cloudera
Ask Bigger Questions

Cloudera Impala: A Modern SQL Engine for Hadoop

Justin Erickson | Senior Product Manager, Cloudera

May 2013

Agenda

- Why Impala?
- Architectural Overview
- Alternative Approaches
- Project Status

Why Hadoop?

- **Scalability**
  - Simply scales just by adding nodes
  - Local processing to avoid network bottlenecks
- **Flexibility**
  - All kinds of data (blobs, documents, records, etc)
  - In all forms (structured, semi-structured, unstructured)
  - Store anything ***then later*** analyze/process what you need
  - Analyze/process the data how you need to
- **Efficiency**
  - Cost efficient software on commodity hardware
  - Unified storage, metadata, security (no duplication or synchronization)

What's Impala?

- **Interactive SQL**
  - Typically 5-65x faster than Hive (observed up to 100x faster)
  - Responses in seconds instead of minutes (sometimes sub-second)
- **Approx. ANSI-92 standard SQL queries with HiveQL**
  - Compatible SQL interface for existing Hadoop/CDH applications
  - Based on industry standard SQL
- **Natively on Hadoop/HBase storage and metadata**
  - Flexibility, scale, and cost advantages of Hadoop
  - No duplication/synchronization of data and metadata
  - Local processing to avoid network bottlenecks
- **Separate runtime from MapReduce**
  - MapReduce is designed and great for batch
  - Impala is purpose-built for low-latency SQL queries on Hadoop

So what?

- **Interactive BI/analytics**
  - BI tools impractical on Hadoop before Impala
  - Move from 10s of Hadoop users per cluster to 100s of SQL users
  - More and faster value from “big data”
- **Exploratory analytics**
  - Ask and explore new questions on raw, granular data
  - No upfront ETL process to access data
- **Queryable archive with full fidelity**
  - Keep historical data active in Hadoop instead of inaccessible tape
- **Data processing with tight SLAs**
  - Sub-minute SLAs now possible

Impala Architecture

- Two binaries: impalad and statestored
- **Impala daemon (impalad)**
  - one Impala daemon on each node with data
  - handles external client requests and all internal requests related to query execution
- **State store daemon (statestored)**
  - provides name service and metadata distribution
  - not part of query execution path

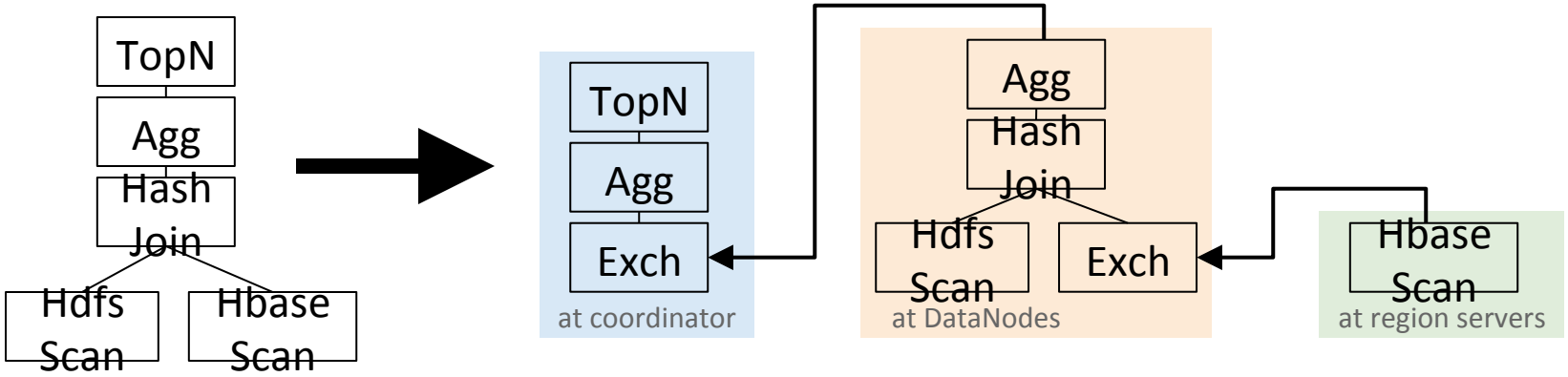
Impala Architecture: Query Execution Phases

- **Client** SQL arrives via ODBC/JDBC/Hue GUI/Shell
- **Planner** turns request into collections of plan fragments
- **Coordinator** initiates execution on impalad's local to data
- During **execution**:
  - intermediate results are streamed between executors
  - query results are streamed back to client
  - subject to limitations imposed to blocking operators (top-n, aggregation)

Impala Architecture: Planner

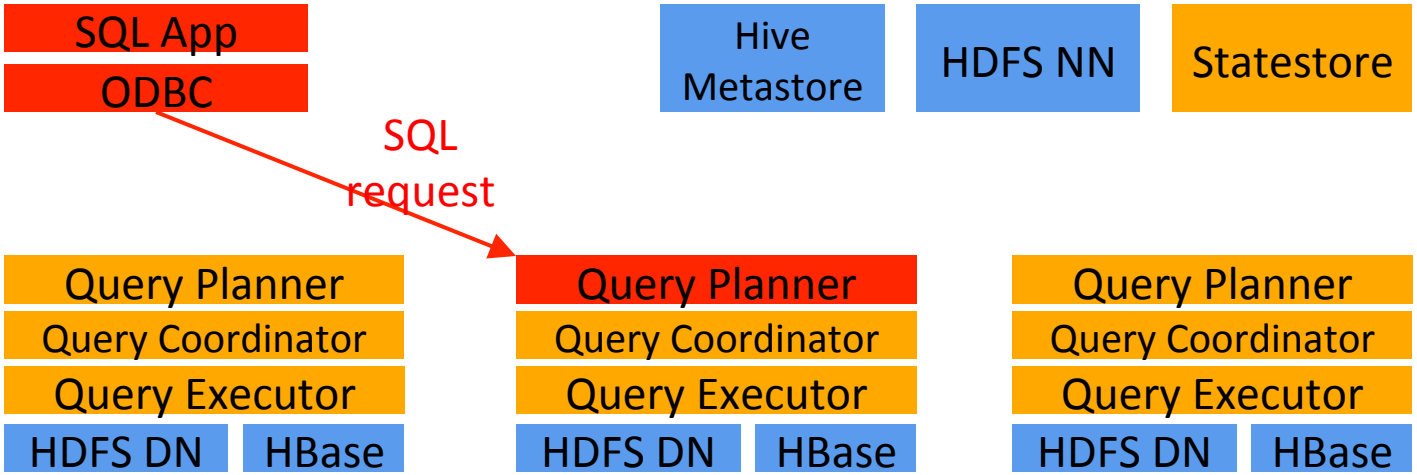
- Example: query with join and aggregation

```
SELECT state, SUM(revenue)
FROM HdfsTbl h JOIN HbaseTbl b ON (...)
GROUP BY 1 ORDER BY 2 desc LIMIT 10
```



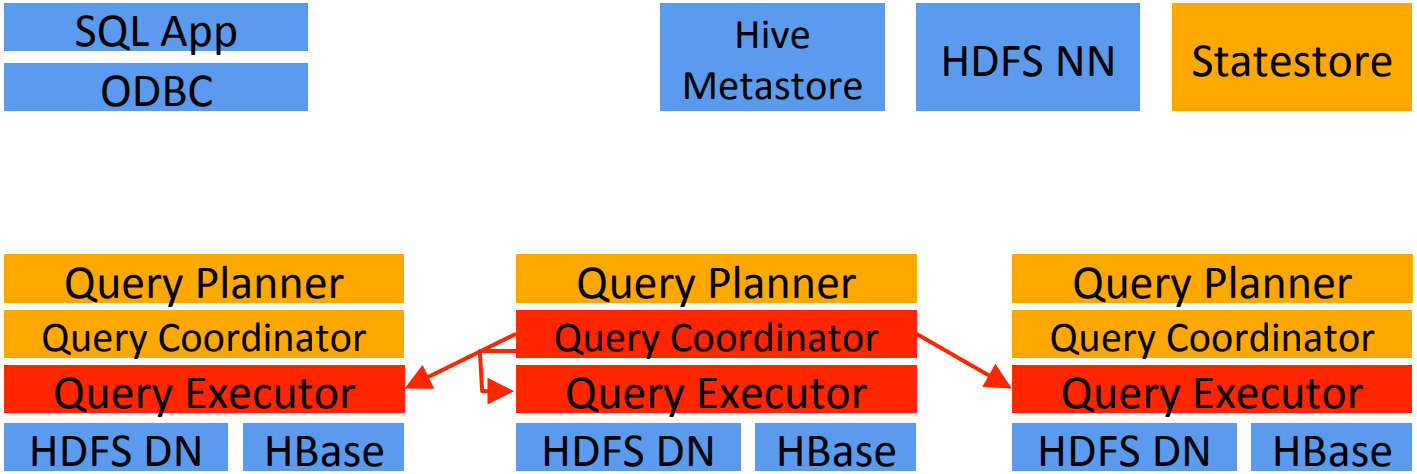
Impala Architecture: Query Execution

- Request arrives via ODBC/JDBC/Hue GUI/Shell



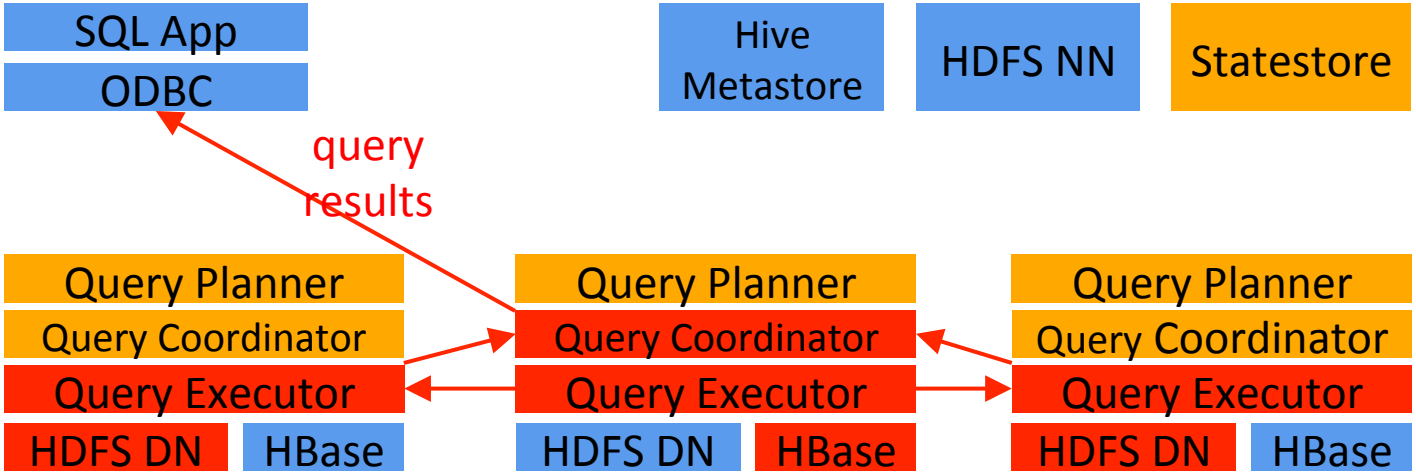
Impala Architecture: Query Execution

- Planner turns request into collections of plan fragments
- Coordinator initiates execution on impalad's local to data



Impala Architecture: Query Execution

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



Impala and Hive

- Everything Client-Facing is Shared with Hive:
  - Metadata (table definitions)
  - ODBC/JDBC drivers
  - Hue GUI
  - SQL syntax (HiveQL)
  - Flexible file formats
  - Machine pool

- Internal Improvements:
  - Purpose-built query engine direct on HDFS and HBase
  - No JVM startup and no MapReduce
  - In-memory data transfers
  - Modern tech including special hardware instructions, runtime code generation, etc
  - Native distributed relational query engine

What about an EDW/RDBMS?

- ***"Right tool for the right job"***

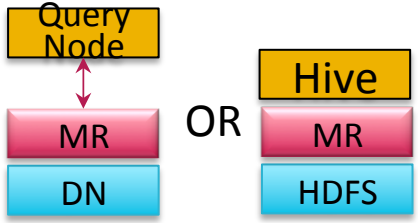
- EDW/RDBMS great for:
  - OLTP's complex transactions
  - Highly planned and optimized known workloads
  - ***Operational reports and drill into repeated known queries***

- Impala's great for:
  - ***Exploratory analytics with new previously unknown queries***
  - Queries on big and growing data sets

- EDW/RDBMS can't:
  - Dump in raw data ***then later*** define schema and query what you want
  - Evolve schemas without an expensive schema upgrade planning process
  - Simply scale just by adding nodes
  - Store at a $/TB conducive to big data

Alternative Hadoop Query Approaches

Batch MapReduce

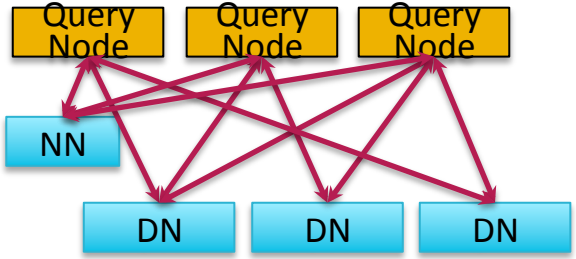


High-latency MR

Separate nodes for SQL/MR

Duplicate metadata, security, SQL, MR, etc.

Remote Query

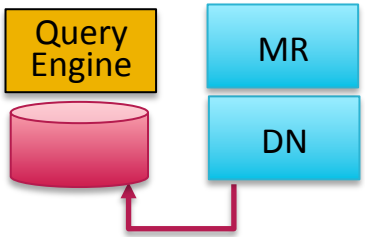


Network bottleneck

Separate nodes for SQL/MR

Duplicate metadata, security, SQL, MR, etc.

Siloed DBMS



Query subset of siloed data

Traditional RDBMS rigidity

Duplicate storage, metadata, security, SQL, etc.

cloudera
Ask Bigger Questions

Comparing Impala to Google Dremel

- What is Google Dremel:
  - columnar storage for data with nested structures
  - distributed scalable aggregation on top of that
- Columnar storage in Hadoop: Parquet
  - joint project between Cloudera and Twitter
  - new columnar format, derived from Doug Cutting's Trevni
  - stores data in appropriate native/binary types
  - can also store nested structures similar to Dremel's ColumnIO
- Distributed aggregation: Impala
- Impala + Parquet: superset of the published version of Dremel adding:
  - Joins
  - Multiple file format support

Impala GA (Beginning of Q2 2013)

- ~ANSI-92 SQL
  - CREATE, ALTER, SELECT, INSERT, JOIN, subqueries, etc.
- Native Hadoop data formats:
  - Avro, SequenceFile, RCFile with Snappy, GZIP, BZIP, or uncompressed
  - Text (uncompressed or LZO-compressed)
  - Parquet columnar format with Snappy or uncompressed
- Full CDH 4 64-bit packages:
  - RHEL 6.2/5.7, Ubuntu, Debian, SLES
- Connectivity:
  - JDBC, ODBC, Hue GUI, command-line shell
- Performance:
  - Bigger and faster joins (partitioned joins)
  - Fully distributed aggregations
  - Fully distributed top-n queries
  - More optimized SQL functions
- Production-readiness:
  - Kerberos authentication
  - MR/Impala resource isolation

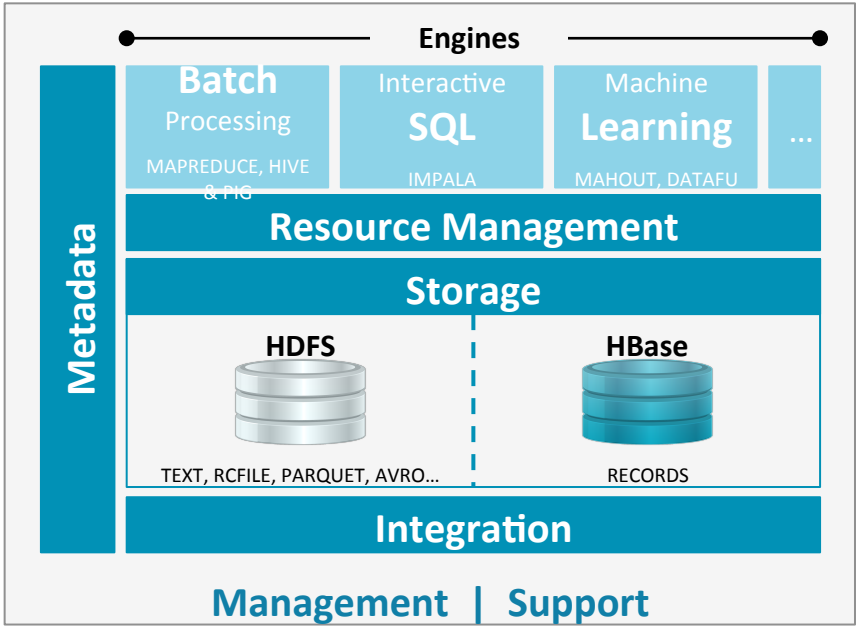
Impala Post-GA Roadmap

- Security
  - Authorization
  - Auditing
  - LDAP/Active Directory username/password authentication
- Additional SQL:
  - UDFs
  - SQL authorization and DDL
  - ORDER BY without LIMIT
  - window functions
  - support for structured data types
- Improved HBase support:
  - CREATE/INSERT/UPDATE/DELETE
  - composite keys, complex types in columns
  - index nested-loop joins
- Continued performance gains:
  - straggler handling
  - join order optimization
  - improved cache management
  - data collocation for improved join performance
- Better metadata handling:
  - automatic metadata distribution through statestore
- Continued resource control enhancements with MapReduce

It's Not Just About SQL on Hadoop

The Platform for Big Data

- Single platform for processing & analytics
- Scales to '000s of servers
- No upfront schema
- 10% the cost per TB
- Open source platform



Try It Out!

- Impala 1.0 GA released 4/30/2013
- 100% Apache-licensed open source
- Questions/comments?
  - Email: impala-user@cloudera.org
  - Join: http://groups.cloudera.org


cloudera®
Ask Bigger Questions