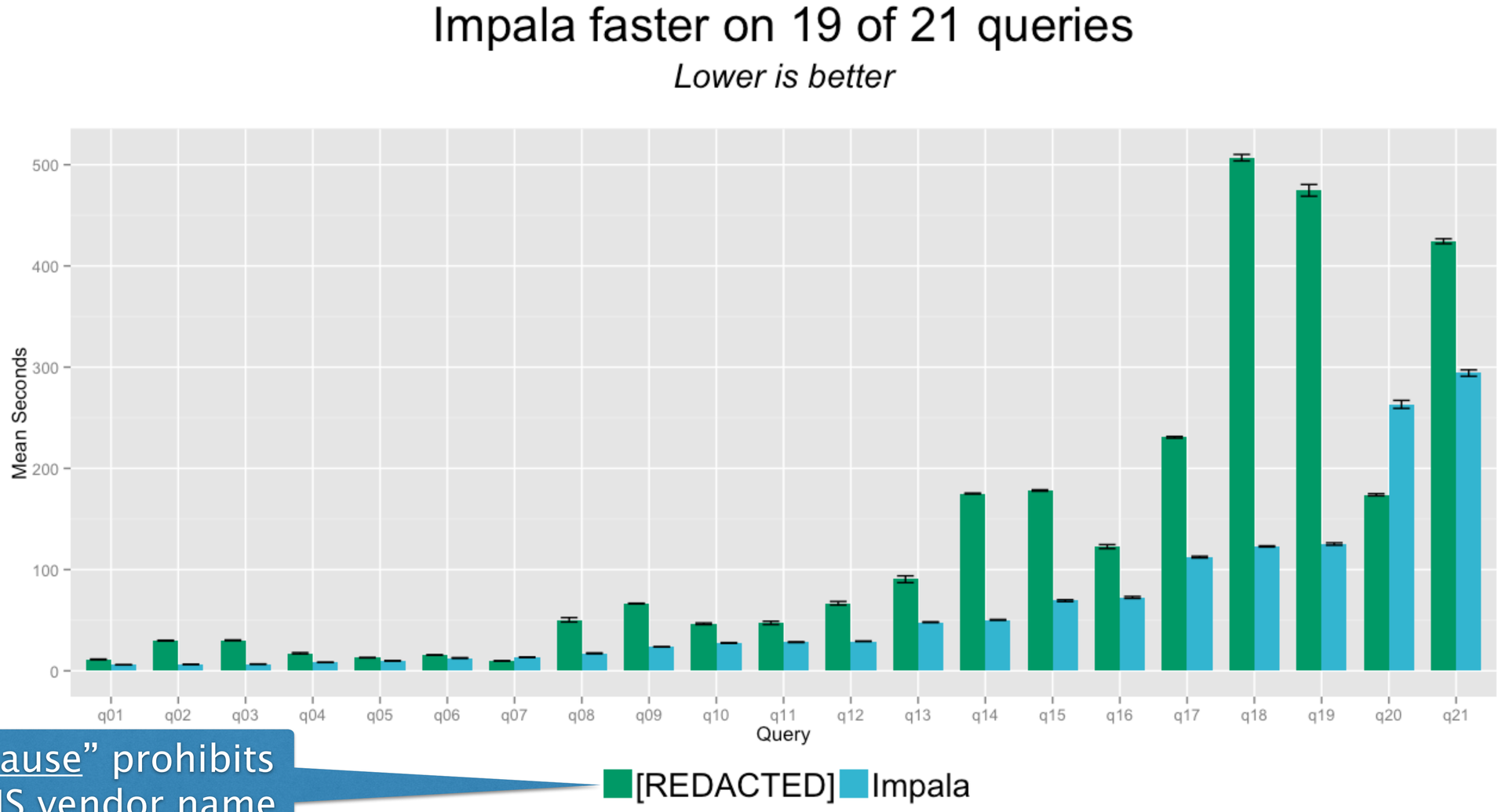
cloudera
Ask Bigger Questions

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

Marcel Kornacker | marcel@cloudera.com

2013-11-12

Analytic Workloads on Hadoop: Where Do We Stand?



"DeWitt Clause" prohibits using DBMS vendor name

cloudera
Ask Bigger Questions

Outline: Hadoop Technologies for Analytic Workloads

- HDFS: a storage system for analytic workloads
- Parquet: columnar storage
- Impala: a modern, open-source SQL engine for Hadoop

HDFS: A Storage System for Analytic Workloads

- Goal: high efficiency: data transfer at or near hardware speed
  - 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: A Storage System for Analytic Workloads

- 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

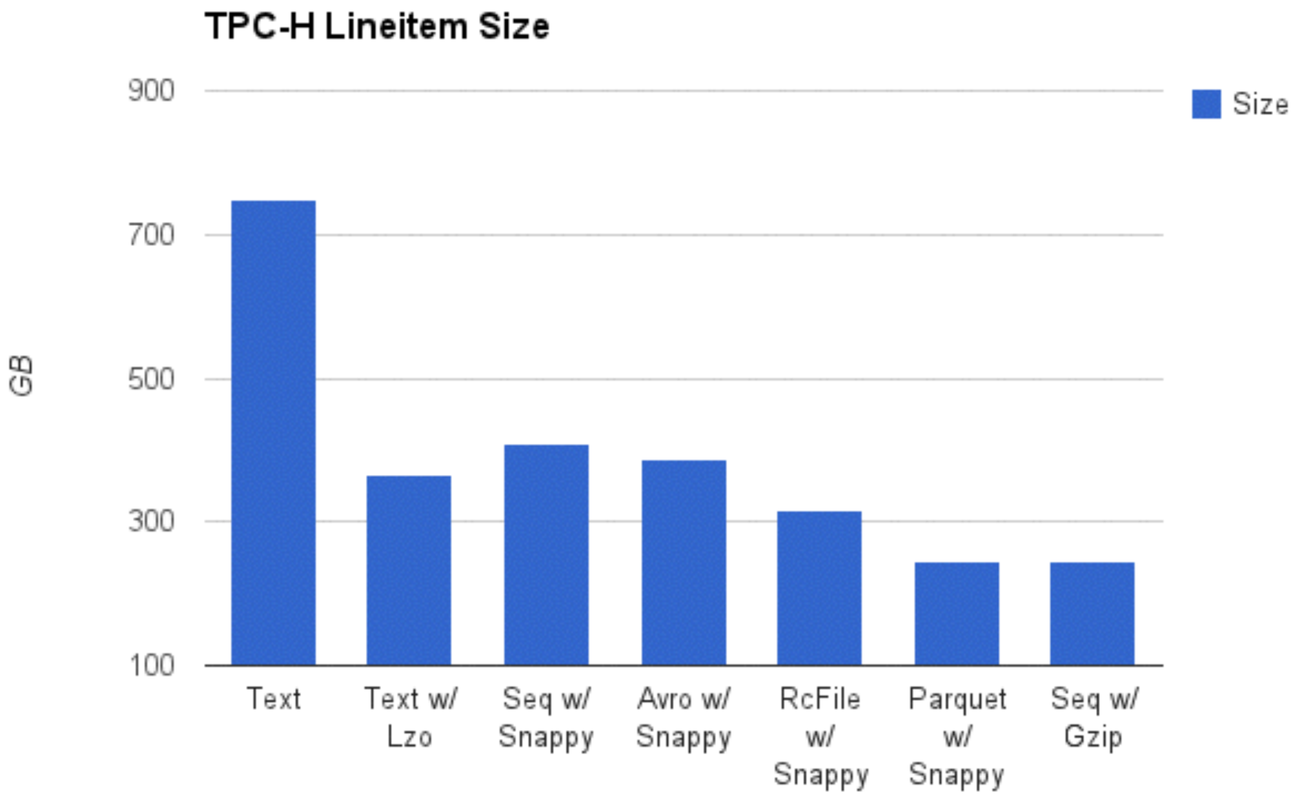
Parquet: Columnar Storage for Hadoop

- 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, …
  - co-developed by Twitter and Cloudera
  - with contributors from Criteo, Stripe, Berkeley AMPlab, LinkedIn
  - used in production at Twitter and Criteo

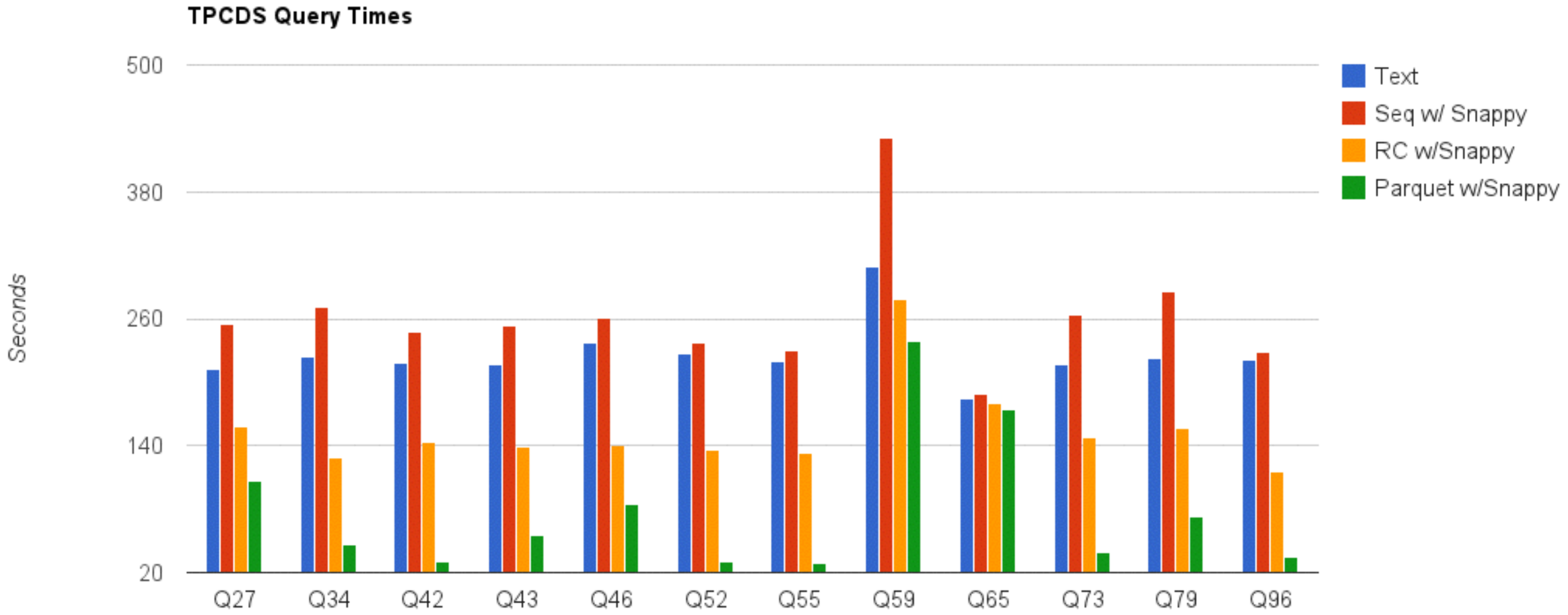
Parquet: The Details

- 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 a parallel SQL query engine for the Hadoop environment
- high-performance, modern execution engine, written in C++; developed by Cloudera and fully open-source
- utilizes standard Hadoop components (HDFS, Hbase, Metastore, Yarn)
- exposes/interacts with industry-standard interfaces (odbc/jdbc, Kerberos and LDAP, ANSI SQL)

Impala: A Modern, Open-Source SQL Engine

- history:
  - released as beta in 10/2012
  - 1.0 version available in 05/2013
  - current version is 1.2, available for CDH5 beta (version 1.2.1 for CDH4.5 to be released in a few weeks)

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, UDFs, UDAs
  - version 2.0: analytic window functions, UDTFs, SQL extensions for nested types (Avro/protocol buffer data)

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 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 in version 1.2.1 and later

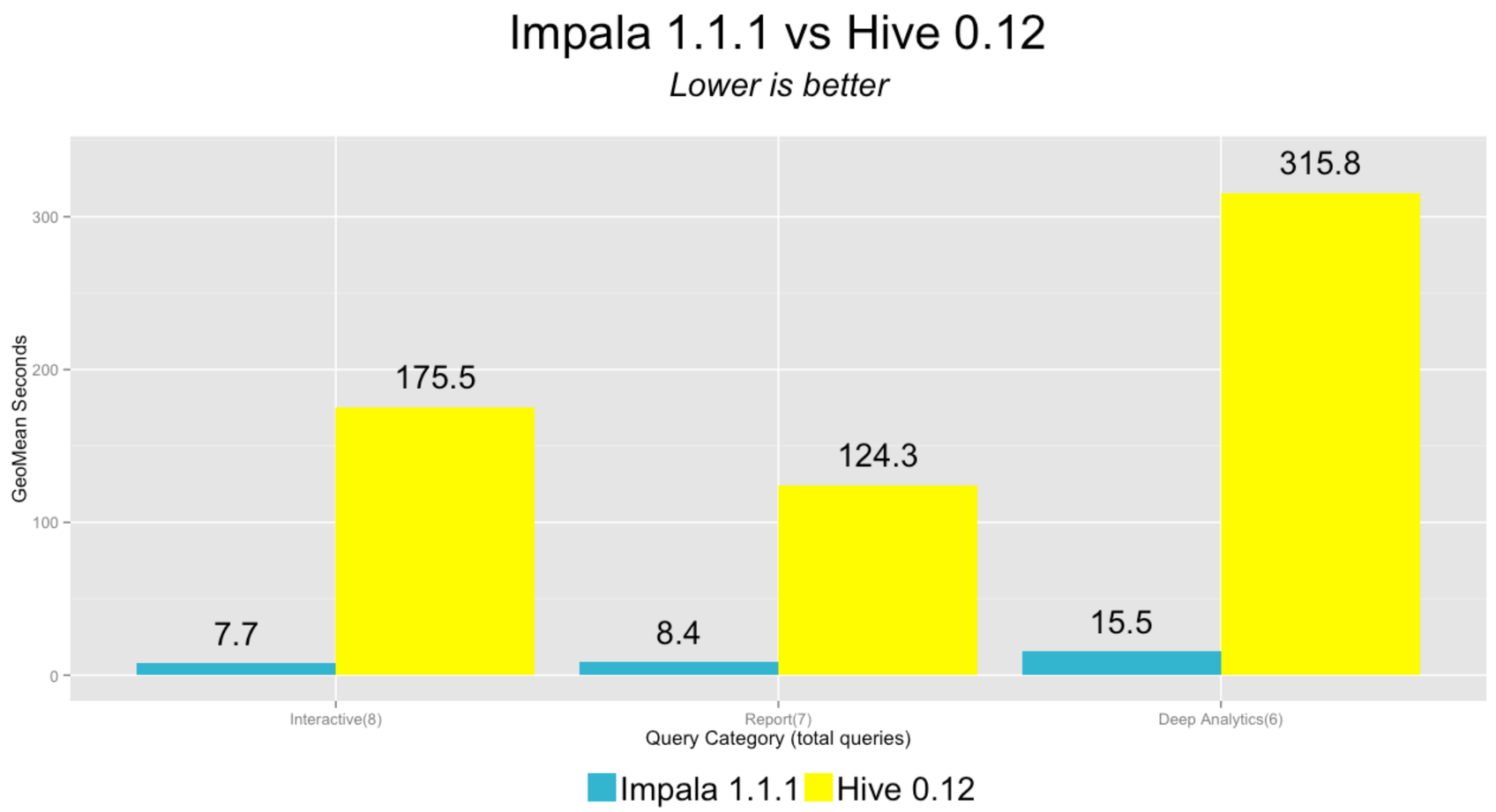
Impala Architecture: Query Execution

- circumvents MapReduce completely
- intermediate results are streamed directly between impala's, never hit disk
- query results are streamed back to client from coordinator
- in-memory execution:
  - right-hand side inputs of joins and aggregation results 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 vs MR for Analytic Workloads

- Impala vs. SQL-on-MR
  - Impala 1.1.1/Hive 0.12
  - 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

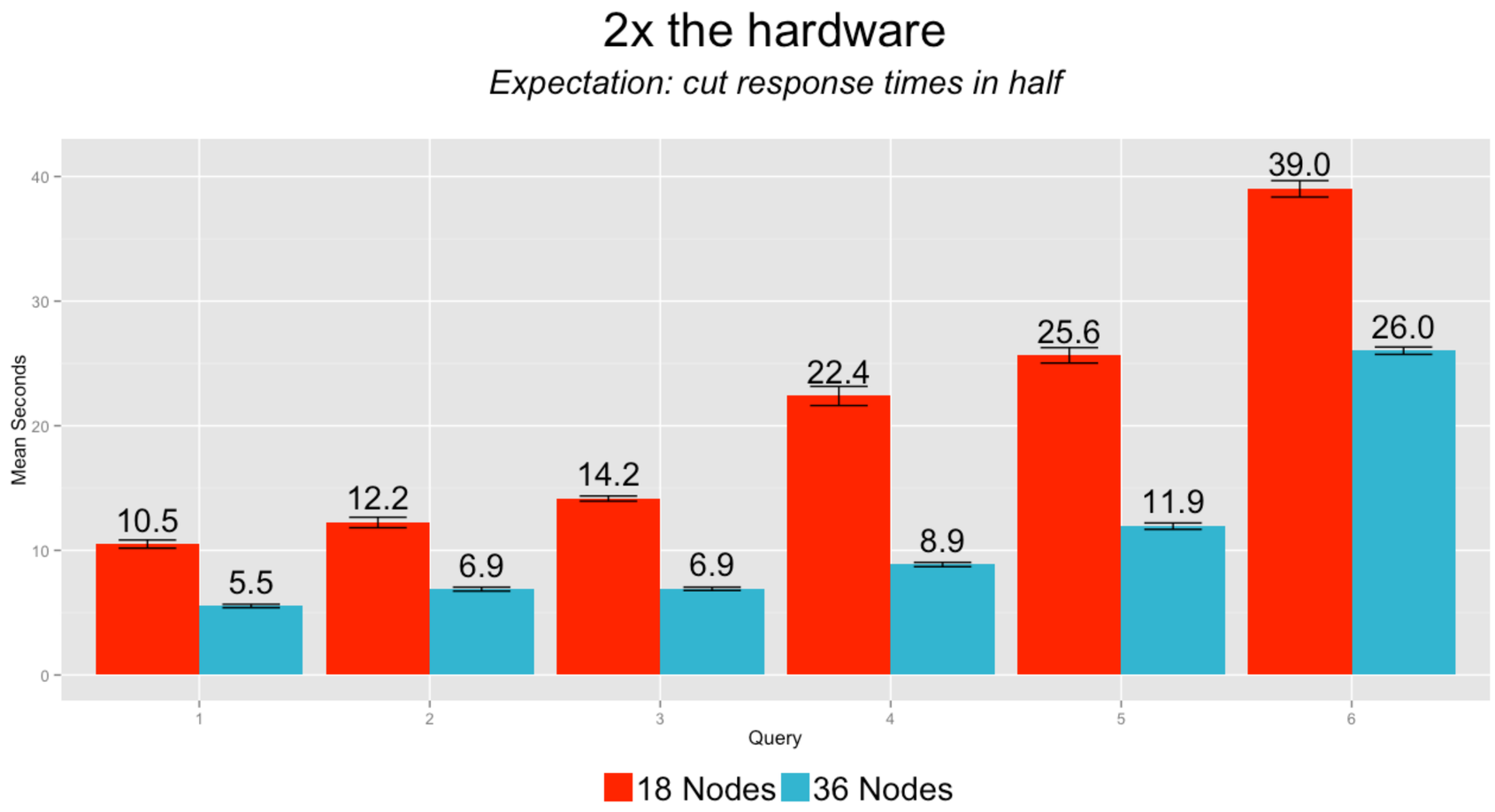
Scalability in Hadoop

- dimensions of scalability:
  - response time scaling
  - concurrency scaling
  - data size scaling
- 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

Impala Scalability: Latency

- goal: cut response time in half by doubling the number of nodes
- test setup:
  - 2 clusters: 18 and 36 nodes
  - 15 TB TPC-DS data set

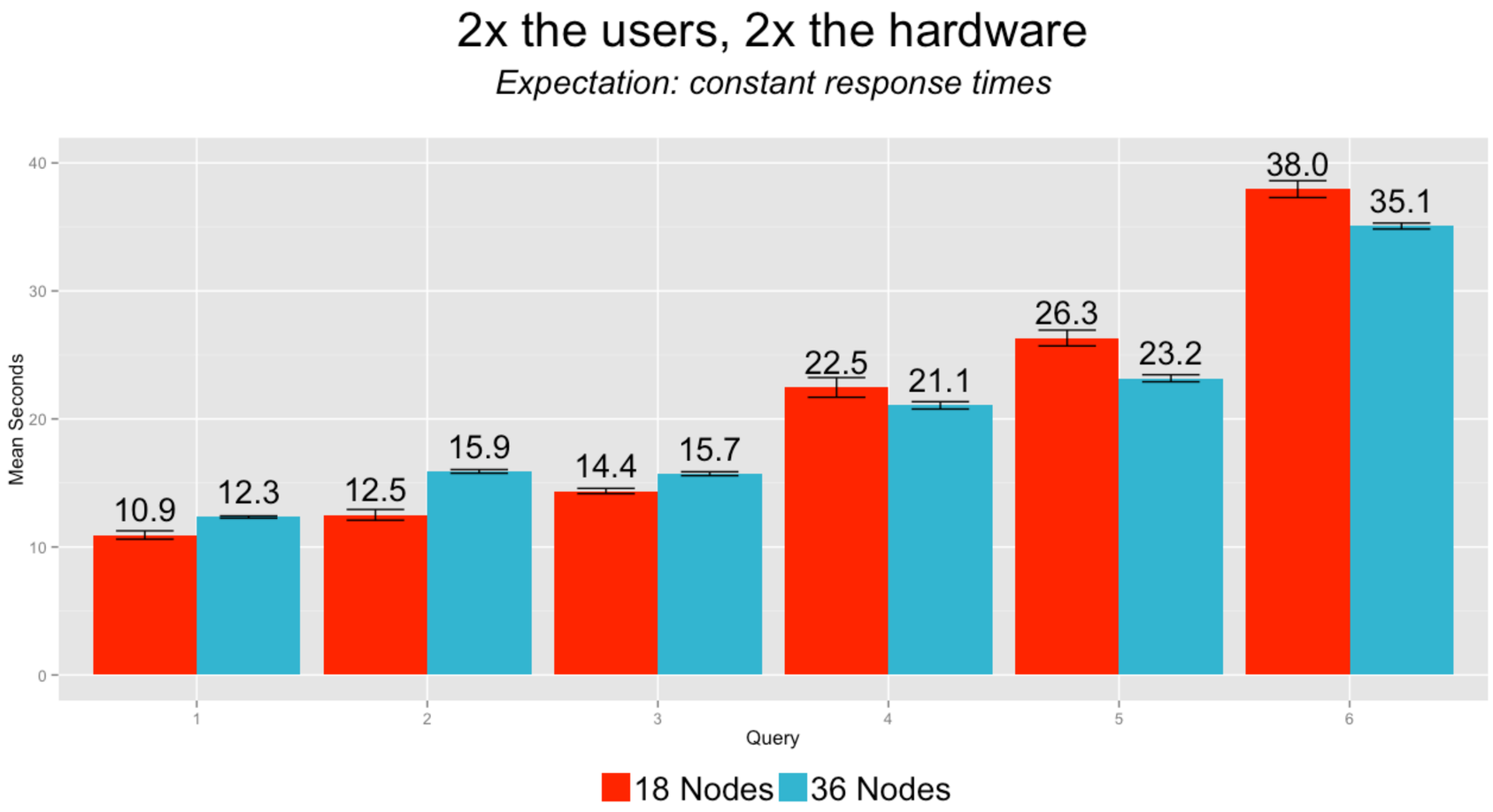
Impala Scalability: Latency



Impala Scalability: Concurrency

- goal: handle 2x concurrent users with 2x nodes (without response time degradation)
- test setup: same as before, with 10 and 20 concurrent users

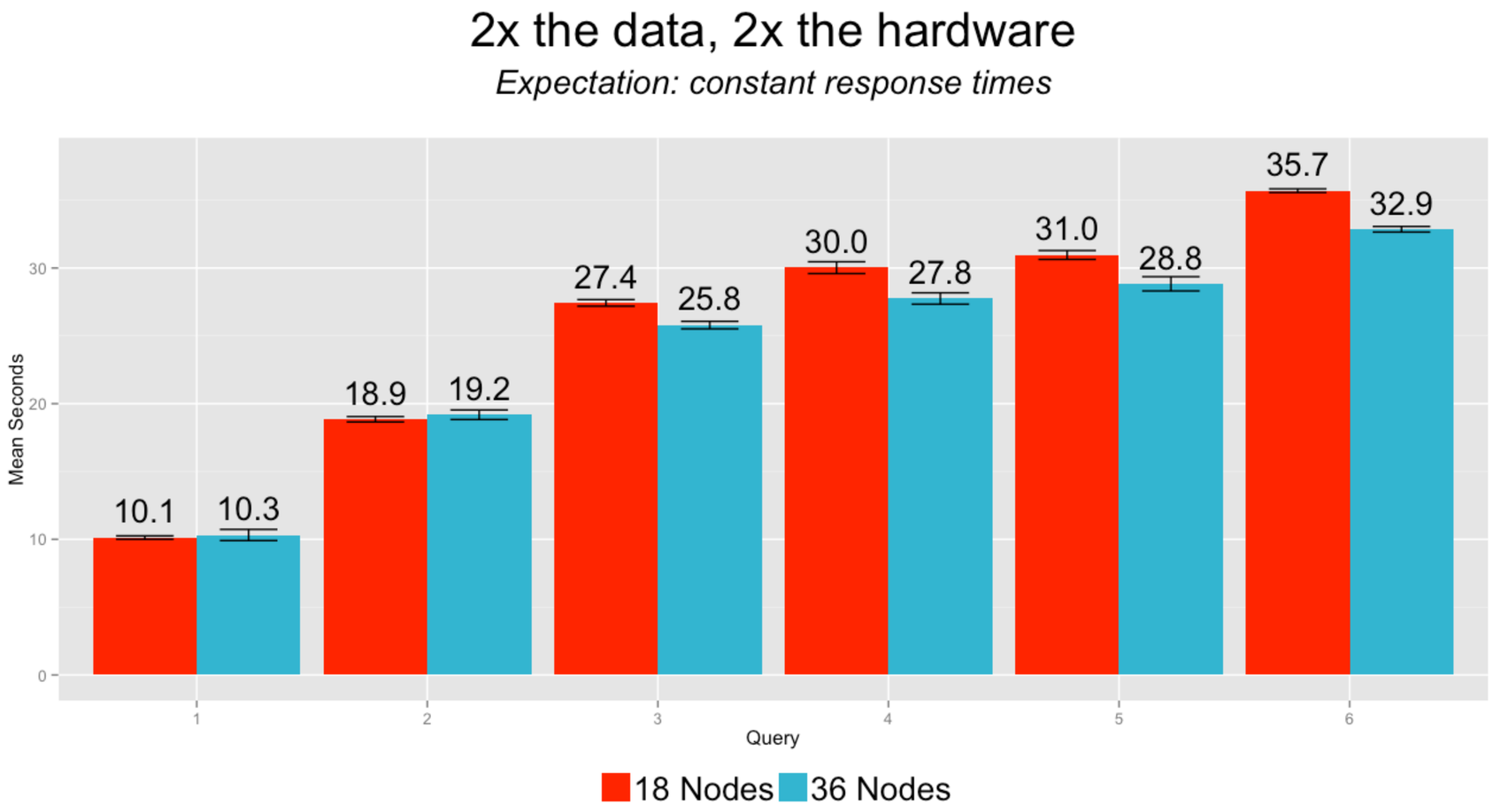
Impala Scalability: Concurrency



Impala Scalability: Data Size

- goal: query 2x data size with 2x hardware (without response time degradation)
- test setup:
  - 18- and 36-node clusters
  - with 15TB and 30TB TPC-DS data sets

Impala Scalability: Data Size



cloudera
Ask Bigger Questions

Summary: Hadoop for Analytic Workloads

- Hadoop has traditional been utilized for offline batch processing: ETL and ELT
- 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

- initial results are encouraging:
  - Hadoop-based open-source stack offers performance that is competitive with or better than top-5 proprietary analytic DBMS solution
  - maintains Hadoop's strengths: flexibility, easy of scaling, cost effectiveness
- 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

The End