache Hadoop, and has been adopted by many organizations for various big data analytics applications. Closely working with many users and organizations, we have identified several shortcomings of Hive in its file formats, query planning, and query execution, which are key factors determining the performance of Hive. In order to make Hive continuously satisfy the requests and requirements of processing increasingly high volumes data in a scalable and efficient way,

than 100 developers have made technical efforts to improve Hive on more than 3000 issues. With its rapid development pace, Hive has been significantly updated by new innovations and research since the original Hive paper [45] was published four years ago. We will present its major technical advancements in this paper.

Hive was originally designed as a translation layer on top of Hadoop MapReduce. It exposes its own dialect of SQL to users and translates data manipulation statements (queries) to a directed acyclic graph (DAG) of MapReduce jobs. With an SQL interface, users do not need to write tedious and sometimes difficult MapRe

Stinger.next Road Map

- ❖ ACID transactions
- ❖ Cost-based query optimization via Apache Θ ptiq Calcite
- ❖ Non-equi joins
- ❖ More subquery support
- ❖ Materialized views (DIMMQ)
- ❖ LLAP (Live Long and Process)



Presto

- ❖ Written in Java
- ❖ Demon based, not MapReduce
- ❖ Shares Hive Metastore
- ❖ Leverages bytecode compilation
- ❖ Connector based approach
 - ❖ Hive, Cassandra, Kafka, RDBMS
 - ❖ Join data across data stores

The Presto logo is displayed on a black rectangular background. It features the word "presto" in a white, lowercase, sans-serif font. To the right of the text is a cluster of approximately 20 dots in white, light blue, and dark blue, arranged in a pattern that suggests a starburst or a stylized arrow pointing to the right.

presto

Presto

- ❖ Requires explicit joins (ANSI SQL-92 syntax)
- ❖ Manual join ordering
- ❖ Non-equi joins not supported
- ❖ Large joins not a strong point
 - ❖ Distributed join (0.77 experimental)
- ❖ Numerous built-in functions
- ❖ Array / Map support

The Presto logo is displayed on a black rectangular background. It features the word "presto" in a white, lowercase, sans-serif font. To the right of the text is a cluster of approximately 20 dots in various shades of blue and white, arranged in a pattern that suggests a starburst or a network of nodes.

presto

Presto

- ❖ Approximate queries (BlinkDB)
- ❖ Distinct-limit optimization
- ❖ Window functions
- ❖ Amazon S3 support
- ❖ HyperLogLog (approx distinct)

The Presto logo is displayed on a black rectangular background. It features the word "presto" in a white, lowercase, sans-serif font. To the right of the text is a cluster of approximately 20 dots in various shades of blue and white, arranged in a pattern that suggests a starburst or a stylized arrow pointing to the right.

presto

Impala

- ❖ Open sourced by Cloudera, October 2012
- ❖ Does not build on top of MapReduce
- ❖ MPP engine for data in HDFS
- ❖ Execution engine written in C++ (LLVM)
- ❖ Leverages Parquet file format
- ❖ Currently the fastest OSS SQL engine for Hadoop



Impala 1.x Additions

- ❖ UDFs & UDAFs
- ❖ Admission Control – allows prioritization and queueing of queries
- ❖ DECIMAL data type
- ❖ Cost-based join reordering
- ❖ In-memory HDFS caching



Impala 2.0 Features

- ❖ Window functions
- ❖ Subqueries in WHERE clause, but not in the HAVING clause
- ❖ Disk-based joins
- ❖ CHAR & VARCHAR data types



Impala 2.1+ Road Map

- ❖ Nested data
- ❖ MERGE
- ❖ ROLLUP, CUBE, GROUPING SET
- ❖ Set operators - MINUS, INTERSECT
- ❖ Apache HBase CRUD
- ❖ UDTFs
- ❖ Intra-node parallelism for aggregations and joins
- ❖ Parquet enhancements including index pages
- ❖ Amazon S3 integration

Analyzing Benchmark Reports

MATURE 17+

TM

®

M

Technical themes
Revealing facts
Intense analysis
Critical thinking

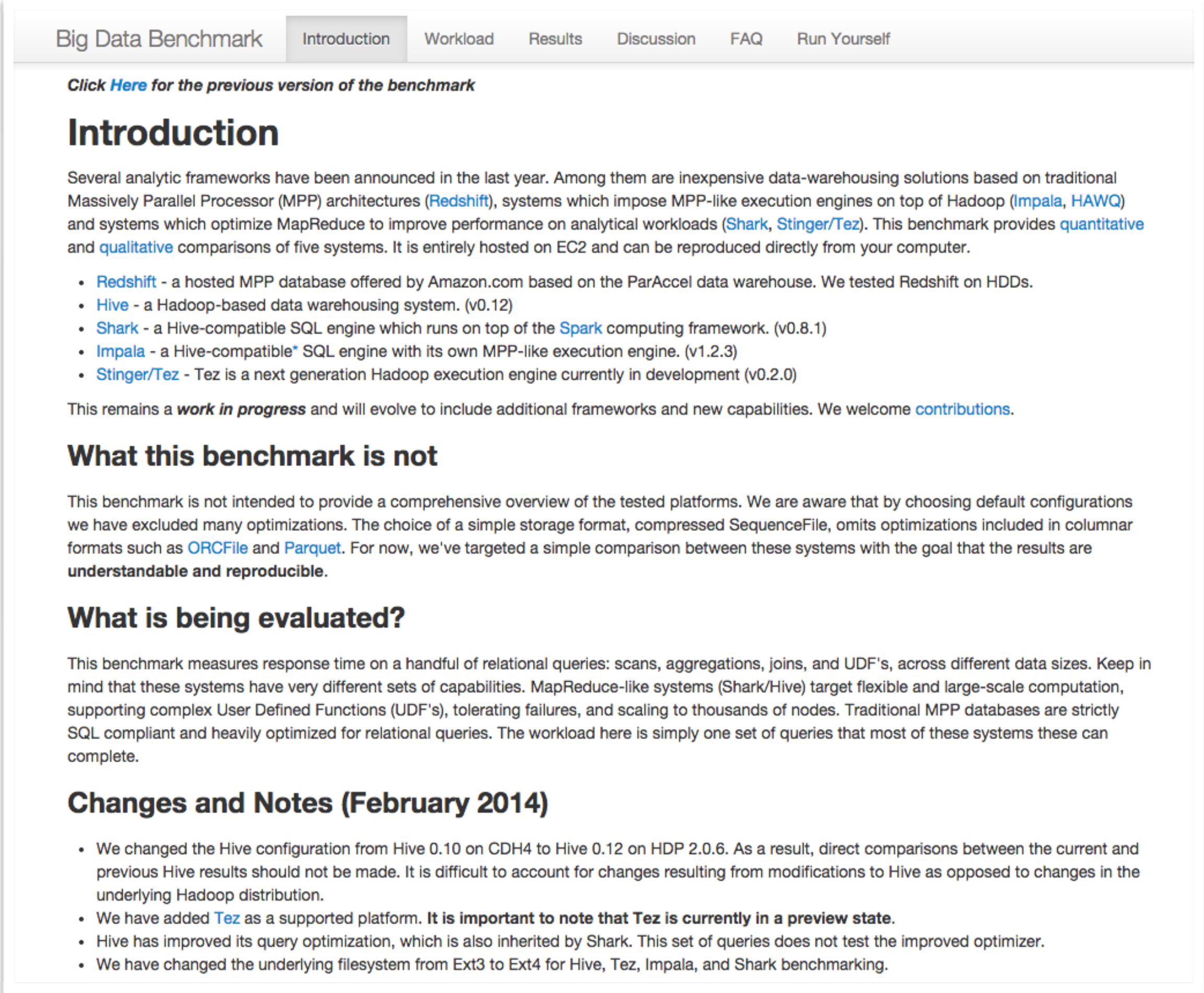
ESRB CONTENT RATING

www.esrb.org

AMPLab Big Data Benchmark

AMPLab Big Data Benchmark

- ❖ Multiple systems (Impala, Hive, Shark)
- ❖ Runs in AWS with GitHub repo
- ❖ Based on “*A Comparison of Approaches to Large-Scale Data Analysis*” by Pavlo et al.
- ❖ Very simple queries, some with very large results
- ❖ Uses common, not optimal, file format



The screenshot shows the 'Introduction' page of the 'Big Data Benchmark' website. The navigation bar includes 'Big Data Benchmark', 'Introduction', 'Workload', 'Results', 'Discussion', 'FAQ', and 'Run Yourself'. A link for the previous version is provided. The main content includes an introduction paragraph, a bulleted list of systems tested (Redshift, Hive, Shark, Impala, Stinger/Tez), a note about the benchmark's progress, a section on what the benchmark is not, a section on what is being evaluated, and a section on changes and notes from February 2014.

Big Data Benchmark Introduction Workload Results Discussion FAQ Run Yourself

[Click Here](#) for the previous version of the benchmark

Introduction

Several analytic frameworks have been announced in the last year. Among them are inexpensive data-warehousing solutions based on traditional Massively Parallel Processor (MPP) architectures ([Redshift](#)), systems which impose MPP-like execution engines on top of Hadoop ([Impala](#), [HAWQ](#)) and systems which optimize MapReduce to improve performance on analytical workloads ([Shark](#), [Stinger/Tez](#)). This benchmark provides [quantitative](#) and [qualitative](#) comparisons of five systems. It is entirely hosted on EC2 and can be reproduced directly from your computer.

- [Redshift](#) - a hosted MPP database offered by Amazon.com based on the ParAccel data warehouse. We tested Redshift on HDDs.
- [Hive](#) - a Hadoop-based data warehousing system. (v0.12)
- [Shark](#) - a Hive-compatible SQL engine which runs on top of the [Spark](#) computing framework. (v0.8.1)
- [Impala](#) - a Hive-compatible* SQL engine with its own MPP-like execution engine. (v1.2.3)
- [Stinger/Tez](#) - Tez is a next generation Hadoop execution engine currently in development (v0.2.0)

This remains a **work in progress** and will evolve to include additional frameworks and new capabilities. We welcome [contributions](#).

What this benchmark is not

This benchmark is not intended to provide a comprehensive overview of the tested platforms. We are aware that by choosing default configurations we have excluded many optimizations. The choice of a simple storage format, compressed SequenceFile, omits optimizations included in columnar formats such as [ORCFile](#) and [Parquet](#). For now, we've targeted a simple comparison between these systems with the goal that the results are **understandable and reproducible**.

What is being evaluated?

This benchmark measures response time on a handful of relational queries: scans, aggregations, joins, and UDF's, across different data sizes. Keep in mind that these systems have very different sets of capabilities. MapReduce-like systems (Shark/Hive) target flexible and large-scale computation, supporting complex User Defined Functions (UDF's), tolerating failures, and scaling to thousands of nodes. Traditional MPP databases are strictly SQL compliant and heavily optimized for relational queries. The workload here is simply one set of queries that most of these systems these can complete.

Changes and Notes (February 2014)

- We changed the Hive configuration from Hive 0.10 on CDH4 to Hive 0.12 on HDP 2.0.6. As a result, direct comparisons between the current and previous Hive results should not be made. It is difficult to account for changes resulting from modifications to Hive as opposed to changes in the underlying Hadoop distribution.
- We have added [Tez](#) as a supported platform. **It is important to note that Tez is currently in a preview state.**
- Hive has improved its query optimization, which is also inherited by Shark. This set of queries does not test the improved optimizer.
- We have changed the underlying filesystem from Ext3 to Ext4 for Hive, Tez, Impala, and Shark benchmarking.

Orca SIGMOD '14 Paper

Orca SIGMOD '14 Paper

- ❖ SIGMOD'14: June 22–27, 2014
- ❖ December 13, 2013: 2nd Paper submission
- ❖ September 16, 2013: 1st Paper submission
- ❖ Paper users Impala 1.1.1 (July 2013)
- ❖ December 2013: Impala 1.2.2 contained join order optimization

Orca: A Modular Query Optimizer Architecture for Big Data

Mohamed A. Soliman*, Lyublena Antova*, Venkatesh Raghavan*, Amr El-Helw*, Zhongxian Gu*, Entong Shen*, George C. Caragea*, Carlos Garcia-Alvarado*, Foyzur Rahman*, Michalis Petropoulos*, Florian Waas†, Sivaramakrishnan Narayanan‡, Konstantinos Krikellas†, Rhonda Baldwin*

* Pivotal Inc.
Palo Alto, USA

‡ Datometry Inc.
San Francisco, USA

† Google Inc.
Mountain View, USA

§ Qubole Inc.
Mountain View, USA

ABSTRACT

The performance of analytical query processing in data management systems depends primarily on the capabilities of the system's query optimizer. Increased data volumes and heightened interest in processing complex analytical queries have prompted Pivotal to build a new query optimizer.

In this paper we present the architecture of Orca, the new query optimizer for all Pivotal data management products, including Pivotal Greenplum Database and Pivotal HAWQ. Orca is a comprehensive development uniting state-of-the-art query optimization technology with own original research resulting in a modular and portable optimizer architecture.

In addition to describing the overall architecture, we highlight several unique features and present performance comparisons against other systems.

Categories and Subject Descriptors

H.2.4 [Database Management]: Systems—*Query processing; Distributed databases*

Keywords

Query Optimization, Cost Model, MPP, Parallel Processing

1. INTRODUCTION

Big Data has brought about a renewed interest in query optimization as a new breed of data management systems has pushed the envelope in terms of unprecedented scalability, availability, and processing capabilities (cf. e.g., [9, 18, 20, 21]), which makes large datasets of hundreds of terabytes or even petabytes readily accessible for analysis

Despite a plethora of research in this area, most existing query optimizers in both commercial and open source projects are still primarily based on technology dating back to the early days of commercial database development [22], and are frequently prone to produce suboptimal results.

Realizing this significant gap between research and practical implementations, we have set out to devise an architecture that meets current requirements, yet promises enough headroom for future developments.

In this paper, we describe *Orca*, the result of our recent research and development efforts at Greenplum/Pivotal. Orca is a state-of-the-art query optimizer specifically designed for demanding analytics workloads. It is distinguished from other optimizers in several important ways:

Modularity. Using a highly extensible abstraction of metadata and system description, Orca is no longer confined to a specific host system like traditional optimizers. Instead it can be ported to other data management systems quickly through plug-ins supported by its Metadata Provider SDK.

Extensibility. By representing all elements of a query and its optimization as first-class citizens of equal footing, Orca avoids the trap of multi-phase optimization where certain optimizations are dealt with as an afterthought. Multi-phase optimizers are notoriously difficult to extend as new optimizations or query constructs often do not match the previously set phase boundaries.

Multi-core ready. Orca deploys a highly efficient multi-core aware scheduler that distributes individual fine-

Orca SIGMOD '14 Paper

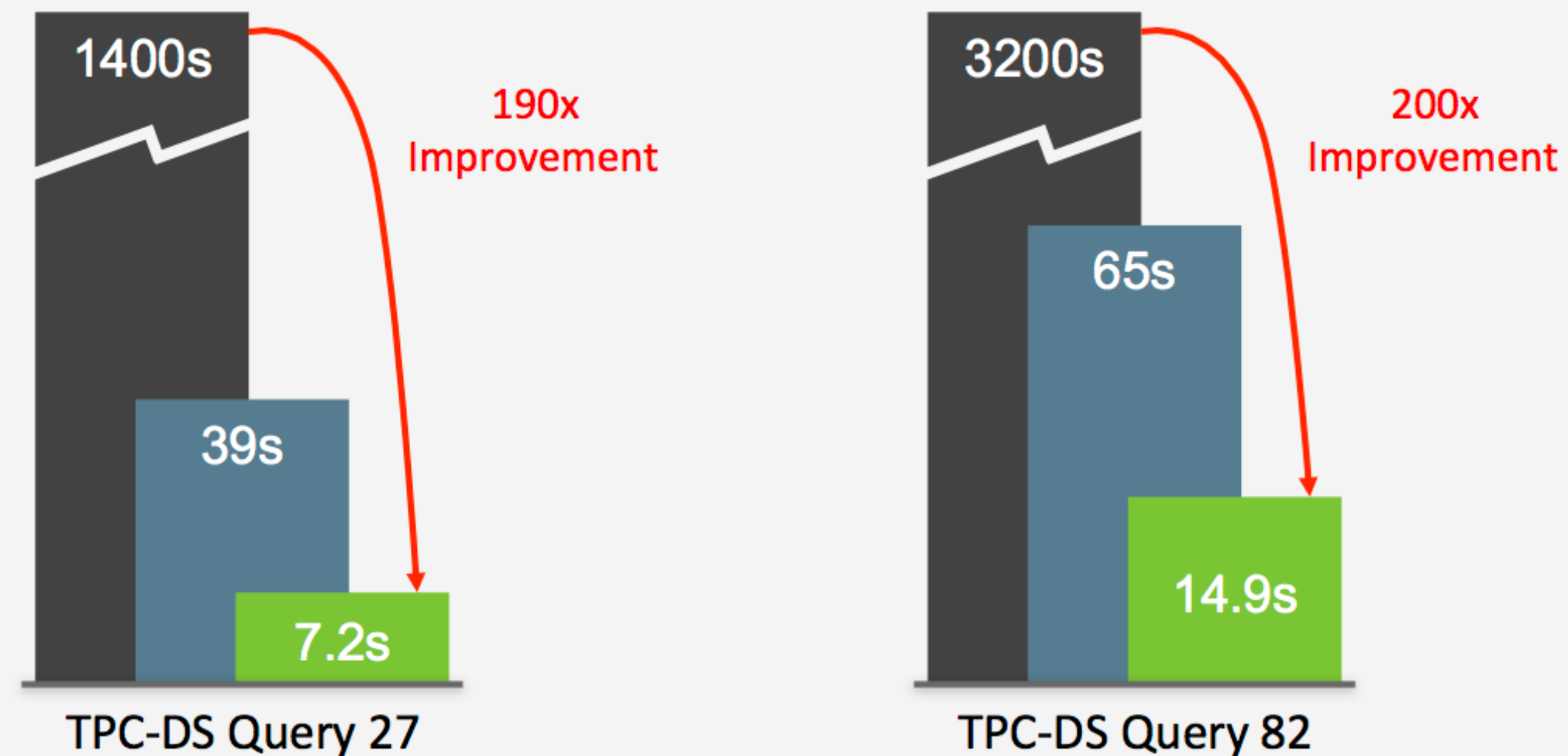
- ❖ Did the comparisons use the same: partitioning strategy? file format?
- ❖ “For [TPC-DS] query 46, 59 and 68, Impala and HAWQ have similar performance.”
- ❖ “For queries where HAWQ has the most speedups, we find that Impala and Stinger handle join orders as literally specified in the query...”
- ❖ “...for 14 queries Orca achieves a speed-up ratio of at least 1000x...”

Hortonworks Benchmarks

Stinger Phase 3: Interactive Query In Hadoop

This slide fails to call out very important details in this comparison. Hive 0.10 does not use partitioning on the fact table (which eliminates 80% of the data for query 27) and it uses text files for storage (not RCFile) resulting in the absolute worst case performance and thus provides inflated “times faster” number.

Query 27: Pricing Analytics using Star Schema Join Query 82: Inventory Analytics Joining 2 Large Fact Tables



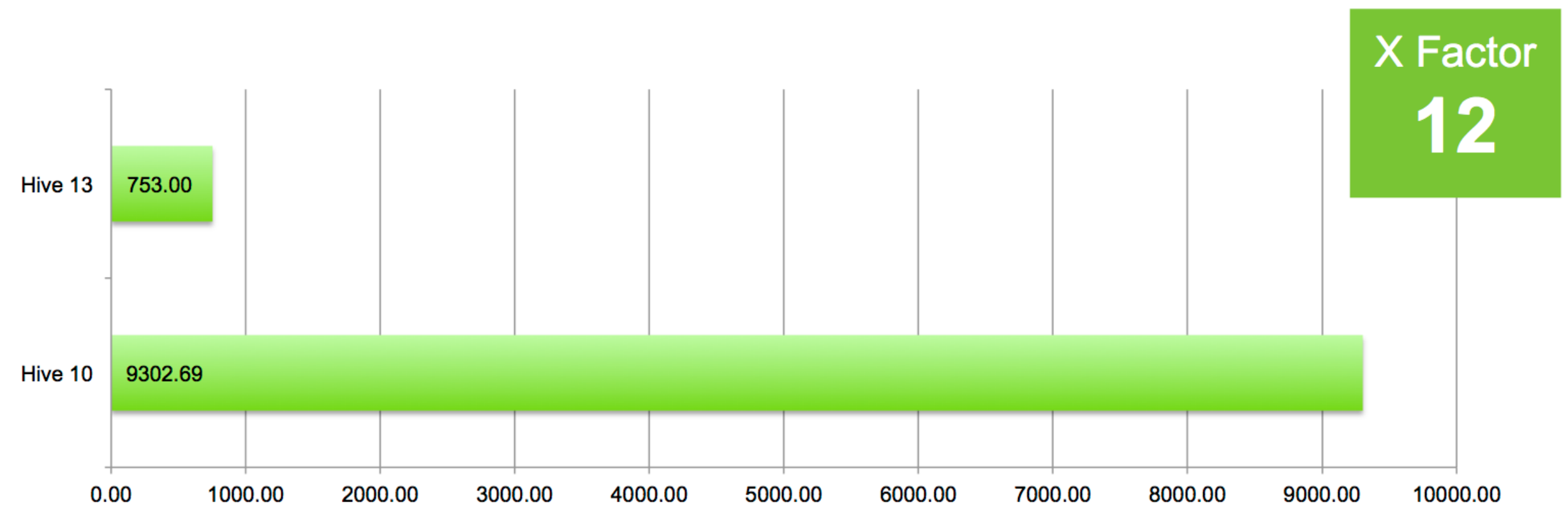
■ Hive 10 ■ Hive 0.11 (Phase 1) ■ Trunk (Phase 3)

All Results at Scale Factor 200 (Approximately 200GB Data)

This comparison uses the same partitioning for both versions but uses RCFile for Hive 0.10 and ORCFile for Hive 0.13 (the “best of” for the given version). Although these results are on a 30TB, not 200GB data set, the “times factor” drops from 200x to 12x for query 82, and 190x to 101x for query 27 compared to the previously slide. These represent a more reasonable comparison between the two versions.

Query 82

Find customers who tend to spend more money (net-paid) on-line than in stores.

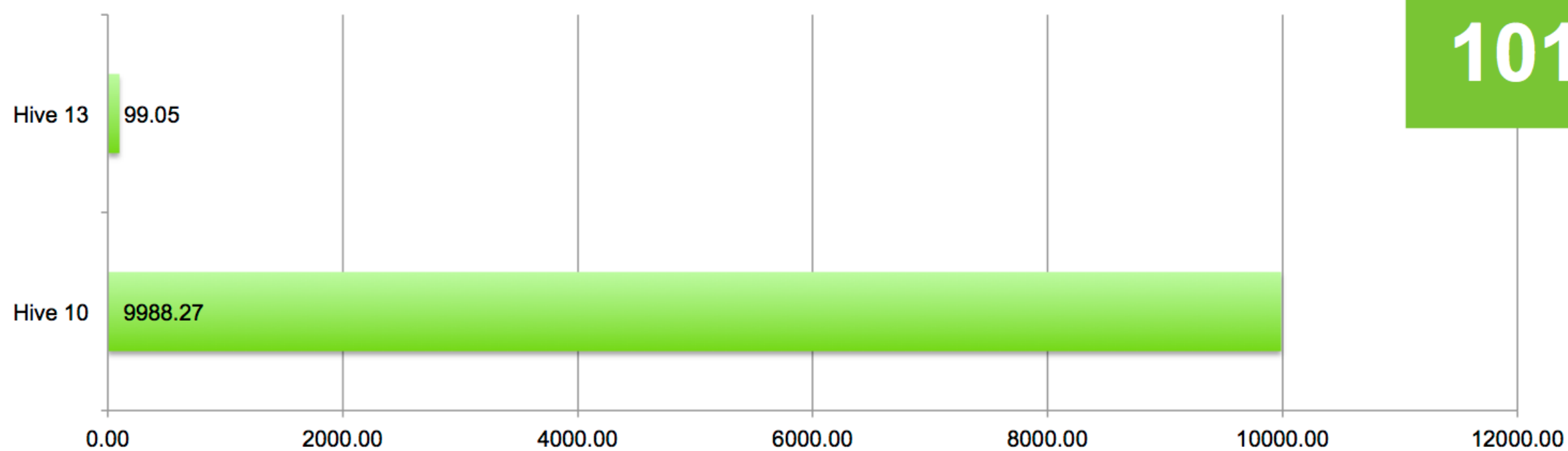


All Values in Seconds



Query 27

For all items sold in stores located in six states during average quantity, average list price, average list sales amount for a given gender, marital status, education a demographic.



All Values in Seconds



SQL-on-Hadoop:

Full Circle Back to Shared-Nothing Database Architectures

SQL-on-Hadoop: Full Circle Back

- ❖ VLDB 2014
- ❖ Researchers from IBM Almaden
- ❖ No IBM product involved
- ❖ Impala 1.2.2
- ❖ Hive 0.13 + Tez 0.3.0
- ❖ TPC-H/TPC-DS inspired workloads

SQL-on-Hadoop: Full Circle Back to Shared-Nothing Database Architectures

Avrilia Floratou
IBM Almaden
Research Center
aflorat@us.ibm.com

Umar Farooq Minhas
IBM Almaden
Research Center
ufminhas@us.ibm.com

Fatma Özcan
IBM Almaden
Research Center
fozcan@us.ibm.com

ABSTRACT

SQL query processing for analytics over Hadoop data has recently gained significant traction. Among many systems providing some SQL support over Hadoop, Hive is the first native Hadoop system that uses an underlying framework such as MapReduce or Tez to process SQL-like statements. Impala, on the other hand, represents the new emerging class of SQL-on-Hadoop systems that exploit a shared-nothing parallel database architecture over Hadoop. Both systems optimize their data ingestion via columnar storage, and promote different file formats: ORC and Parquet. In this paper, we compare the performance of these two systems by conducting a set of cluster experiments using a TPC-H like benchmark and two TPC-DS inspired workloads. We also closely study the I/O efficiency of their columnar formats using a set of micro-benchmarks. Our results show that Impala is 3.3X to 4.4X faster than Hive on MapReduce and 2.1X to 2.8X than Hive on Tez for the overall TPC-H experiments. Impala is also 8.2X to 10X faster than Hive on MapReduce and about 4.3X faster than Hive on Tez for the TPC-DS inspired experiments. Through detailed analysis of experimental results, we identify the reasons for this performance gap and examine the strengths and limitations of each system.

1. INTRODUCTION

Enterprises are using Hadoop as a central data repository for all their data coming from various sources, including operational systems, social media and the web, sensors and smart devices, as well as their applications. Various Hadoop frameworks are used to manage and run deep analytics in order to gain actionable insights from the data, including text analytics on unstructured text, log analysis over semi-structured data, as well as relational-like SQL processing over semi-structured and structured data.

Hadoop, which uses another framework such as MapReduce or Tez to process SQL-like queries, leveraging its task scheduling and load balancing features. Shark [7, 24] is somewhat similar to Hive in that it uses another framework, Spark [8] as its runtime. In this category, Impala [10] moved away from MapReduce to a shared-nothing parallel database architecture. Impala runs queries using its own long-running daemons running on every HDFS DataNode, and instead of materializing intermediate results, pipelines them between computation stages. Similar to Impala, LinkedIn Tajo [20], Facebook Presto [17], and MapR Drill [4], also resemble parallel databases and use long-running custom-built processes to execute SQL queries in a distributed fashion.

In the second category, Hadapt [2] also exploits Hadoop scheduling and fault-tolerance, but uses a relational database (PostgreSQL) to execute query fragments. Microsoft PolyBase [11] and Pivotal HAWQ [9], on the other hand, use database query optimization and planning to schedule query fragments, and directly read HDFS data into database workers for processing. Overall, we observe a big convergence to shared-nothing database architectures among the SQL-on-Hadoop systems.

In this paper, we focus on the first category of native SQL-on-Hadoop systems, and investigate the performance of Hive and Impala, highlighting their different design trade-offs through detailed experiments and analysis. We picked these two systems due to their popularity as well as their architectural differences. Impala and Hive are the SQL offerings from two major Hadoop distribution vendors, Cloudera and Hortonworks. As a result, they are widely used in the enterprise. There are other SQL-on-Hadoop systems, such as Presto and Tajo, but these systems are mainly used by companies that created them, and are not as widely used. Impala is a good representative of emerging SQL-on-Hadoop systems, such as Presto, and Tajo, which follow a shared-nothing database like ar-

Figure 7 removes explicit partition key filters resulting in more data to scan & join. Due to Impala being more efficient, the Impala “times faster than Hive” number actually increases in Fig. 7 compared to Fig. 6.

More details in section 3.7 in the paper.

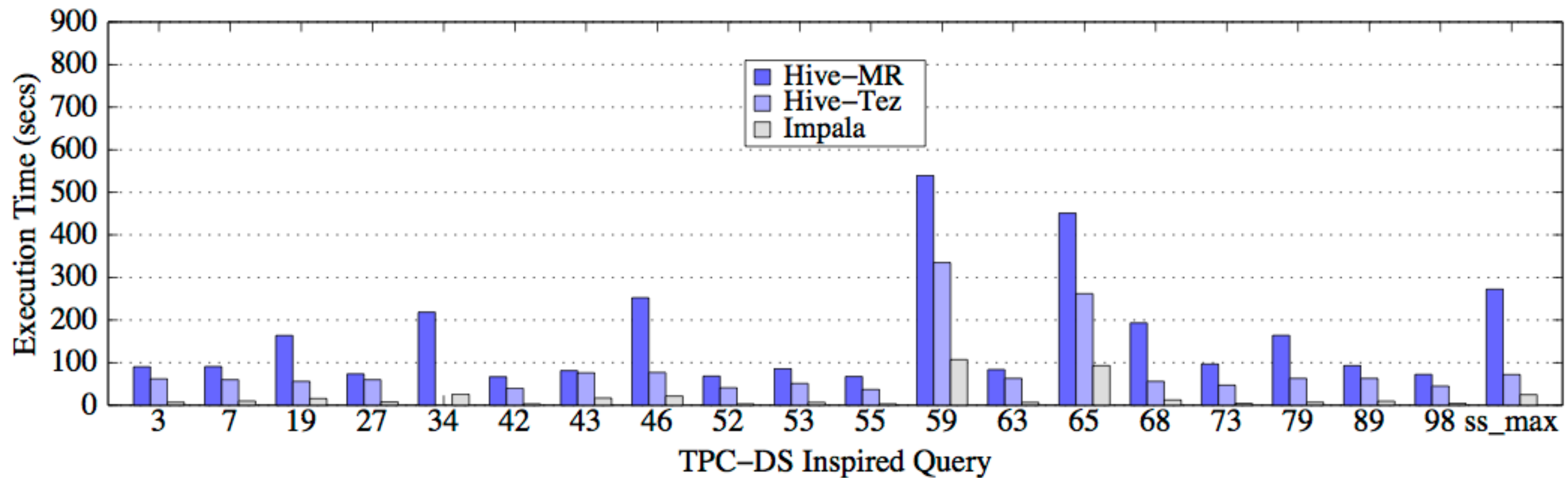


Figure 6: TPC-DS inspired Workload 1

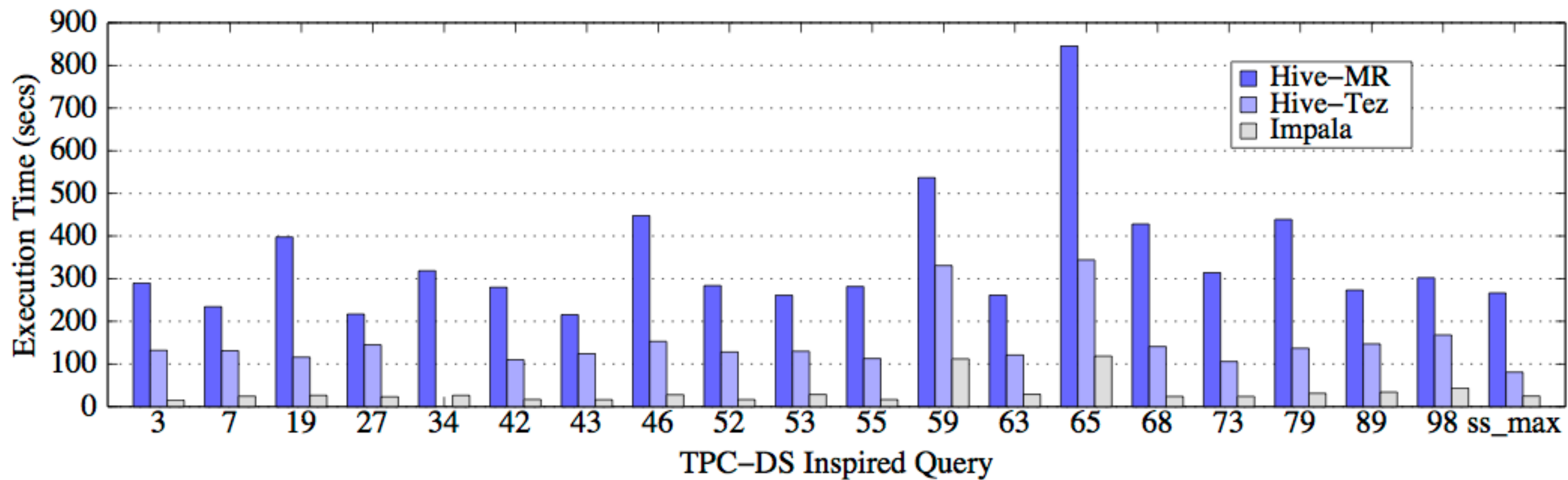
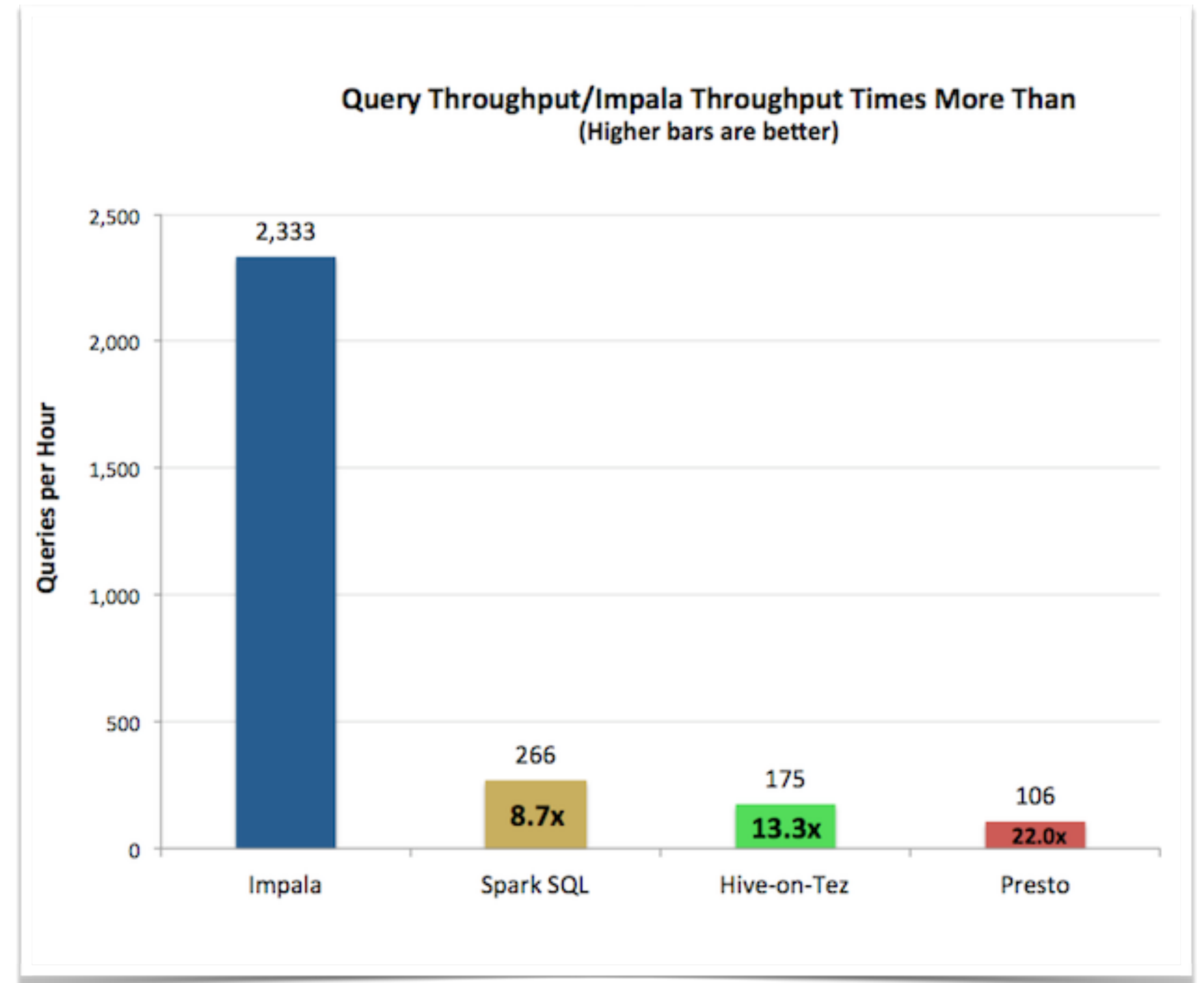


Figure 7: TPC-DS inspired Workload 2

Cloudera Benchmarks

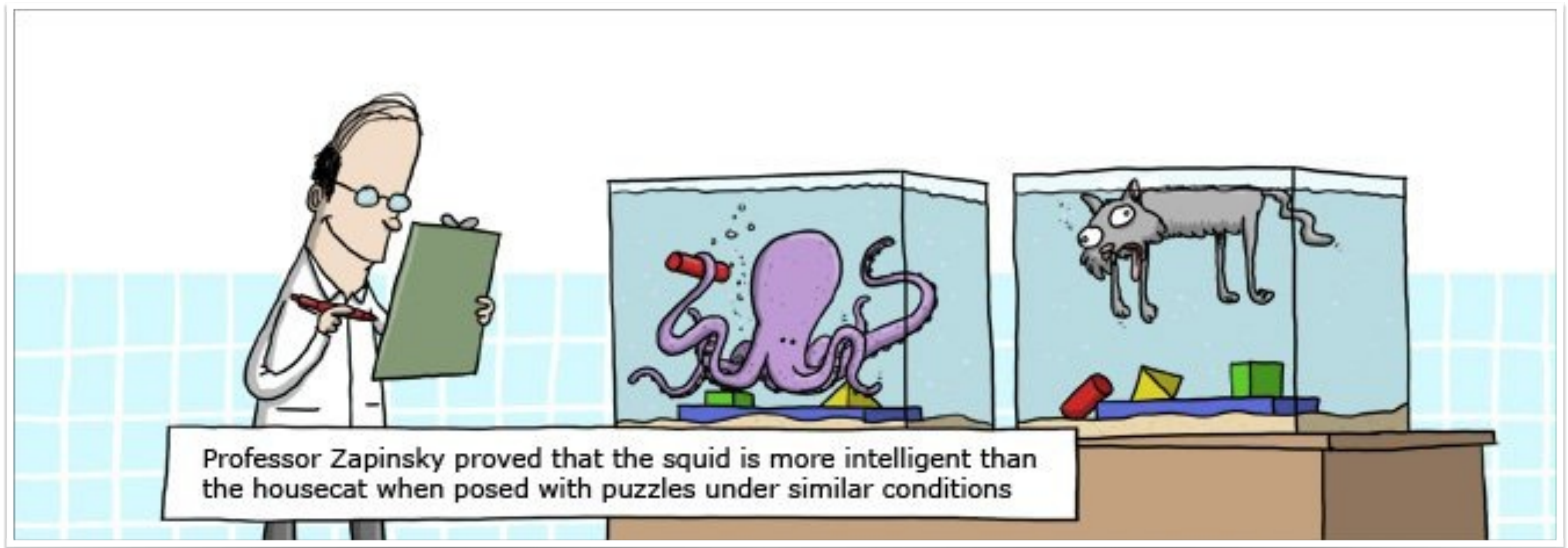
Cloudera Benchmarks

- ❖ Based on TPC-DS ([GitHub repo](#))
 - ❖ Single fact table
 - ❖ Queries add partition key pruning predicates
- ❖ Three perf blog posts this year [[1](#), [2](#), [3](#)]
- ❖ Multi-user workloads use the “interactive” group queries
- ❖ Will be interesting to see Impala 2.0 benchmarks



Thoughts on Benchmarks

What are we actually comparing?



Read the Fine Print

- ❖ Software versions
- ❖ Hardware configuration
 - ❖ # of nodes
 - ❖ RAM
 - ❖ Storage
 - ❖ Networking
- ❖ File format
- ❖ Partitioning



Benchmarking vs. Benchmarketing

Gregorio's Benchmarking Theorem

Given any benchmarking claim c , there exists at least one workload w or at least one query q that will prove claim c correct.

Closing Thoughts On Benchmarks

- ❖ Most benchmark[eting] reports are lossy
- ❖ No benchmark is perfect, so get over it
- ❖ Take what is given to you and learn from it, if possible
- ❖ Use simple experiments to prove simple things
- ❖ Be aware of unknown unknowns
- ❖ If you modify standard benchmarks (TPC-H / TPC-DS)
 - ❖ Very, very, very, clearly state so
 - ❖ Share your modifications on GitHub
- ❖ Nothing bests your queries on your data



/ Thank You */*

SELECT question

FROM audience

WHERE isAwesome(question);

 [@GregRahn](https://twitter.com/GregRahn)

References

- ❖ <http://hortonworks.com/blog/100x-faster-hive/>
- ❖ <http://web.cse.ohio-state.edu/hpcs/WWW/HTML/publications/papers/TR-14-2.pdf>
- ❖ <http://hortonworks.com/blog/stinger-next-enterprise-sql-hadoop-scale-apache-hive/>
- ❖ <https://code.facebook.com/videos/1418527681712988/introducing-presto-analytics-scale-2013/>
- ❖ <https://www.facebook.com/notes/facebook-engineering/presto-interacting-with-petabytes-of-data-at-facebook/10151786197628920>
- ❖ <http://www.slideshare.net/dain1/presto-meetup-20140514-34731104>
- ❖ <http://blog.cloudera.com/blog/2014/08/whats-next-for-impala-focus-on-advanced-sql-functionality/>
- ❖ <http://blog.cloudera.com/blog/2014/10/new-in-cdh-5-2-more-sql-functionality-and-compatibility-for-impala-2-0/>
- ❖ <https://amplab.cs.berkeley.edu/benchmark/>
- ❖ <http://www.pivotal.io/sites/default/files/SIGMODMay2014HAWQAdvantages.pdf>
- ❖ <http://www.slideshare.net/alanfgates/strata-stingertalk-oct2013>
- ❖ <http://www.slideshare.net/hortonworks/apache-hive-013-performance-benchmarks>
- ❖ <https://github.com/cartershanklin/hive-testbench/>
- ❖ <http://www.vldb.org/pvldb/vol7/p1295-floratou.pdf>
- ❖ <http://blog.cloudera.com/blog/2014/01/impala-performance-dbms-class-speed/>
- ❖ <http://blog.cloudera.com/blog/2014/05/new-sql-choices-in-the-apache-hadoop-ecosystem-why-impala-continues-to-lead/>
- ❖ <http://blog.cloudera.com/blog/2014/09/new-benchmarks-for-sql-on-hadoop-impala-1-4-widens-the-performance-gap/>
- ❖ <https://github.com/cloudera/impala-tpcds-kit>