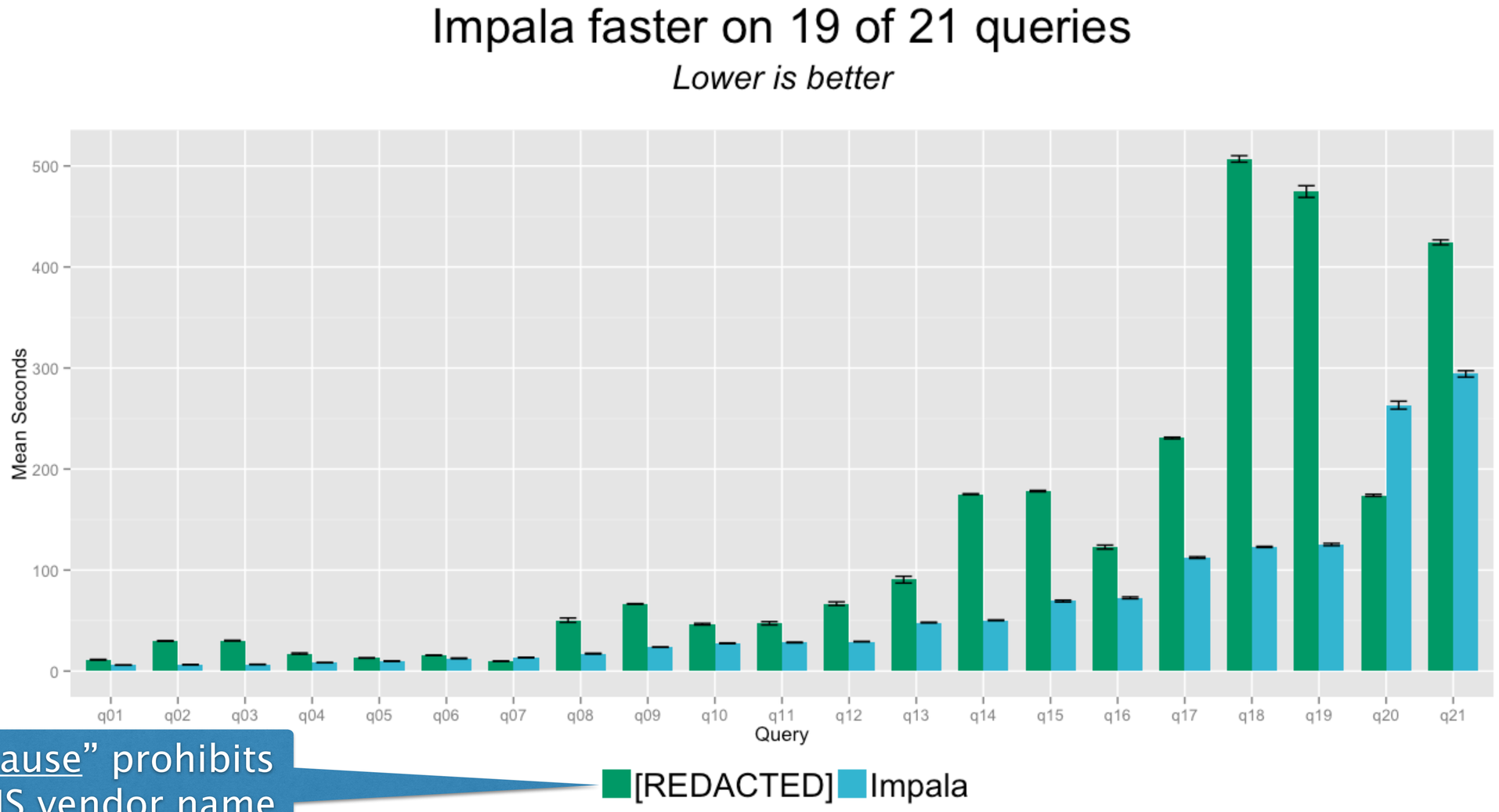
cloudera
Ask Bigger Questions

Hadoop Beyond Batch: Real-time Workloads, SQL-on-Hadoop, and theVirtual EDW

Marcel Kornacker | marcel@cloudera.com

February 2014

Analytic Workloads on Hadoop: Where Do We Stand?



"DeWitt Clause" prohibits using DBMS vendor name

cloudera
Ask Bigger Questions

Hadoop for Analytic Workloads

- Hadoop has traditional been utilized for offline batch processing: ETL and ELT
- Next step: Hadoop for traditional business intelligence (BI)/data warehouse (EDW) workloads:
  - interactive
  - concurrent users
- Topic of this talk: a Hadoop-based open-source stack for EDW workloads:
  - HDFS: a high-performance storage system
  - Parquet: a state-of-the-art columnar storage format
  - Impala: a modern, open-source SQL engine for Hadoop

Hadoop for Analytic Workloads

- Thesis of this talk:
  - techniques and functionality of established commercial solutions are either already available or are rapidly being implemented in Hadoop stack
  - Hadoop stack is effective solution for certain EDW workloads
  - Hadoop-based EDW solution maintains Hadoop's strengths: flexibility, ease of scaling, cost effectiveness

HDFS: A Storage System for Analytic Workloads

- Available in Hdfs today:
  - high-efficiency data scans at or near hardware speed, both from disk and memory
- On the immediate roadmap:
  - co-partitioned tables for even faster distributed joins
  - temp-FS: write temp table data straight to memory, bypassing disk

HDFS: The Details

- High efficiency data transfers
  - short-circuit reads: bypass DataNode protocol when reading from local disk
    -> read at 100+MB/s per disk
  - HDFS caching: access explicitly cached data w/o copy or checksumming
    -> access memory-resident data at memory bus speed

HDFS: The Details

- Coming attractions:
  - affinity groups: collocate blocks from different files
    -> create co-partitioned tables for improved join performance
  - temp-fs: write temp table data straight to memory, bypassing disk
    -> ideal for iterative interactive data analysis

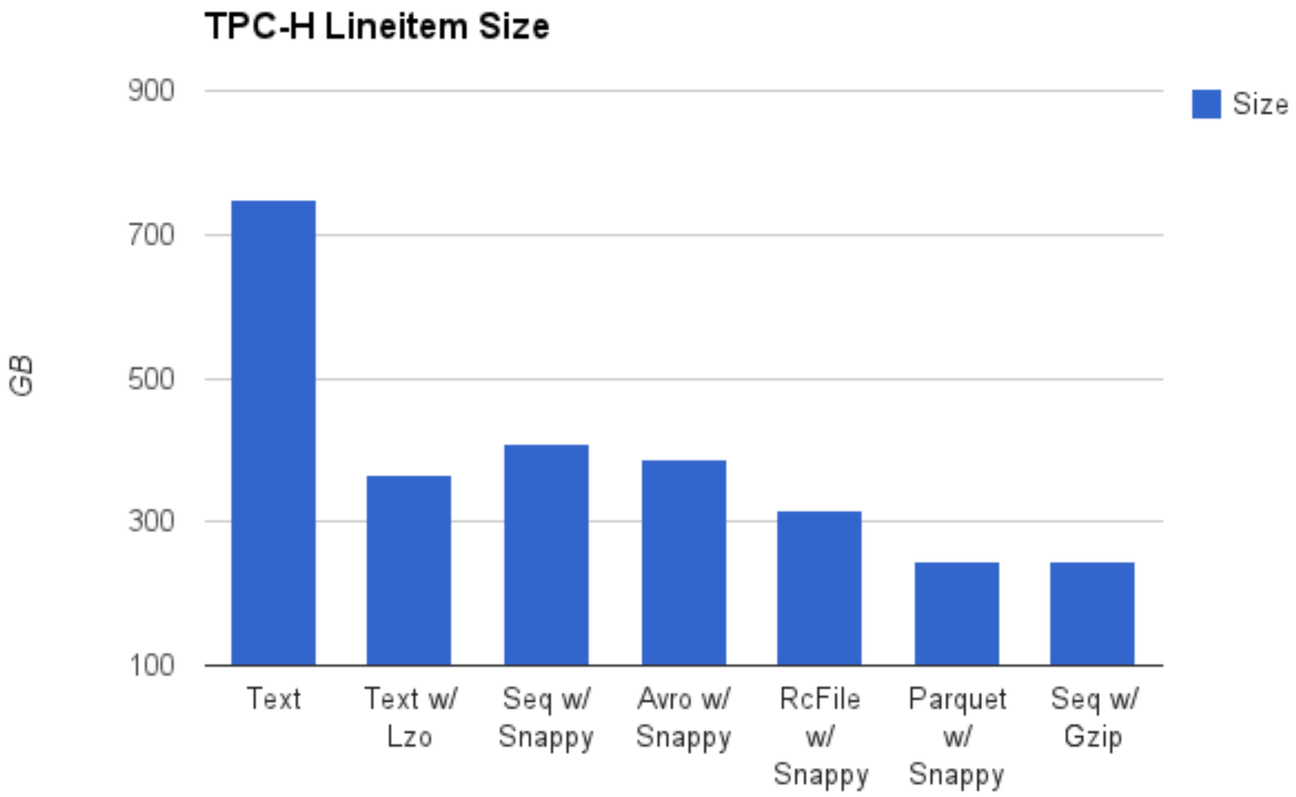
Parquet: Columnar Storage for Hadoop

- What it is:
  - state-of-the-art, open-source columnar file format that's available for (most) Hadoop processing frameworks: Impala, Hive, Pig, MapReduce, Cascading, …
  - offers both high compression and high scan efficiency
  - co-developed by Twitter and Cloudera; hosted on github and soon to be an Apache incubator project
  - with contributors from Criteo, Stripe, Berkeley AMPlab, LinkedIn
  - used in production at Twitter and Criteo

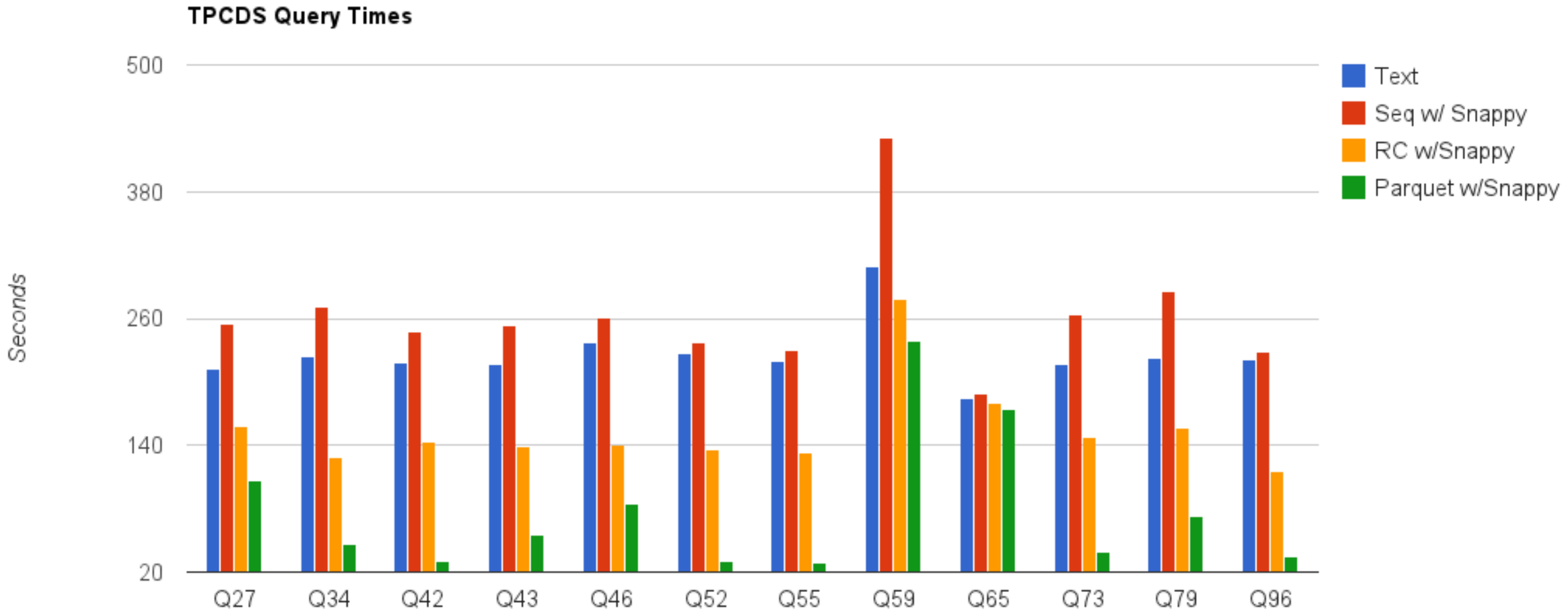
Parquet: The Details

- columnar storage: column-major instead of the traditional row-major layout; used by all high-end analytic DBMSs
- optimized storage of nested data structures: patterned after Dremel's ColumnIO format
- extensible set of column encodings:
  - run-length and dictionary encodings in current version (1.2)
  - delta and optimized string encodings in 2.0
- embedded statistics: version 2.0 stores inlined column statistics for further optimization of scan efficiency

Parquet: Storage Efficiency



Parquet: Scan Efficiency



Impala: A Modern, Open-Source SQL Engine

- implementation of an MPP SQL query engine for the Hadoop environment
- highest-performance SQL engine for the Hadoop ecosystem; already outperforms some of its commercial competitors
- effective for EDW-style workloads
- maintains Hadoop flexibility by utilizing standard Hadoop components (HDFS, Hbase, Metastore, Yarn)
- plays well with traditional BI tools: exposes/interacts with industry-standard interfaces (odbc/jdbc, Kerberos and LDAP, ANSI SQL)

Impala: A Modern, Open-Source SQL Engine

- history:
  - developed by Cloudera and fully open-source; hosted on github
  - released as beta in 10/2012
  - 1.0 version available in 05/2013
  - current version is 1.2.3, available for CDH4 and CDH5 beta

Impala from The User's Perspective

- create tables as virtual views over data stored in HDFS or Hbase;
  schema metadata is stored in Metastore (shared with Hive, Pig, etc.; basis of HCatalog)
- connect via odbc/jdbc; authenticate via Kerberos or LDAP
- run standard SQL:
  - current version: ANSI SQL-92 (limited to SELECT and bulk insert) minus correlated subqueries, has UDFs and UDAs

Impala from The User's Perspective

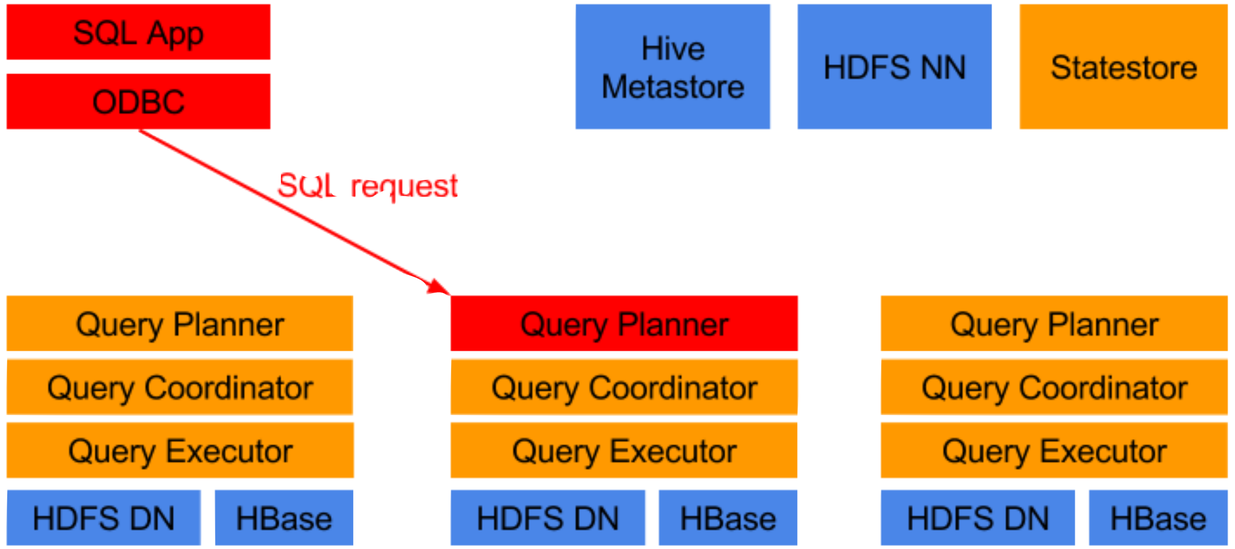
- 2014 SQL roadmap:
  - 1.3: analytic window functions, Order By without Limit, Decimal(<precision>, <scale>)
  - 2.0: support for nested types (structs, arrays, maps), UDTFs
  - >2.0: disk-based joins and aggregation, subqueries and Exists, set operators (Intersect, Minus)

Impala Architecture

- distributed service:
  - daemon process (impalad) runs on every node with data
  - easily deployed with Cloudera Manager
  - each node can handle user requests; load balancer configuration for multi-user environments recommended
- query execution phases:
  - client request arrives via odbc/jdbc
  - planner turns request into collection of plan fragments
  - coordinator initiates execution on remote impala's

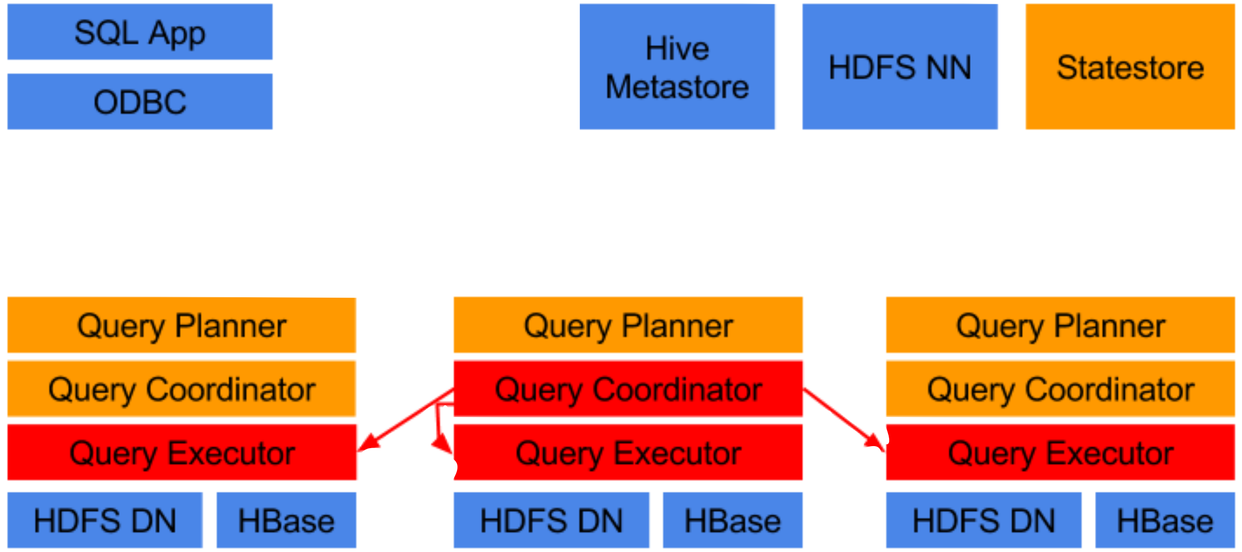
Impala Query Execution

- Request arrives via odbc/jdbc



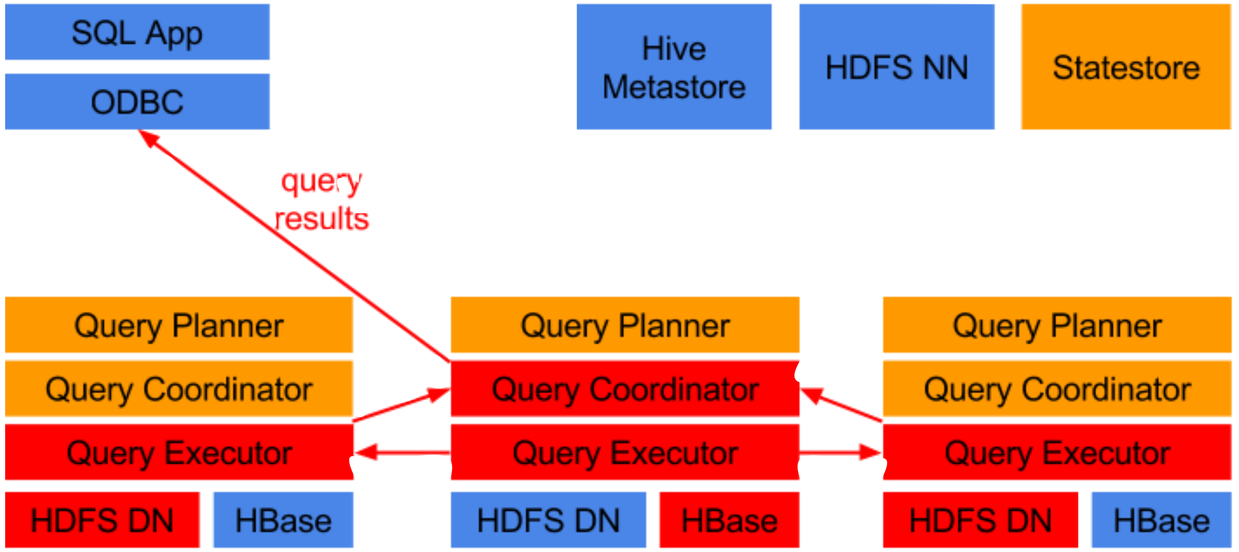
Impala Query Execution

- Planner turns request into collection of plan fragments
- Coordinator initiates execution on remote impalad nodes



Impala Query Execution

- Intermediate results are streamed between impala's
- Query results are streamed back to client



Impala Architecture: Query Planning

- 2-phase process:
  - single-node plan: left-deep tree of query operators
  - partitioning into plan fragments for distributed parallel execution:
    maximize scan locality/minimize data movement, parallelize all query operators
- cost-based join order optimization
- cost-based join distribution optimization

Impala Architecture: Query Execution

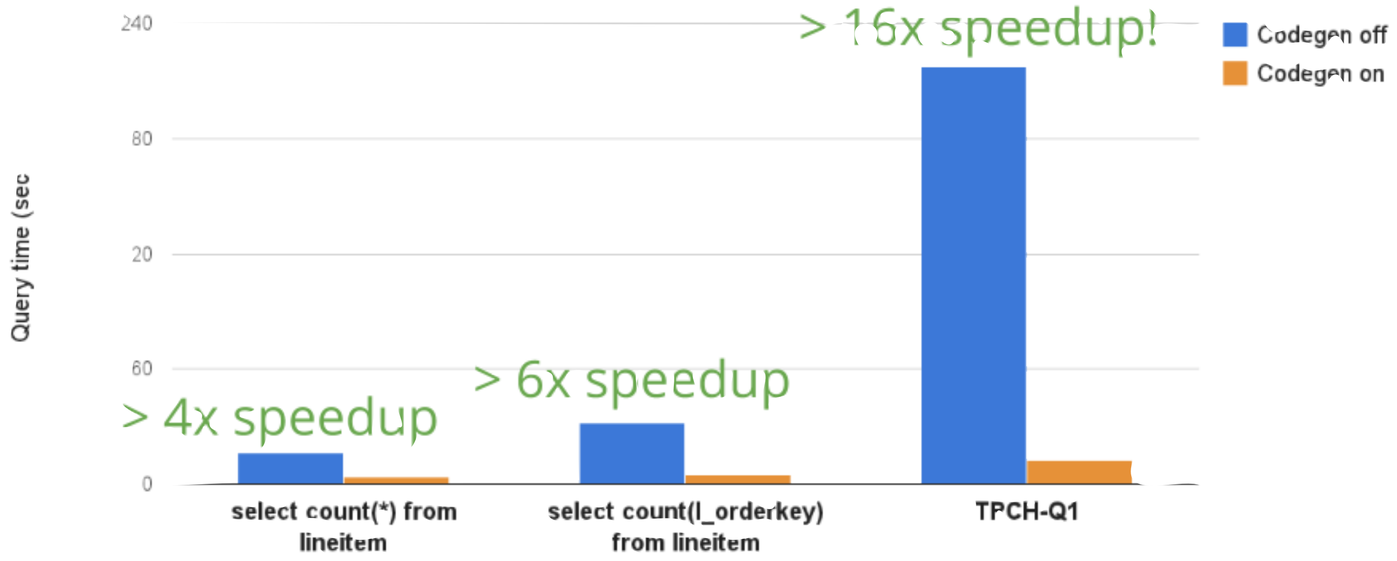
- execution engine designed for efficiency, written from scratch in C++; no reuse of decades-old open-source code
- circumvents MapReduce completely
- in-memory execution:
  - aggregation results and right-hand side inputs of joins are cached in memory
  - example: join with 1TB table, reference 2 of 200 cols, 10% of rows
    -> need to cache 1GB **across all nodes** in cluster
    -> not a limitation for most workloads

Impala Architecture: Query Execution

- runtime code generation:
  - uses llvm to jit-compile the runtime-intensive parts of a query
  - effect the same as custom-coding a query:
    - remove branches
    - propagate constants, offsets, pointers, etc.
    - inline function calls
  - optimized execution for modern CPUs (instruction pipelines)

Impala Architecture: Query Execution

Results

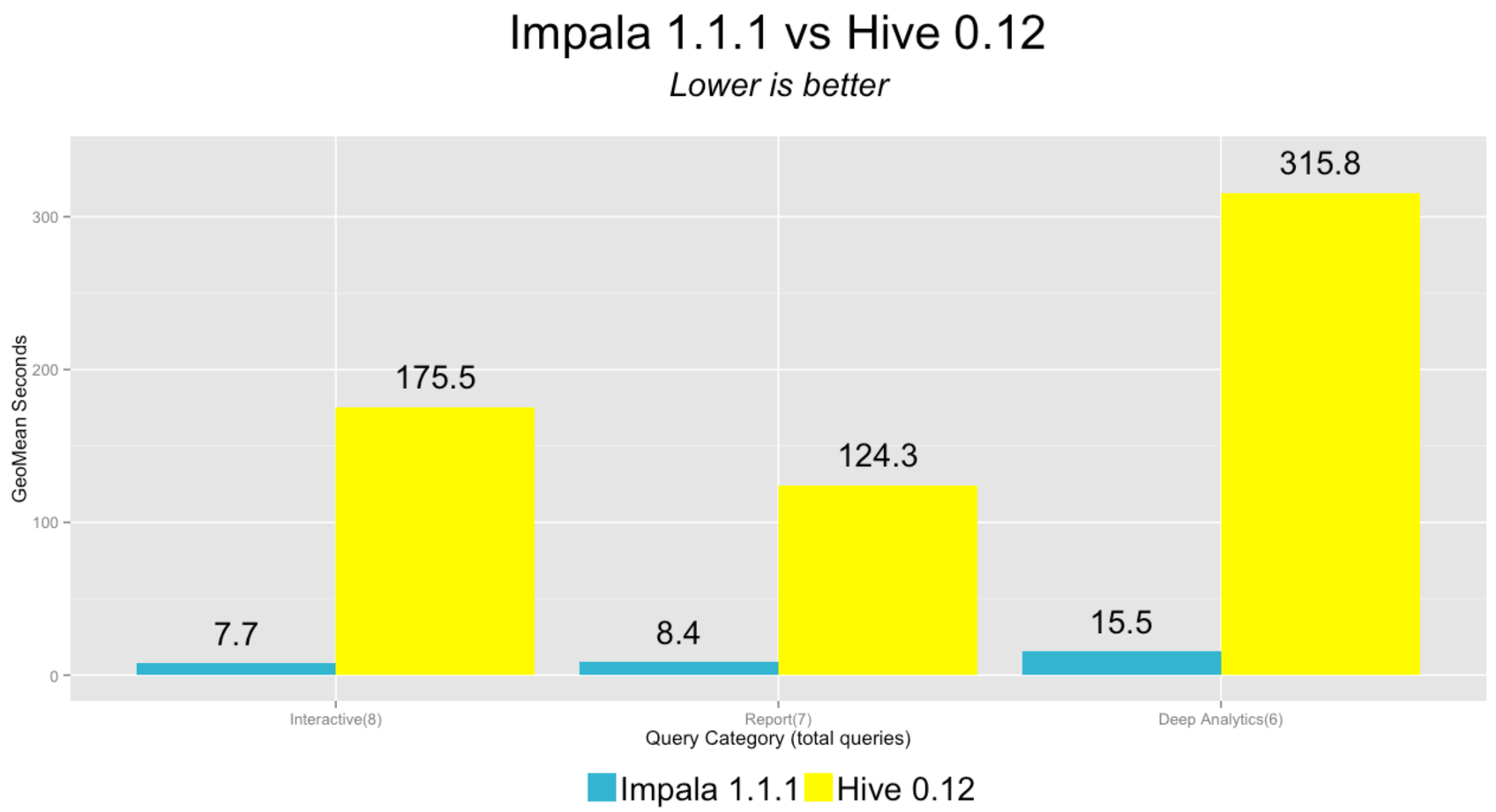


10 node cluster (12 disks / 48GB RAM / 8 cores per node)
40 GB / 60M row Avro dataset

Impala vs MR for Analytic Workloads

- Impala vs. SQL-on-MR
  - Impala 1.1.1/Hive 0.12 (“Stinger Phases 1 and 2”)
  - file formats: Parquet/ORCfile
  - TPC-DS, 3TB data set running on 5-node cluster

Impala vs MR for Analytic Workloads

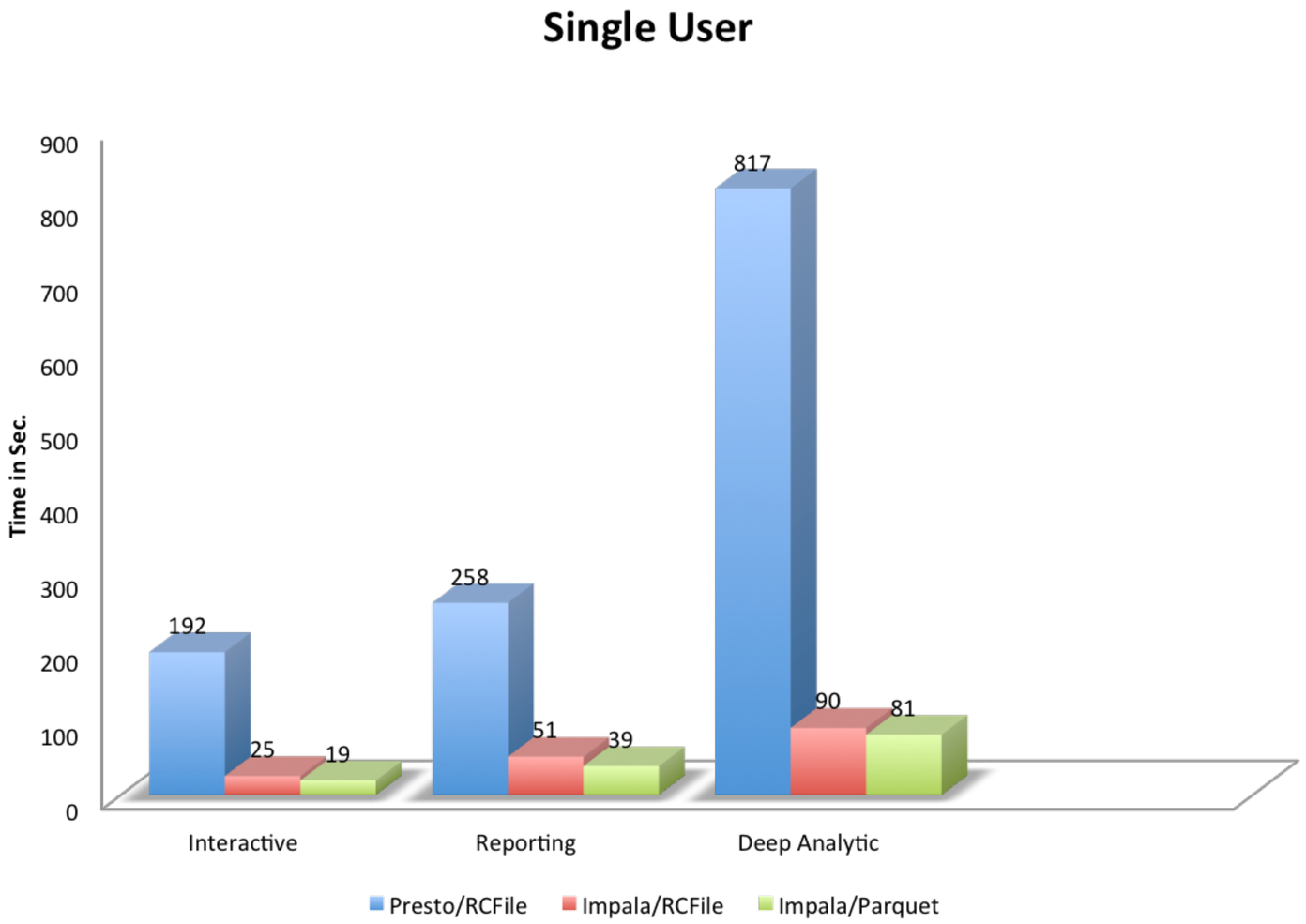


- Impala speedup:
  - interactive: 8–69x
  - report: 6–68x
  - deep analytics: 10–58x

Impala vs Presto for Analytic Workloads

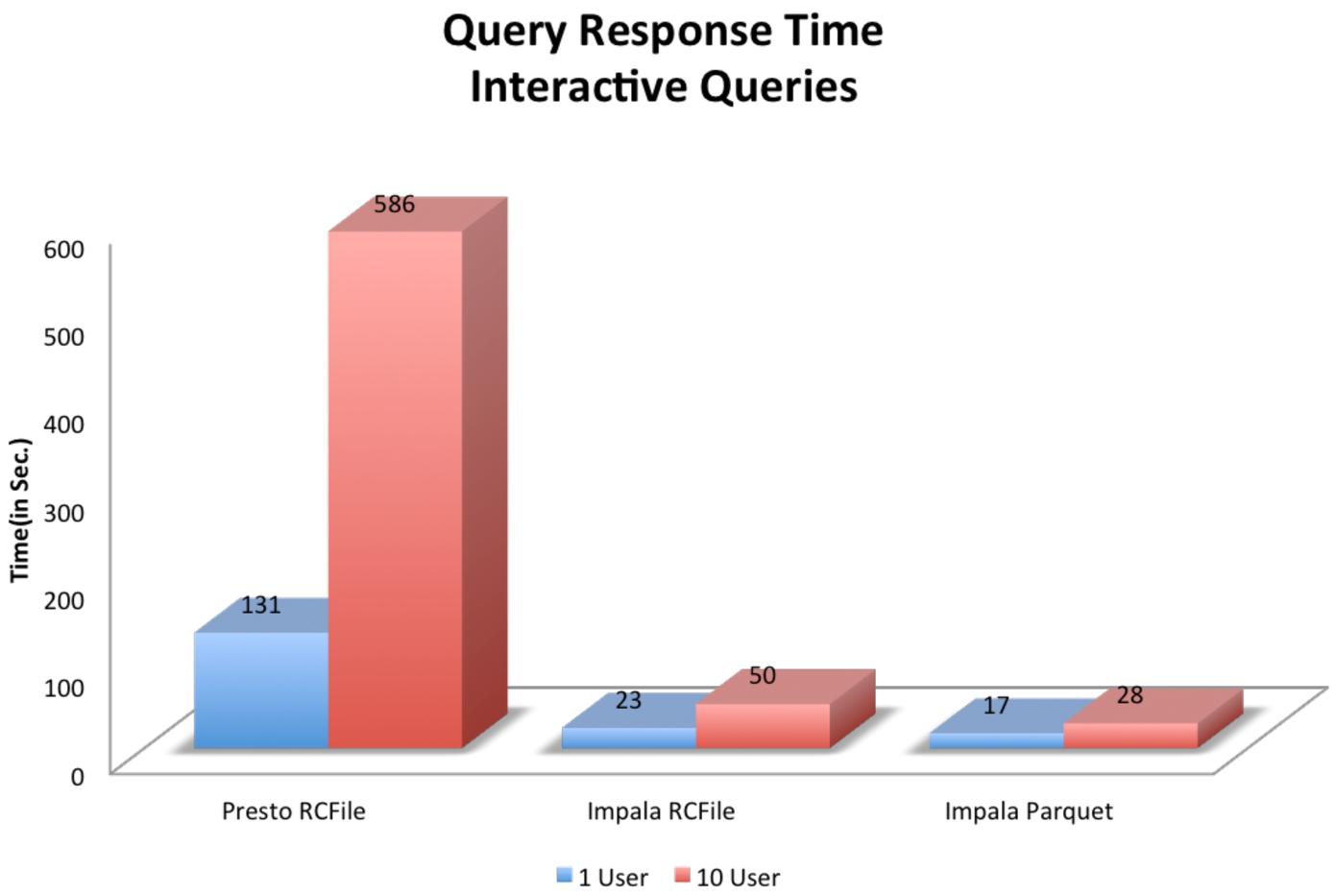
- Impala 1.2.3/Presto 0.54
- file formats: RCfile (+ Parquet)
- TPC-DS, 15TB data set running on 21-node cluster

Impala vs Presto for Analytic Workloads



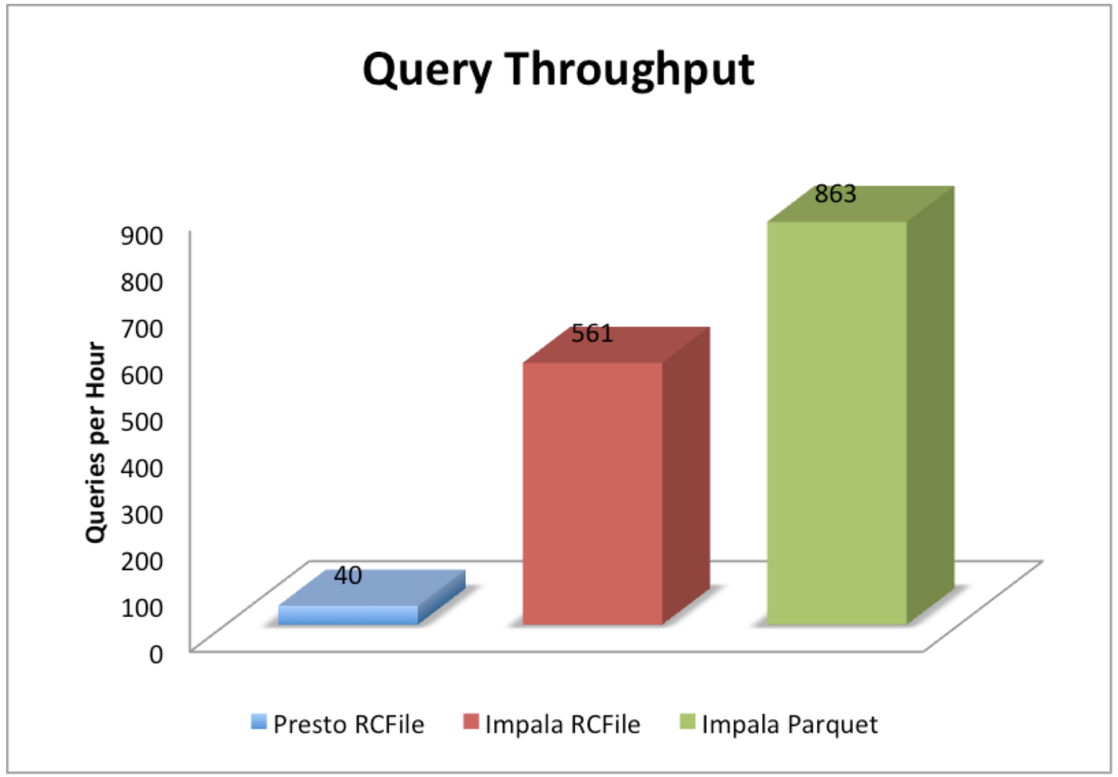
- Impala speedup:
  - interactive: 2-52x
  - report: 4-8x
  - deep analytics: 4-59x
  - total: 12x

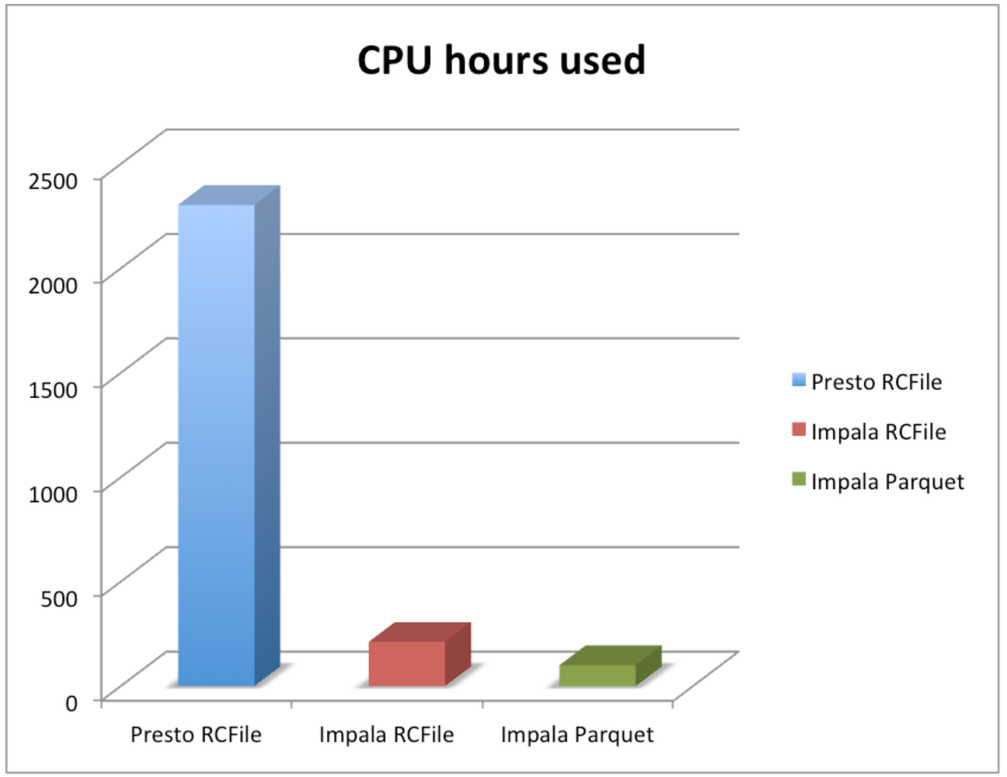
Impala vs Presto for Analytic Workloads



- Multi-user benchmark:
  - 10 users concurrently
  - same dataset, same hardware
  - workload: queries from “interactive” group

Impala vs Presto for Analytic Workloads





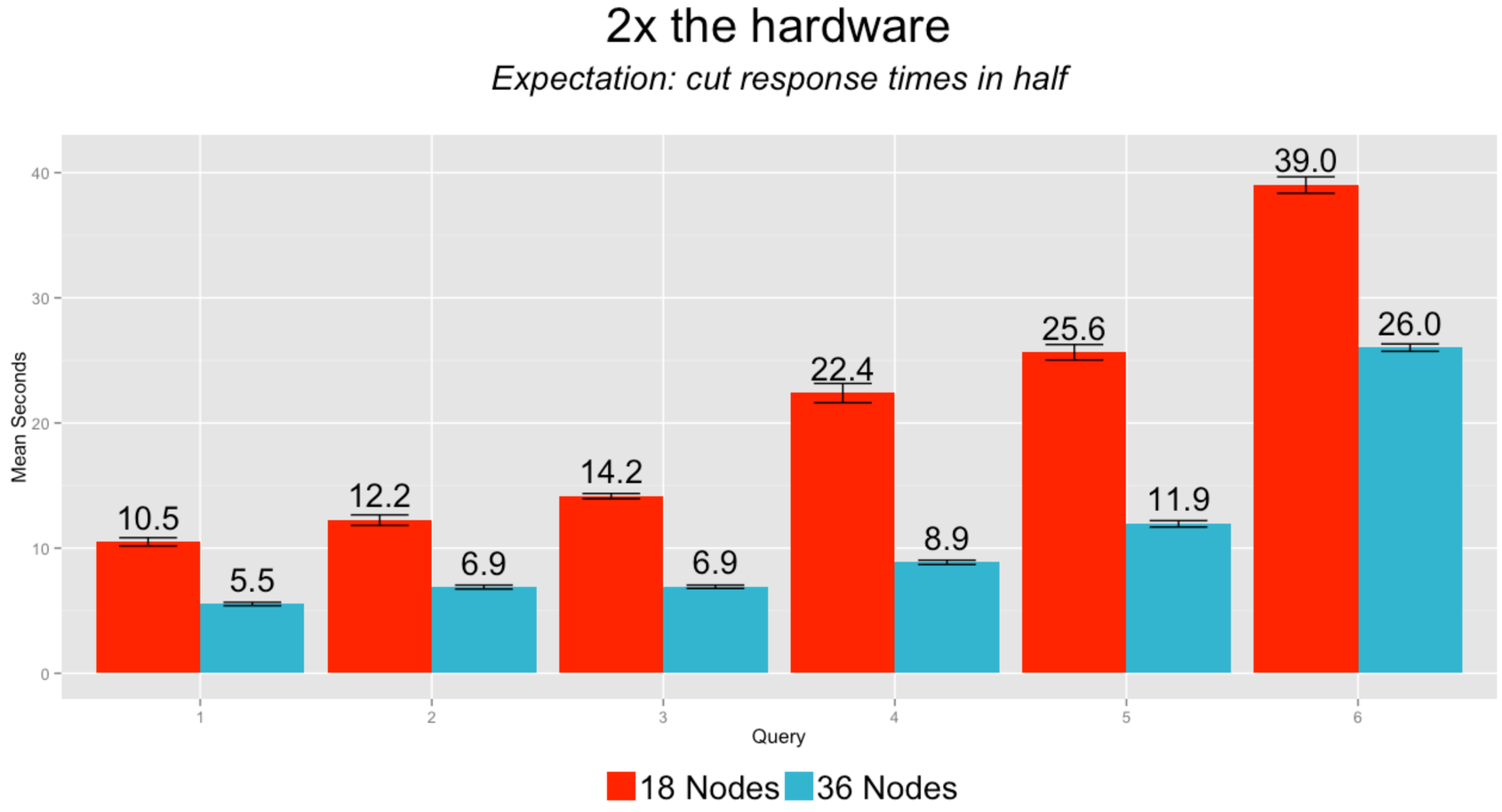
Scalability in Hadoop

- Hadoop's promise of linear scalability: add more nodes to cluster, gain a proportional increase in capabilities
  -> adapt to any kind of workload changes simply by adding more nodes to cluster
- scaling dimensions for EDW workloads:
  - response time
  - concurrency/query throughput
  - data size

Scalability in Hadoop

- Scalability results for Impala:
  - tests show linear scaling along all 3 dimensions
  - setup:
    - 2 clusters: 18 and 36 nodes
    - 15TB TPC-DS data set
    - 6 “interactive” TPC-DS queries

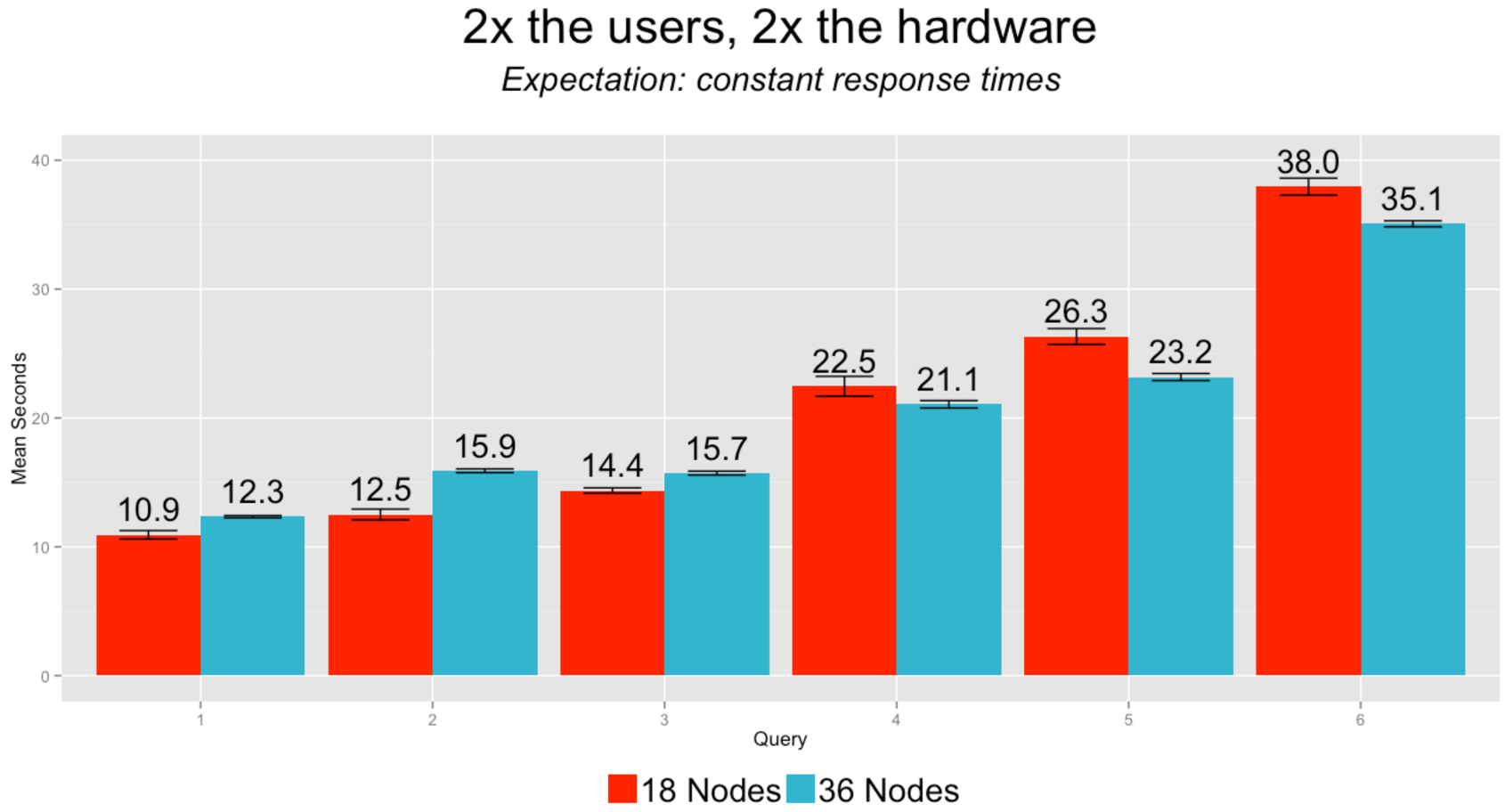
Impala Scalability: Latency



cloudera
Ask Bigger Questions

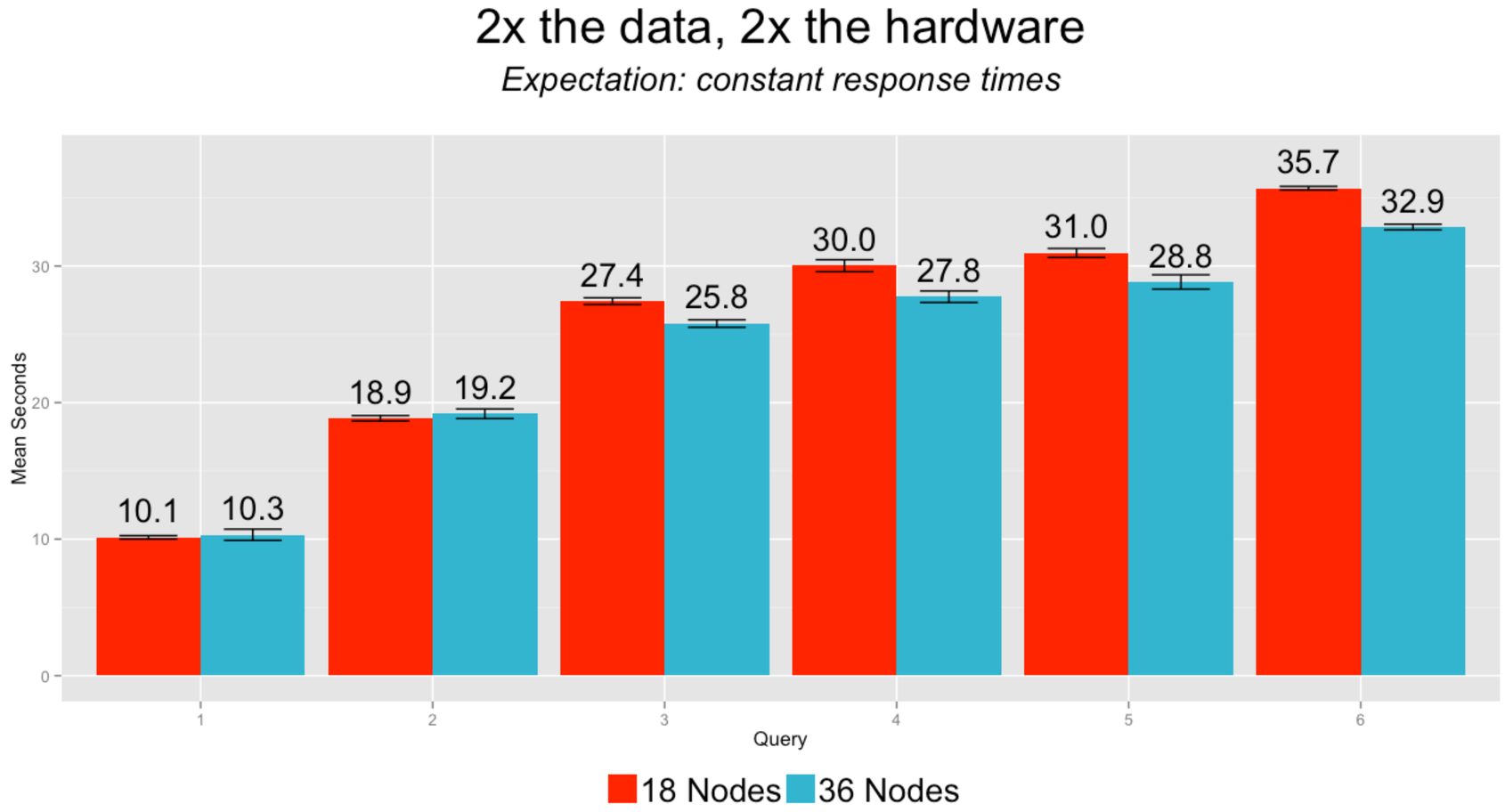
Impala Scalability: Concurrency

- Comparison: 10 vs 20 concurrent users



Impala Scalability: Data Size

- Comparison: 15TB vs. 30TB data set



Summary: Hadoop for Analytic Workloads

- Thesis of this talk:
  - techniques and functionality of established commercial solutions are either already available or are rapidly being implemented in Hadoop stack
  - Impala/Parquet/Hdfs is effective solution for certain EDW workloads
  - Hadoop-based EDW solution maintains Hadoop's strengths: flexibility, ease of scaling, cost effectiveness

Summary: Hadoop for Analytic Workloads

- latest technological innovations add capabilities that originated in high-end proprietary systems:
  - high-performance disk scans and memory caching in HDFS
  - Parquet: columnar storage for analytic workloads
  - Impala: high-performance parallel SQL execution

Summary: Hadoop for Analytic Workloads

- Impala/Parquet/Hdfs for EDW workloads:
  - integrates into BI environment via standard connectivity and security
  - comparable or better performance than commercial competitors
  - currently still SQL limitations
  - but those are rapidly diminishing

Summary: Hadoop for Analytic Workloads

- Impala/Parquet/Hdfs maintains traditional Hadoop strengths:
  - flexibility: Parquet is understood across the platform, natively processed by most popular frameworks
  - demonstrated scalability and cost effectiveness

The End

Summary: Hadoop for Analytic Workloads

- what the future holds:
  - further performance gains
  - more complete SQL capabilities
  - improved resource mgmt and ability to handle multiple concurrent workloads in a single cluster