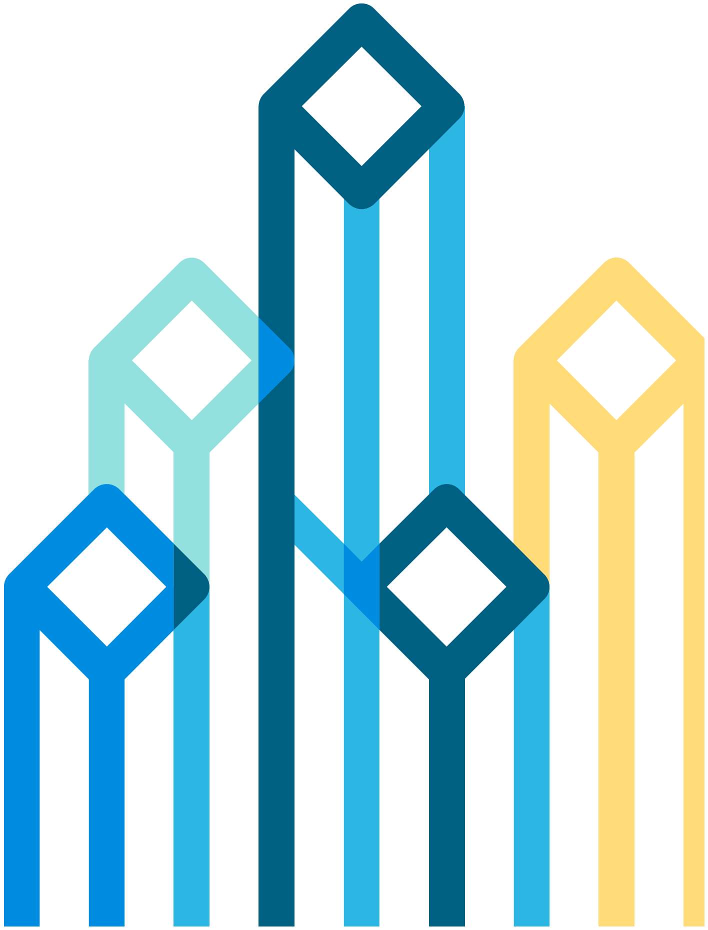
cloudera

SQL Engines for Hadoop – The case for Impala

Budapest Data Forum, June 4th, 2015

tiny.cloudera.com/mark-sql-budapest

Mark Grover | @mark_grover



cloudera

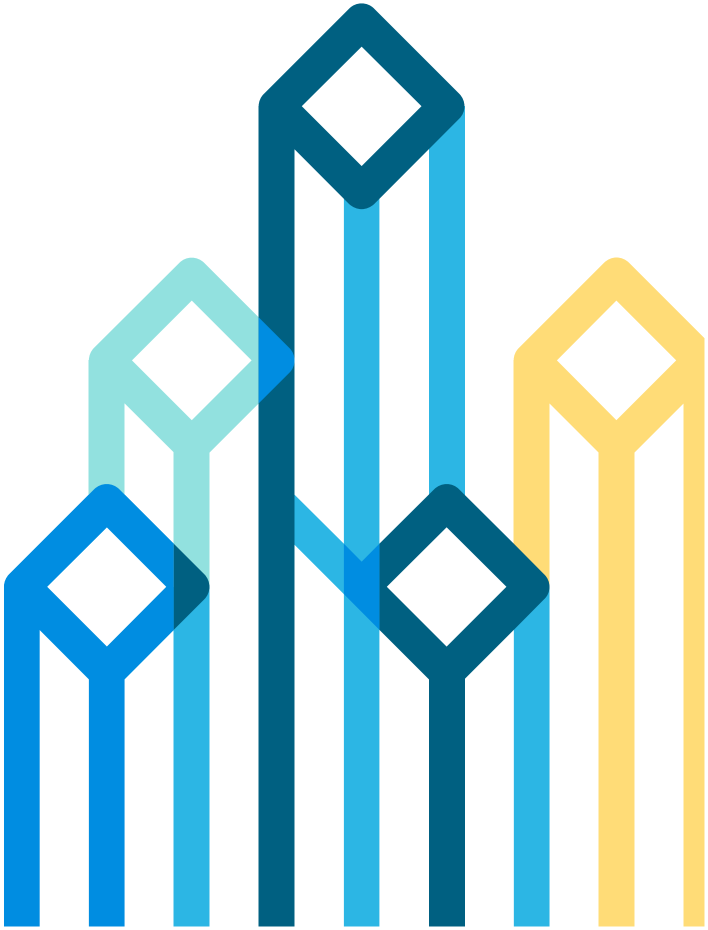
SQL Engines for Hadoop – The case for Impala

Budapest Data Forum, June 4th, 2015

tiny.cloudera.com/mark-sql-budapest

~~Mark Grover~~ | @mark_grover

Grover Mark



cloudera®

SQL Engines for Hadoop – The case for Impala

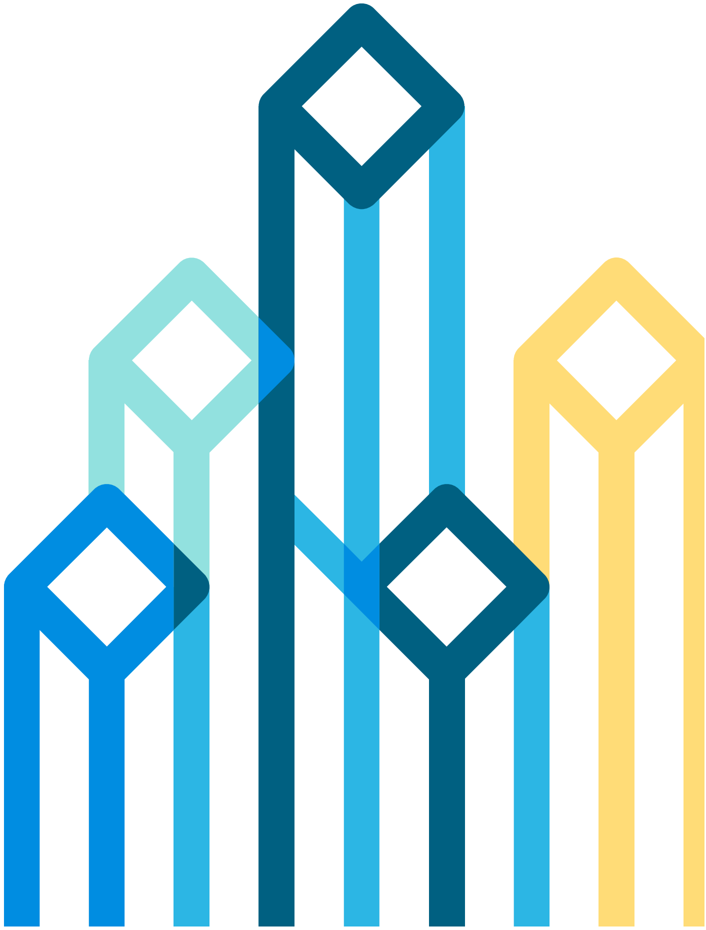
Budapest Data Forum, June 4th, 2015

tiny.cloudera.com/mark-sql-budapest

~~Mark Grover~~ | @mark_grover

~~Grover Mark~~

József



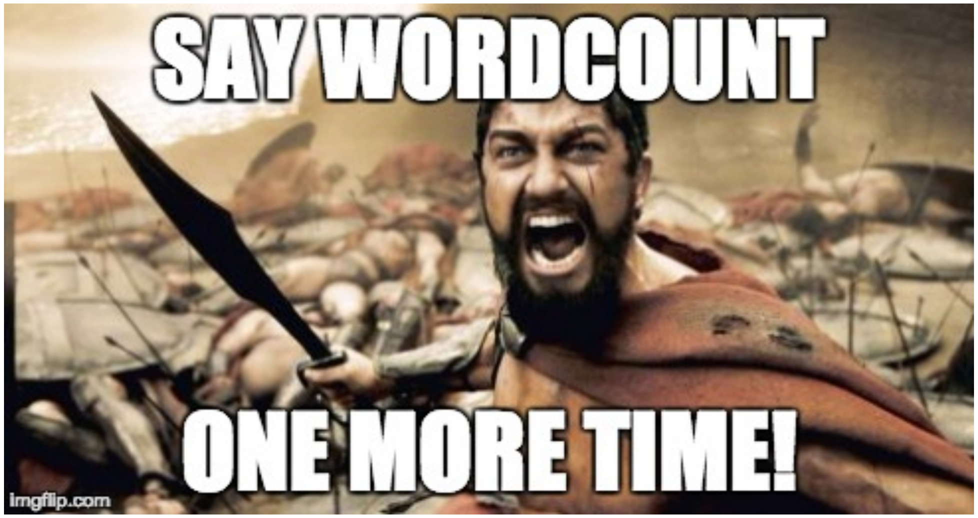
Agenda

cloudera

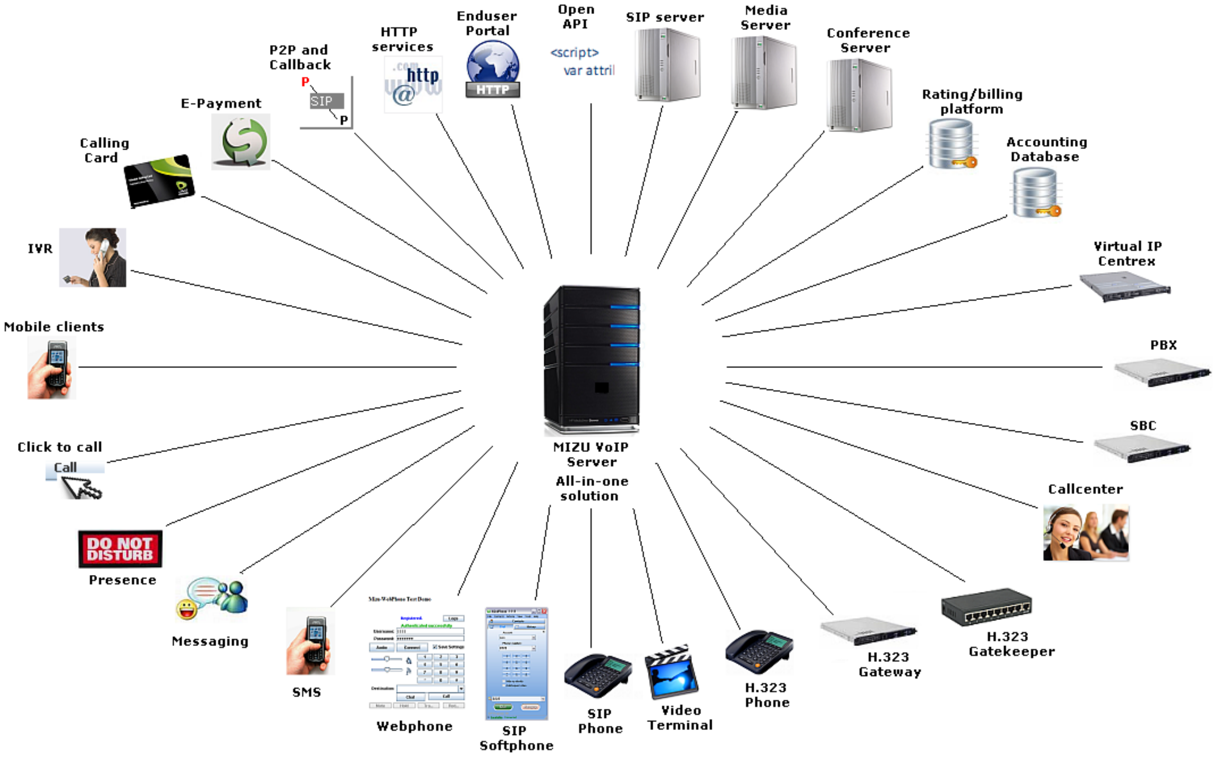
Agenda

- Word Count

Agenda



Past


Enduser Portal
Open API
<script>
var attril
SIP server
Media Server
HTTP services
http
P2P and Callback
SIP
Conference Server
Rating/billing platform
E-Payment
Accounting Database
Calling Card
IVR
Virtual IP Centrex
Mobile clients
PBX
SBC
MIZU VoIP Server
All-in-one solution
Click to call
Call
Callcenter
DO NOT DISTURB
Presence
H.323 Gatekeeper
Messaging
H.323 Gateway
SMS
H.323 Phone
Webphone
SIP Phone
Video Terminal
SIP Softphone

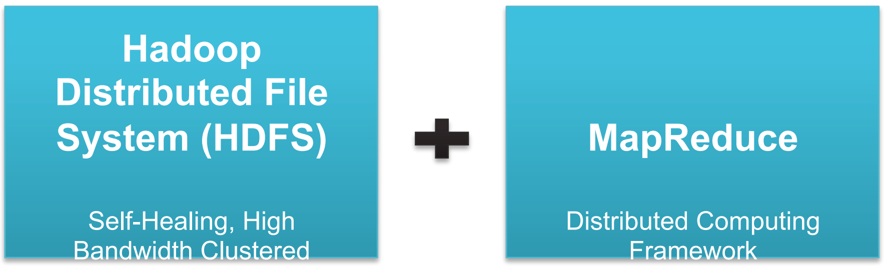
Future

What is Apache Hadoop?

Apache Hadoop is an open source platform for data storage and processing that is…

- ✓ Scalable
- ✓ Fault tolerant
- ✓ Distributed

CORE HADOOP SYSTEM COMPONENTS



Hadoop Distributed File System (HDFS)

Self-Healing, High Bandwidth Clustered

➕

MapReduce

Distributed Computing Framework

Has the Flexibility to Store and Mine Any Type of Data

- Ask questions across structured and unstructured data that were previously impossible to ask or solve
- Not bound by a single schema

Excels at Processing Complex Data

- Scale-out architecture divides workloads across multiple nodes
- Flexible file system eliminates ETL bottlenecks

Scales Economically

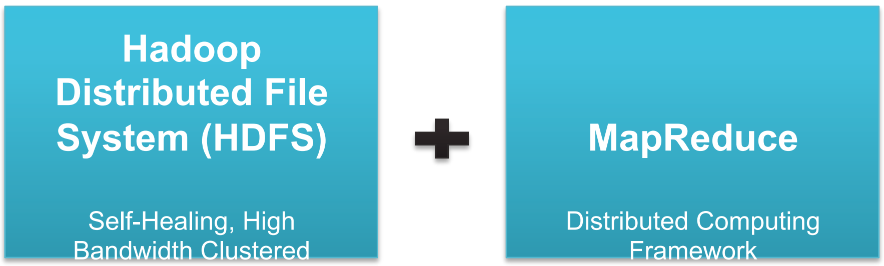
- Can be deployed on commodity hardware
- Open source platform guards against vendor lock

What is Apache Hadoop?

Apache Hadoop is an open source platform for data storage and processing that is…

- ✓ Scalable
- ✓ Fault tolerant
- ✓ Distributed

CORE HADOOP SYSTEM COMPONENTS



Hadoop Distributed File System (HDFS)

Self-Healing, High Bandwidth Clustered

MapReduce

Distributed Computing Framework

Has the Flexibility to Store and Mine Any Type of Data

- Ask questions across structured and unstructured data that were previously impossible to ask or solve
- Not bound by a single schema

Excels at Processing Complex Data

- Scale-out architecture divides workloads across multiple nodes
- Flexible file system eliminates ETL bottlenecks

Scales Economically

- Can be deployed on commodity hardware
- Open source platform guards against vendor lock

What? No schema?

- How do you use SQL?
- Do you lose Schema-on-read when using SQL-on-Hadoop?

SQL engines

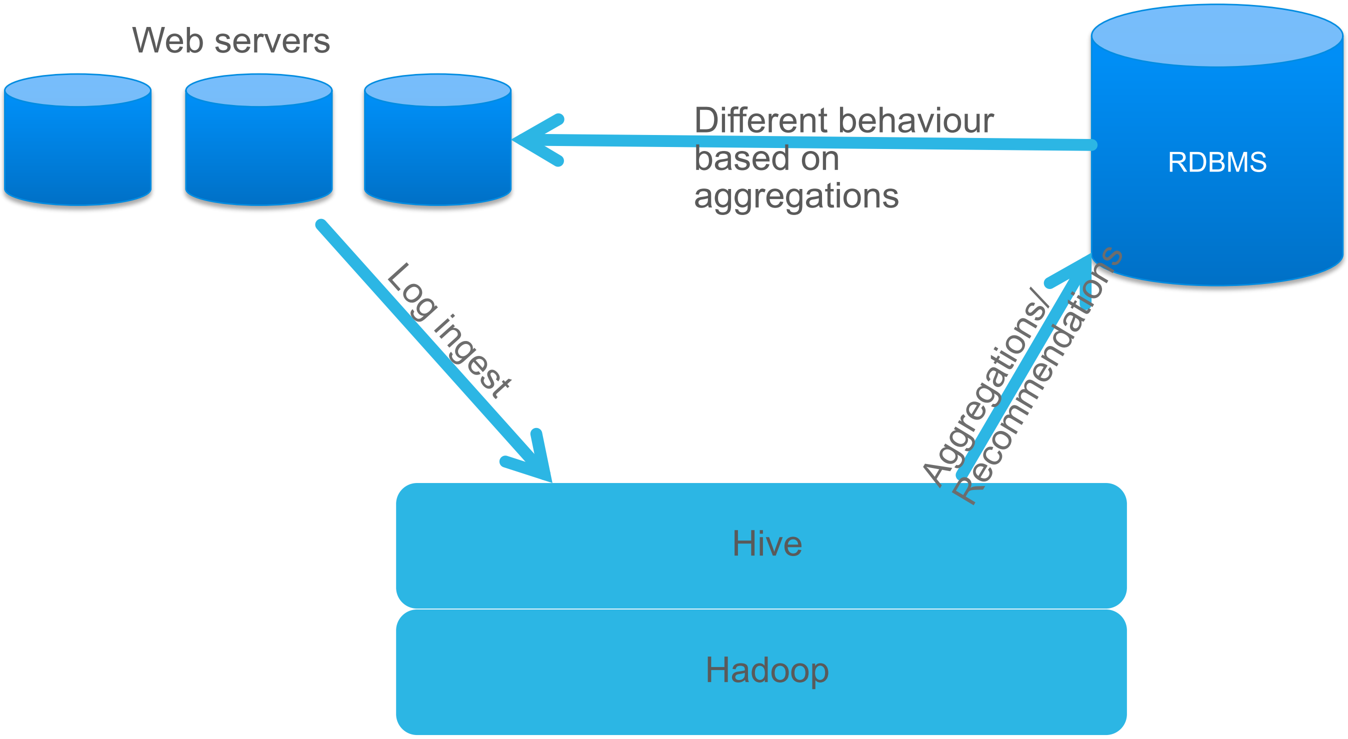
Hive

- First SQL engine
- Converts SQL to MapReduce
- New execution engine additions
  - Hive-on-Spark
  - Hive-on-Tez

Confusion #1

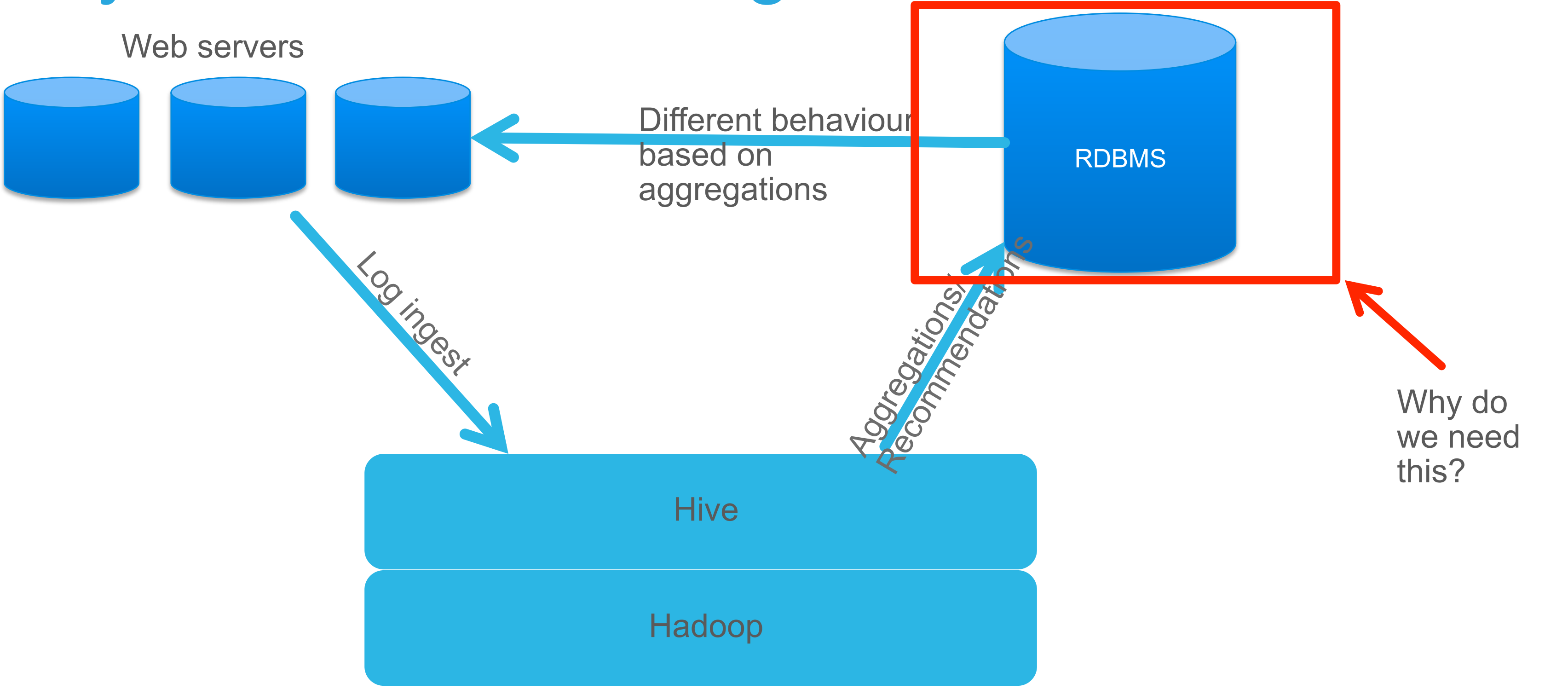
- If there's Hive, why we need more execution engines?
  - Just a matter of speed?

Early architecture using Hive


Web servers
Different behaviour based on aggregations
RDBMS
Log ingest
Aggregations/ Recommendations
Hive
Hadoop

cloudera

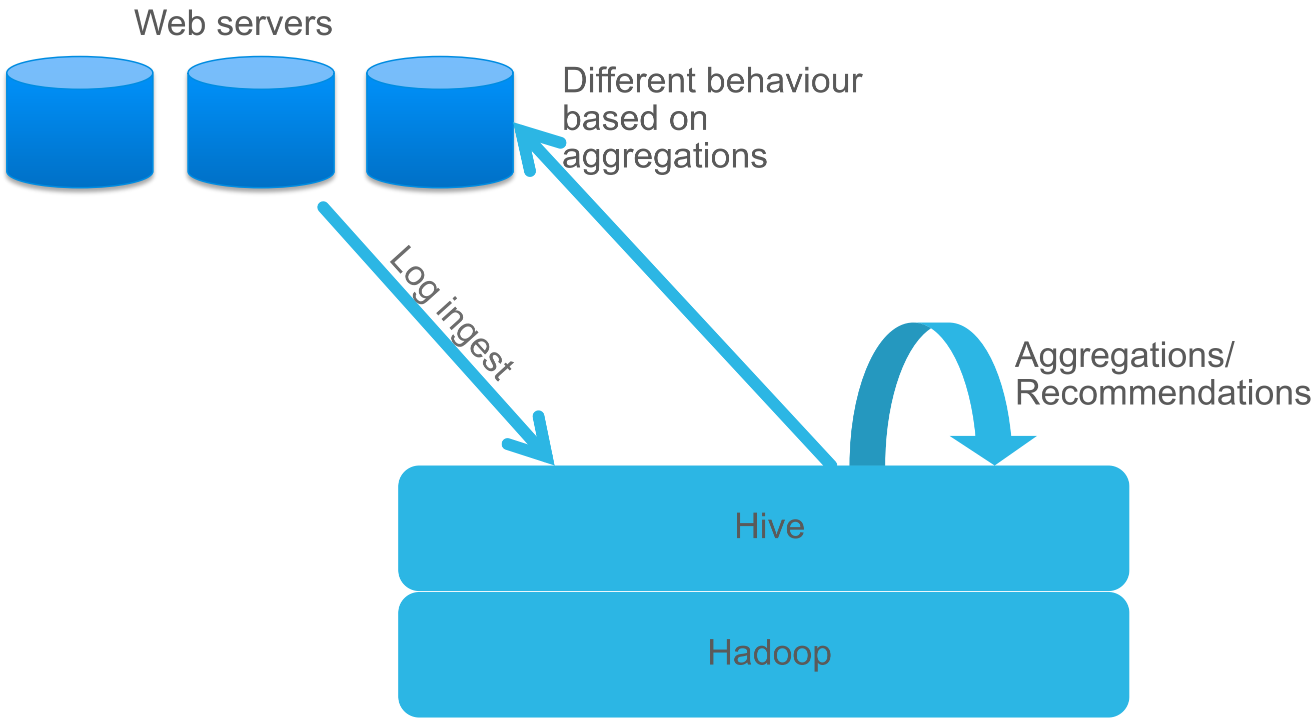
Early architecture using Hive


Web servers
Different behaviour based on aggregations
RDBMS
Log ingest
Aggregations/ Recommendations
Hive
Hadoop

Why do we need this?

cloudera

Why not?


Web servers
Different behaviour based on aggregations
Log ingest
Aggregations/ Recommendations
Hive
Hadoop

Aha #1

- Performance
  - Speed of execution
  - Concurrency
- Need an MPP like solution

Aha #1

- Impala
- Drill
- Presto
- Hive
  - Hive-on-Tez
  - Hive-on-Spark

Impala

Impala - Goals

- General-purpose SQL query engine:
    - Works for both for analytical and transactional/single-row workloads
- Runs directly within Hadoop:
    - reads widely used Hadoop file formats
    - talks to widely used Hadoop storage managers
    - runs on same nodes that run Hadoop processes
- High performance
    - Execution times
    - Concurrency
- Open source

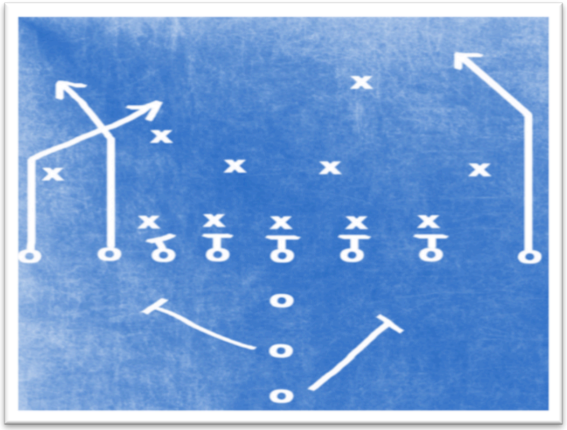
User view of Impala

- There is no ‘Impala format’!
- Supported file formats:
  - uncompressed/lzo-compressed text files
  - sequence files with snappy/gzip compression
  - RCFile with snappy/gzip compression
  - Avro data files
  - Parquet (columnar format)
  - HBase
  - And, more…

Impala Use Cases

Cost-effective, ad hoc query environment that offloads/ replaces the data warehouse for:

- Interactive BI/analytics on more data
- Asking new questions – exploration, ML
- Data processing with tight SLAs
- Query-able archive w/full fidelity



Global Financial Services Company

Saved 90% on incremental EDW spend & improved performance by 5x

Offload data warehouse for query-able archive

Store decades of data cost-effectively

Process & analyze on the same system

Improved capabilities through interactive query on more data

Digital Media Company

20x performance improvement for exploration & data discovery

Easily identify new data sets for modeling

Interact with raw data directly to test hypotheses

Avoid expensive DW schema changes

Accelerate ‘time to answer’

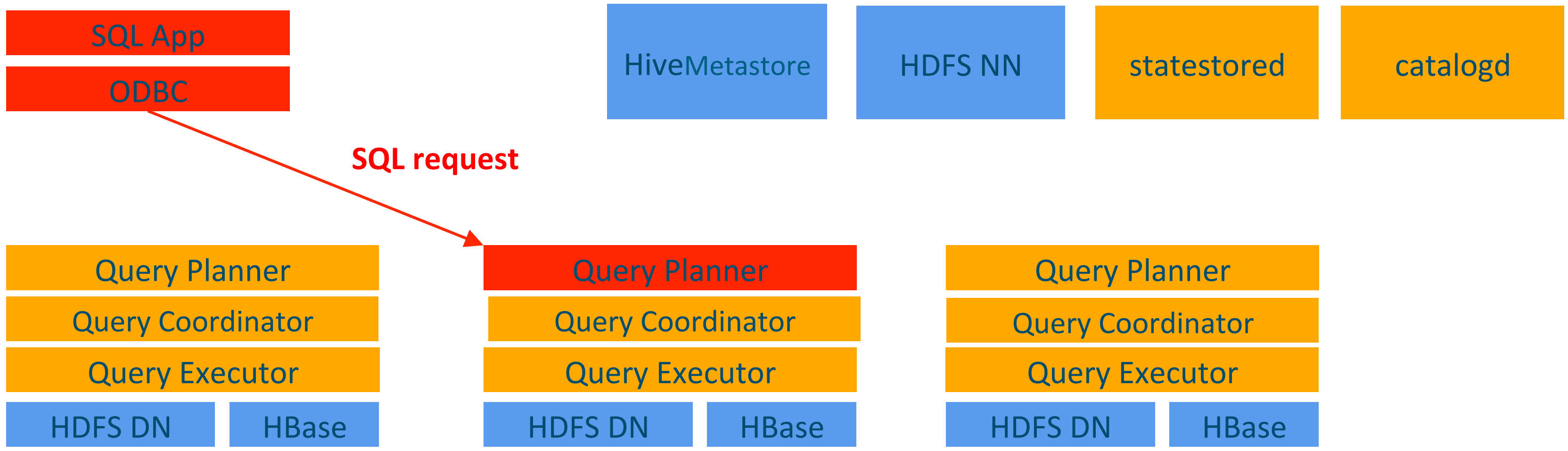
Architecture of Impala

Impala Architecture

- Three binaries: impalad, statestored, catalogd
- **Impala daemon (impalad)** – N instances
  - handles client requests and all internal requests related to query execution
- **State store daemon (statestored)** – 1 instance
  - Provides name service and metadata distribution
- **Catalog daemon (catalogd)** – 1 instance
  - Relays metadata changes to all impalad's

Impala Architecture: Query Execution

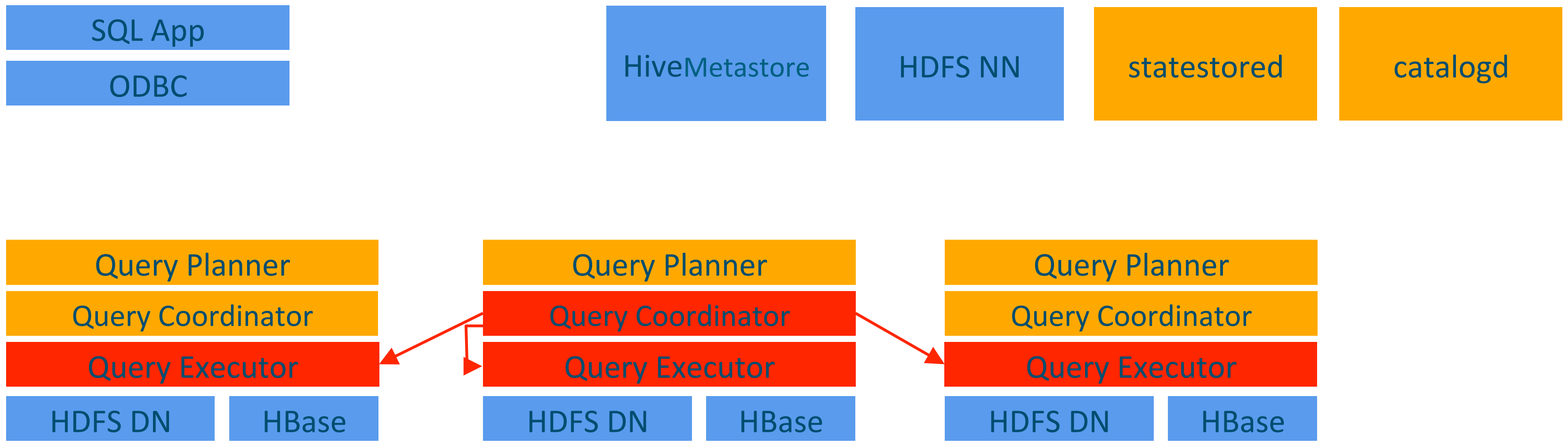
- Request arrives via odbc/jdbc



cloudera

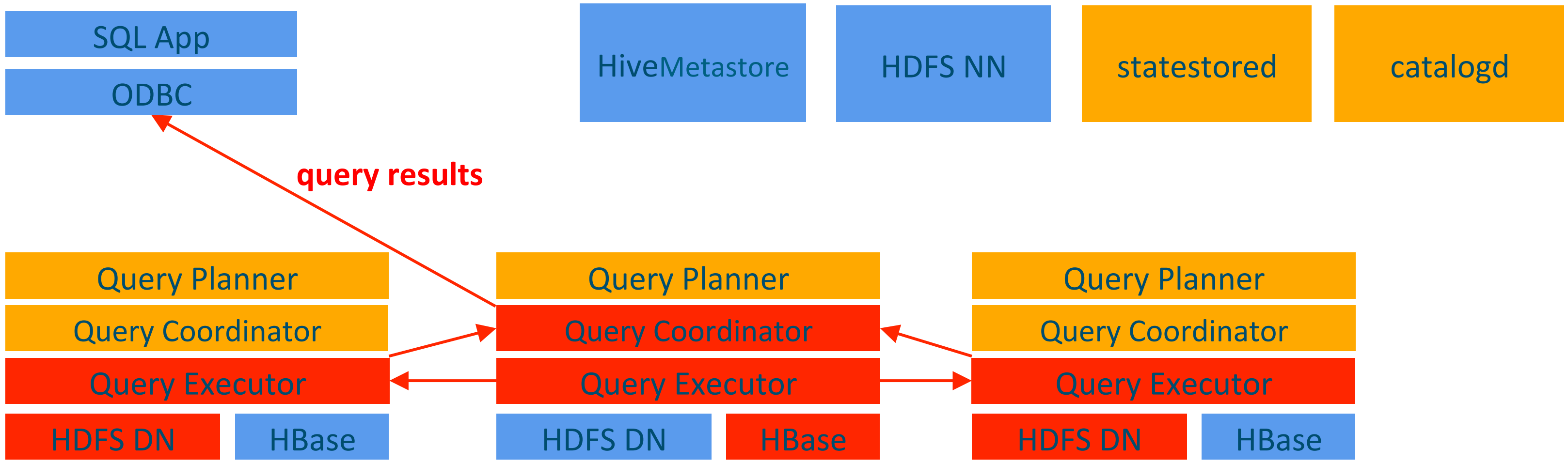
Impala Architecture: Query Execution

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



Impala Architecture: Query Execution

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



cloudera

Query Planning: Overview

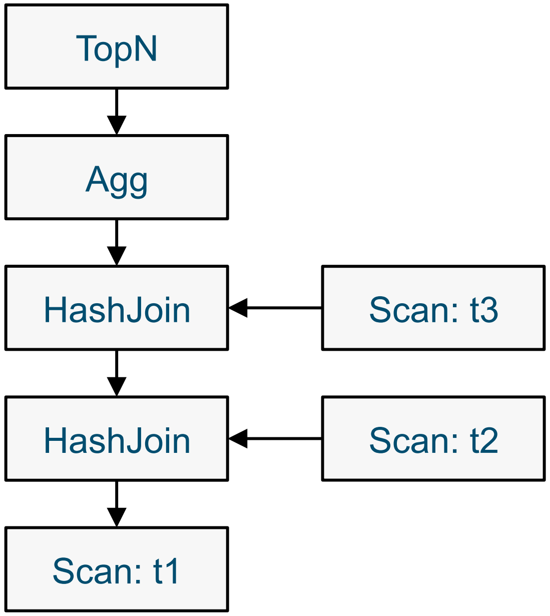
- 2-phase planning process:
  - single-node plan
  - plan partitioning: partition single-node plan to maximize scan locality, minimize data movement
- Parallelization of operators:
  - All query operators are fully distributed

Single-Node Plan: Example Query

```
SELECT t1.custid,
       SUM(t2.revenue) AS revenue
FROM LargeHdfsTable t1
JOIN LargeHdfsTable t2 ON (t1.id1 = t2.id)
JOIN SmallHbaseTable t3 ON (t1.id2 = t3.id)
WHERE t3.category = 'Online'
GROUP BY t1.custid
ORDER BY revenue DESC LIMIT 10;
```

Query Planning: Single-Node Plan

- Single-node plan for example:



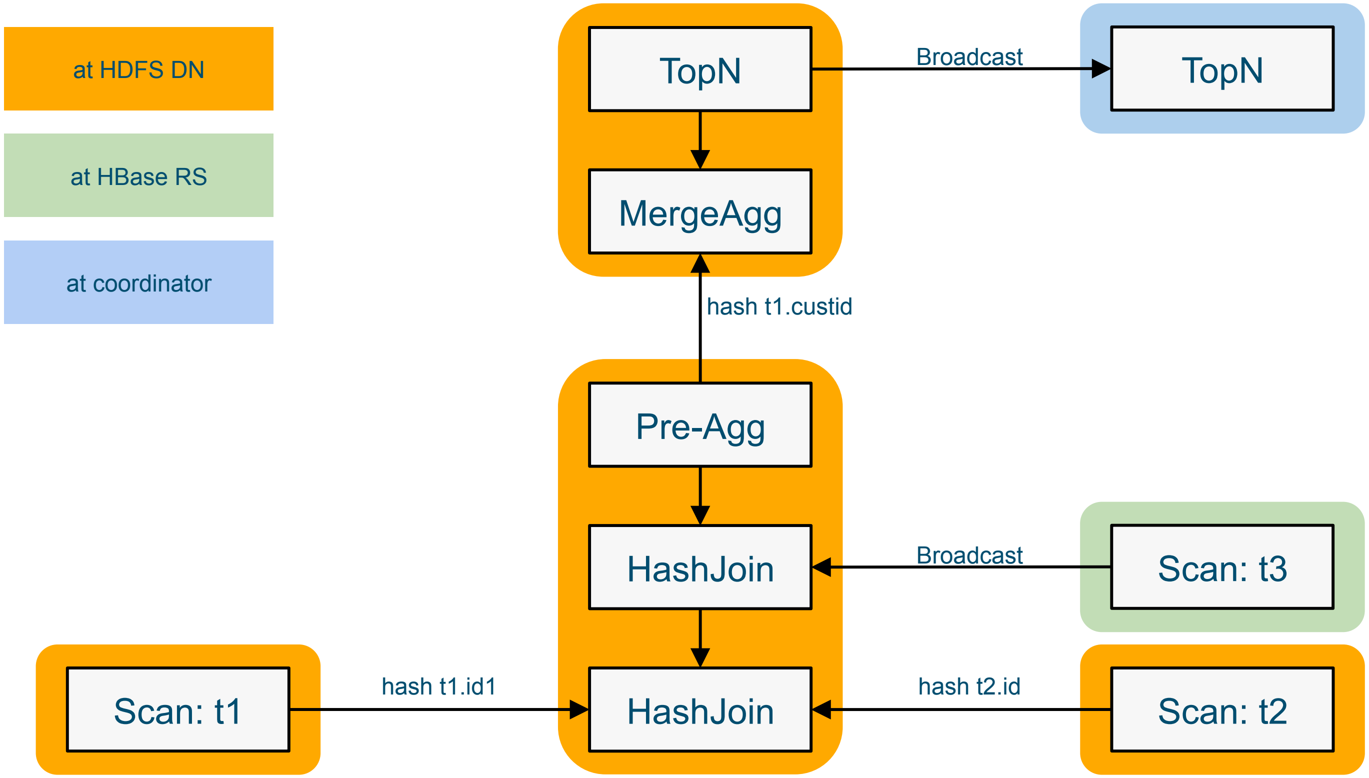
Single-node plan

- SQL query as a left-deep tree of plan operators
- Scan, HashJoin, HashAggregation, Union, TopN, Exchange

Plan Partitioning

- Partition single-node plan
  - Maximize scan locality
  - Minimize data movement
- Parallelization of operators:
  - All query operators are fully distributed

Query Planning: Distributed Plans



at HDFS DN

at HBase RS

at coordinator

TopN

Broadcast

TopN

MergeAgg

hash t1.custid

Pre-Agg

HashJoin

Broadcast

Scan: t3

Scan: t1

hash t1.id1

HashJoin

hash t2.id

Scan: t2

Impala Execution Engine

- Written in C++ for minimal execution overhead
- Internal in-memory tuple format puts fixed-width data at fixed offsets
- Uses intrinsics/special cpu instructions for text parsing, crc32 computation, etc.
- Runtime code generation for “big loops”

Runtime code generation

- example of "big loop": insert batch of rows into hash table
- known at query compile time: # of tuples in a batch, tuple layout, column types, etc.
- generate at compile time: unrolled loop that inlines all function calls, contains no dead code, minimizes branches
- code generated using LLVM

Comparing Impala to Dremel

- What is Dremel?
  - columnar storage for data with nested structures
  - distributed scalable aggregation on top of that
- Columnar storage in Hadoop: Parquet
  - stores data in appropriate native/binary types
  - can also store nested structures similar to Dremel's ColumnIO
  - Parquet is open source: github.com/parquet
- Distributed aggregation: Impala
- Impala plus Parquet: a superset of the published version of Dremel (which didn't support joins)

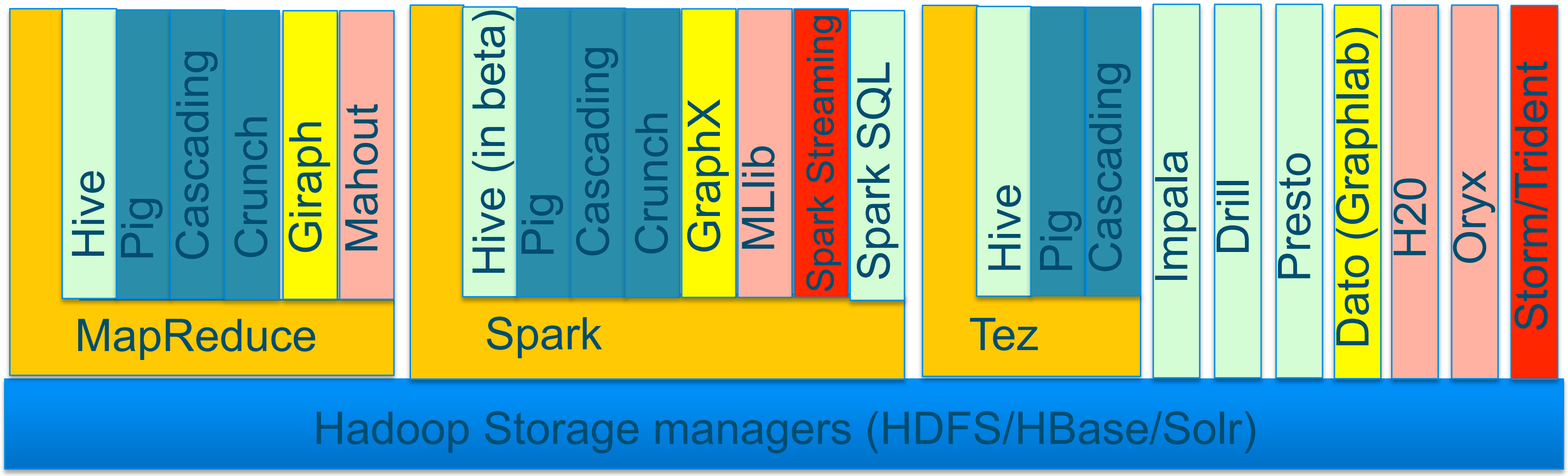
But, what makes Impala fast?

- No MapReduce
- Use of memory
- LLVM
- C++
- Vectorization
- Tight integration with Parquet

Confusion #2 – What do I use?

- Hive on Mapreduce?
- Hive on Tez?
- Hive on Spark?
- Impala?
- Spark SQL?
- Drill?

Processing Frameworks in Hadoop



MapReduce: Hive, Pig, Cascading, Crunch, Giraph, Mahout

Spark: Hive (in beta), Pig, Cascading, Crunch, GraphX, MLlib, Spark Streaming, Spark SQL

Tez: Hive, Pig, Cascading

Impala

Drill

Presto

Dato (Graphlab)

H20

Oryx

Storm/Trident

Hadoop Storage managers (HDFS/HBase/Solr)

🟦 Storage managers

🟧 General purpose execution engines

🟩 SQL engines

🟦 Abstraction engines

🟥 Real-time frameworks

🟨 Graph processing engines

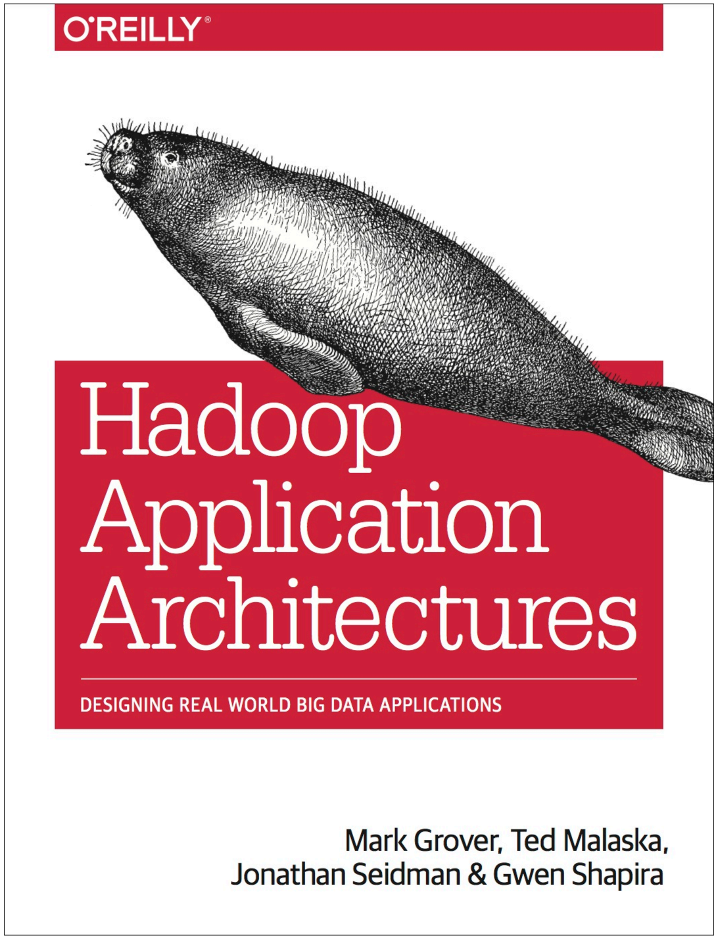
🟧 Machine Learning engines

cloudera

Free books!

- @hadooparchbook
- hadooparchitecturebook.com
- github.com/hadooparchitecturebook
- slideshare.com/hadooparchbook

- Later today at 3:10 PM



O'REILLY®

Hadoop Application Architectures

DESIGNING REAL WORLD BIG DATA APPLICATIONS

Mark Grover, Ted Malaska,
Jonathan Seidman & Gwen Shapira

Impala

- SQL-on-Hadoop engine
- Near real time SQL
- Reads YOUR file formats
- Allows to write custom UDFs
- Open source
- Commonly used for Data Warehouse offloading

I’d love to talk more!

- @mark_grover
- Find me at Cloudera booth!