Cloudera Impala: A Modern SQL Engine for Hadoop

Justin Erickson | Senior Product Manager
February 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 4-35x 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”

• Data processing with tight SLAs
  • Sub-minute SLAs now possible

• Cost efficiency
  • Fewer nodes to meet response time SLAs
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

MapReduce

- **Query Node**
- **Hive**
- **MR**
- **DN**
- **HDFS**

- **High-latency MR**
- **Separate nodes for SQL/MR**
- **Duplicate metadata, security, SQL, MR, etc.**

Remote Query

- **Query Node**
- **Query Node**
- **Query Node**
- **NN**
- **DN**
- **DN**
- **DN**

- **Network bottleneck**
- **Separate nodes for SQL/MR**
- **Duplicate metadata, security, SQL, MR, etc.**

Side Storage

- **Query Engine**
- **MR**
- **DN**

- **Traditional RDBMS rigidity**
- **Query subset of data**
- **Duplicate storage, metadata, security, SQL, etc.**

©2013 Cloudera, Inc. All Rights Reserved.
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
More about Parquet

• What it is:
  • general Hadoop file format
  • columnar container format for all popular serialization formats: Avro, Thrift, Protocol Buffers
  • successor to Trevni
  • jointly developed between Cloudera and Twitter
  • open source; hosted on github

• Features:
  • rowgroup format: file contains multiple horiz. slices
  • supports storing each column in separate file
  • supports fully shredded nested data; repetition and definition levels similar to Dremel's ColumnIO
  • column values stored in native types (bool, int<x>, float, double, byte array)
  • support for index pages for fast lookup
  • extensible value encodings
Try It Out!

- Apache licensed open source
- Public beta version available since 10/24/12
- Latest version is 0.6 (as of 2/26)
- Questions/comments? impala-user@cloudera.org