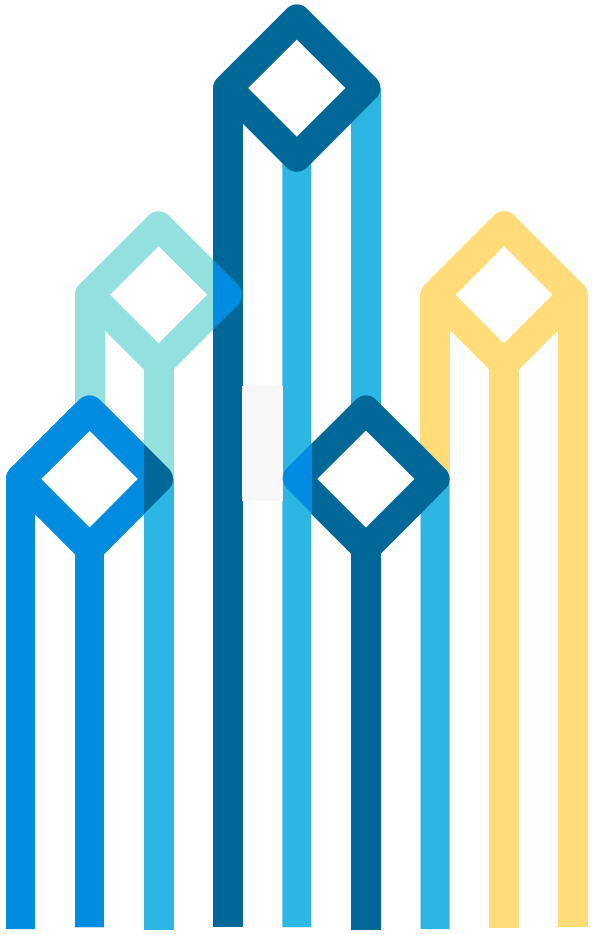
cloudera

Spark
Lightning-Fast Cluster Computing



Cloudera Improvements in Apache Spark

Brian Baillod | Sales Engineer

Agenda

- Introduction
- Spark One Platform Initiative
- Spark Overview and Improvements
- Spark Proof of Concept
- Kudu and Record Service

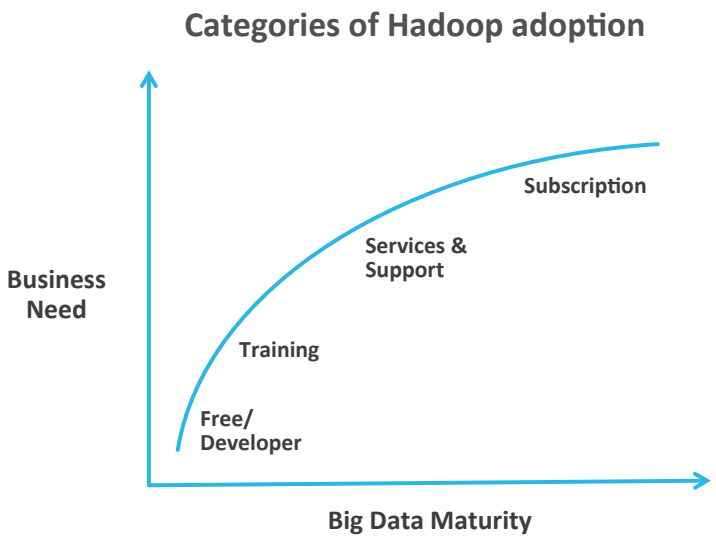
cloudera

Cloudera company snapshot

Founded	2008, by former employees of ORACLE YAHOO! facebook Google
Employees Today	900+ worldwide
World Class Support	More than 75 24x7 global staff
Cloudera University	Over 40,000 trained
We help code Hadoop	Cloudera employees are leading developers & contributors to the complete Apache Hadoop ecosystem of projects
We help fix Hadoop	Cloudera fixed 60% of all Hadoop JIRA bugs

cloudera

Hadoop Adoption



Free/Developer → Over **2.5 million** downloads

Training → **60% of Fortune 100** attended Cloudera training, **over 40,000** trained since 2009

Service & Support → **9/10** for support satisfaction, ability to solve technical issues **#1 recommendation**

Subscription → Over **2x** revenue of nearest competitor, **90%** renewal rate

cloudera

What is Spark

- Fast general purpose processing engine for large data
- Provides API's in Java, Scala and Python
- Includes an advanced DAG execution engine that supports in-memory computing
- Includes high level tools like SparkSQL, Mllib, GraphX, and Spark Streaming
- Can run in a cluster, standalone, or local
- Latest version is 1.5.1
- Spark.apache.org
- LOTS of momentum

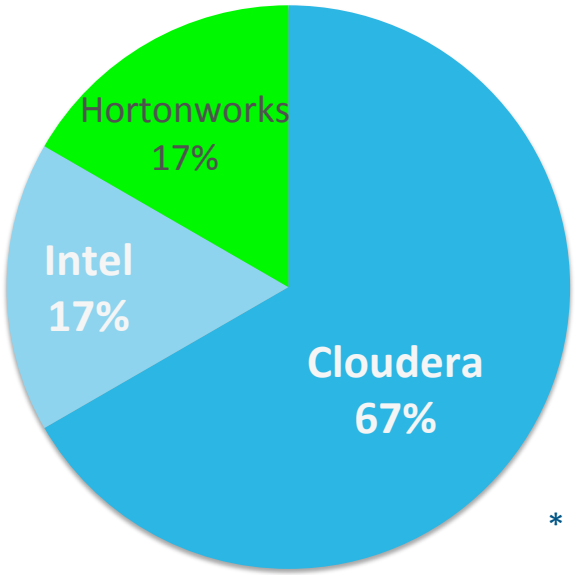
Cloudera One Platform Initiative

- Cloudera is doubling down on Spark
- Outlining a vision for the future
  - Kudu, Record Service, Auto-tuning, Security, Kafka integration
- Challenging other vendors to participate in Spark Development

cloudera

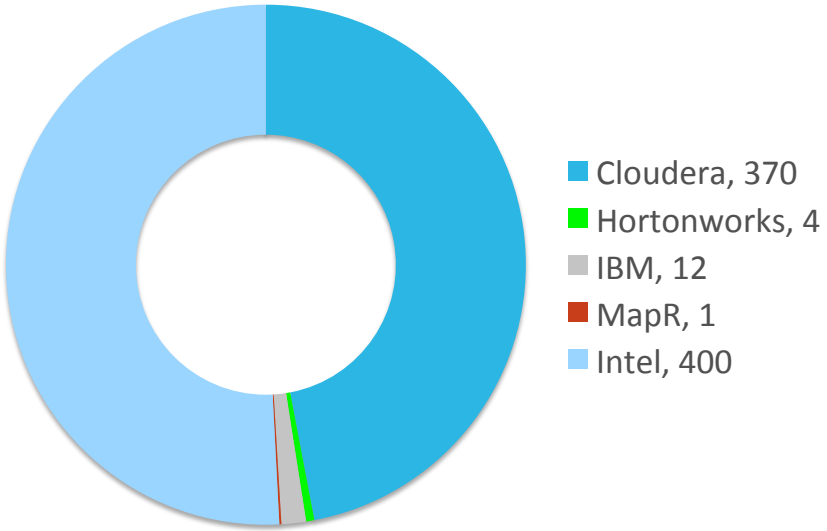
Cloudera's Engineering Commitment to Spark

Spark Committers by Hadoop Distribution*



* IBM and MapR have **0 committers**

Spark Patches by Hadoop Distribution



cloudera

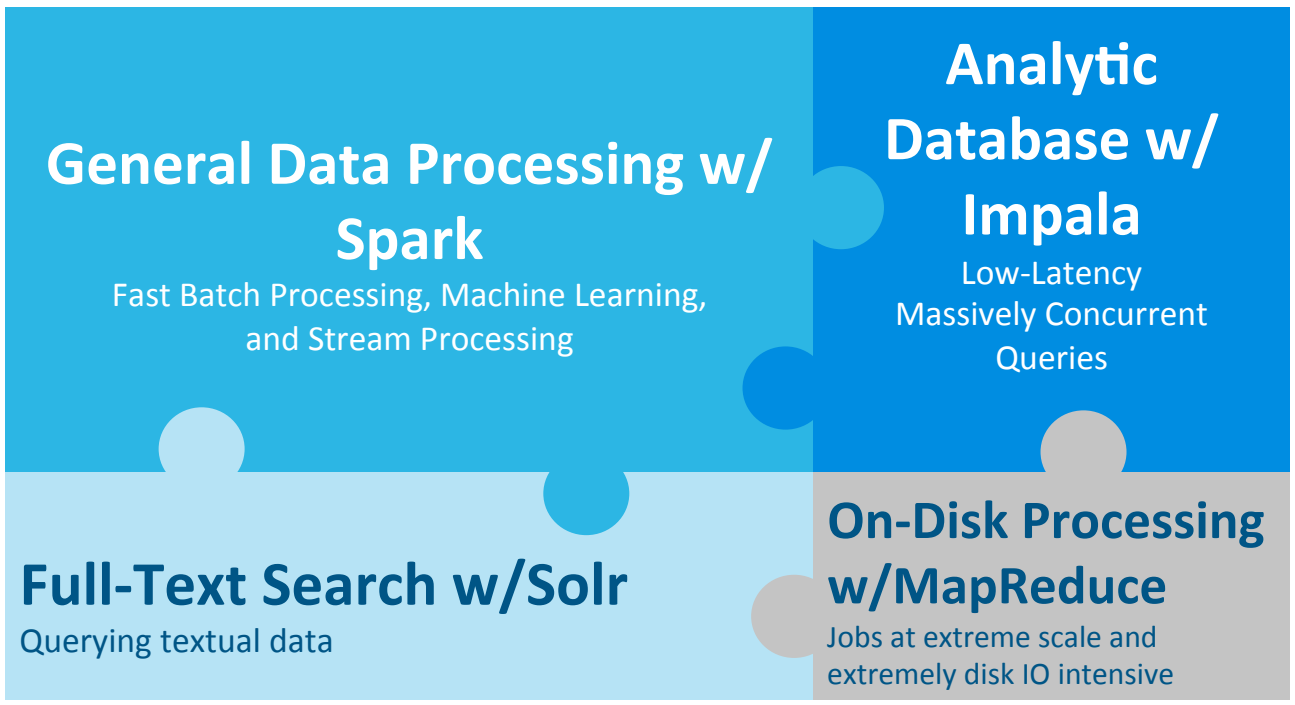
Spark will replace MapReduce

To become the standard execution engine for Hadoop

cloudera

The Future of Data Processing on Hadoop

Spark complemented by specialized fit-for-purpose engines



Shared:

- Data Storage
- Metadata
- Resource Management
- Administration
- Security
- Governance

cloudera

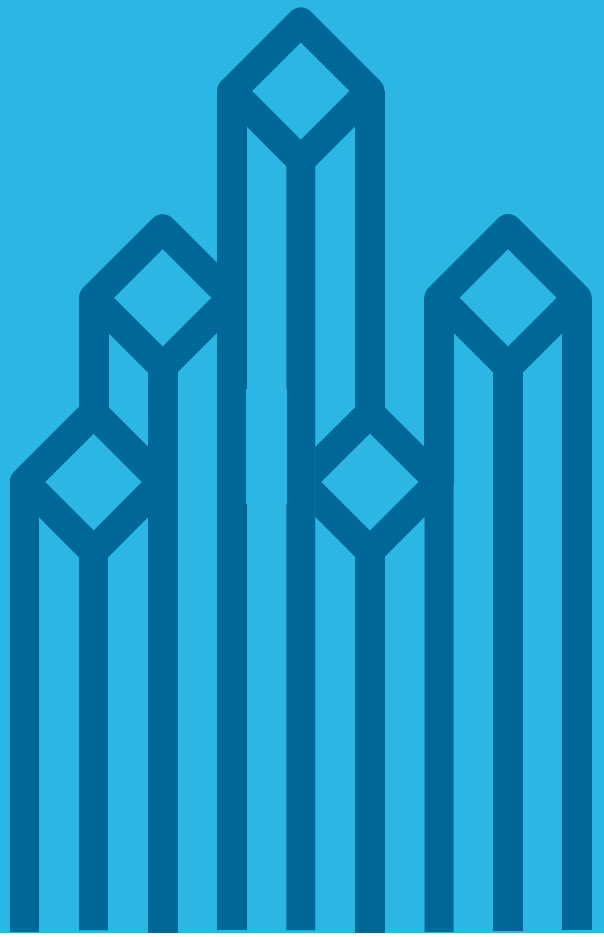
Why is Cloudera leading this initiative?

- Cloudera was the **first** Hadoop vendor to ship and support Spark
- Spark is a **fully integrated** part of Cloudera's platform
  - Shared data, metadata, resource management, administration, security, and governance
- Cloudera is the **first** Hadoop vendor to offer Spark training
  - Trained more customers than any other vendor
- Cloudera has **more Spark customers** in production than all other companies combined

cloudera

cloudera

Spark Overview and Improvements



Apache Spark

Flexible, in-memory data processing for Hadoop

Easy Development	Flexible Extensible API	Fast Batch & Stream Processing

• Rich APIs for Scala, Java, and Python
• Interactive shell
	• APIs for different types of workloads:
	  • Batch
	  • Streaming
	  • Machine Learning
	  • Graph
		• In-Memory processing and caching

cloudera

Easy Development

High Productivity Language Support

Python

```
lines = sc.textFile(...)
lines.filter(lambda s: "ERROR" in s).count()
```

Scala

```
val lines = sc.textFile(...)
lines.filter(s => s.contains("ERROR")).count()
```

Java

```
JavaRDD<String> lines = sc.textFile(...);
lines.filter(new Function<String, Boolean>() {
  Boolean call(String s) {
    return s.contains("error");
  }
}).count();
```

- Native support for multiple languages with identical APIs
  - Scala, Java, Python
- Use of closures, iterations, and other common language constructs to minimize code
  - 2-5x less code

cloudera

Python Or Scala?

- Use Python for prototyping
- Spark Python API is slower than Scala

- Use Scala for development
  - Steep learning curve for functional programming

Easy Development
Use Interactively

```
percolateur:spark srowen$ ./bin/spark-shell --master local[*]
...
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.5.0-SNAPSHOT
      /_/

Using Scala version 2.10.4
 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_51)
Type in expressions to have them evaluated.
Type :help for more information.
...

scala> val words = sc.textFile("file:/usr/share/dict/words")
...
words: org.apache.spark.rdd.RDD[String] =
 MapPartitionsRDD[1] at textFile at <console>:21

scala> words.count
...
res0: Long = 235886

scala>
```

- Interactive exploration of data for data scientists
  - No need to develop "applications"
- Developers can prototype application on live system

cloudera

Easy Development

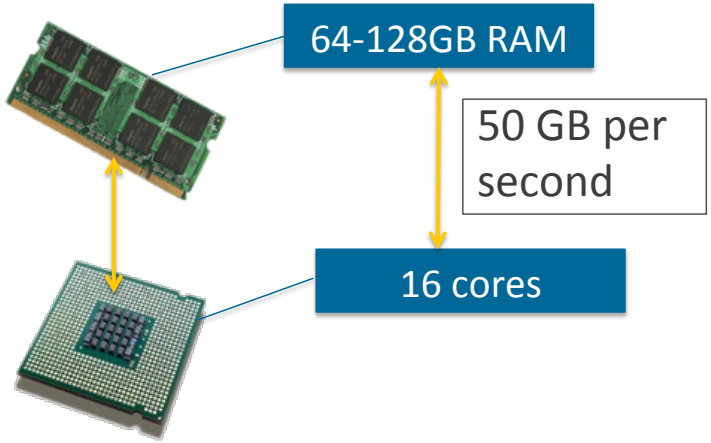
Expressive API

- map
- filter
- groupBy
- sort
- union
- join
- leftOuterJoin
- rightOuterJoin
- reduce
- count
- fold
- reduceByKey
- groupByKey
- cogroup
- cross
- zip
- sample
- take
- first
- partitionBy
- mapWith
- pipe
- save
- …

cloudera

Memory Management for Greater Performance

Memory can be enabler for high performance big data applications



Trends:

- ½ price every 18 months
- 2x bandwidth every 3 years

cloudera

Spark Concepts

- RDD – Resilient Distributed Dataset
- Transformations
- Actions
- Caching
- DataFrames
- Spark Streaming
- SparkSQL
- Pluggable Spark

cloudera

Resilient Distributed Dataset (RDD)

- Read-only partitioned collection of records
- Created through:
  - Transformation of data in storage
  - Transformation of RDDs
- Contains lineage to compute from storage
- Lazy materialization
- Users control persistence and partitioning

RDD Operations

- **Transformations** create new RDD from an existing one
- **Actions** run computation on RDD and return a value

- Transformations are lazy
- Actions materialize RDDs by computing transformations
- RDDs can be cached to avoid re-computing

Example Operations

Transformations

- `Map`
- `Filter`
- `Sample`
- `Join`

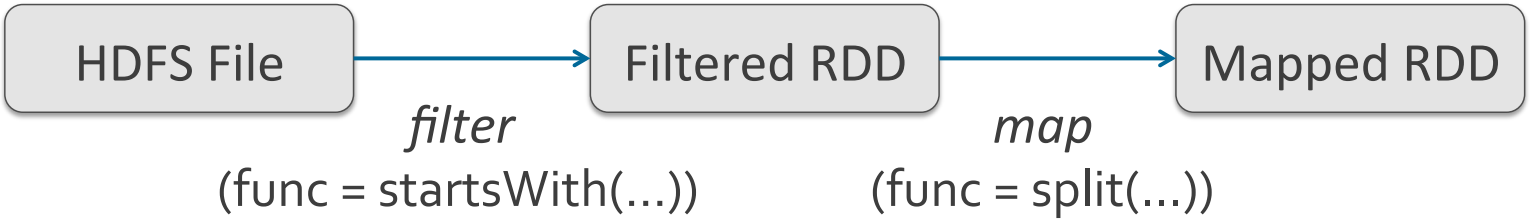
Actions

- `Reduce`
- `Count`
- `First, Take`
- `SaveAs`

Fault-Tolerance

- RDDs contain lineage
  - Lineage: Source location and list of transformations
  - Lost partitions can be re-computed from source data

```
msgs = textFile.filter(lambda s: s.startsWith("ERROR"))
               .map(lambda s: s.split("\t")[2])
```



cloudera

Caching – Storage Levels

Different options provide tradeoffs between memory usage and CPU efficiency. Cache when using iterative algorithms.

- MEMORY_ONLY – most CPU efficient, data has to fit in memory
- MEMORY_ONLY_SER – More space efficient but still reasonably fast
- MEMORY_AND_DISK
- MEMORY_AND_DISK_SER
- DISK_ONLY
- MEMORY_ONLY_2, MEMORY_AND_DISK_2…

cloudera

Data Frames

- Distributed collection of rows organized into named columns
- Spark SQL's Data Source API can read and write Data Frames using a variety of formats
  - Hive, JSON, Parquet, HDFS
- Calling the DataFrame API can let you
  - Select the columns you want
  - Join data sources
  - Aggregate and Filter

- Spark 1.5 lets you access the Hive Metastore to read/write schemas directly.

cloudera

Spark Streaming

What is it?

- Run *continuous* processing of data using Spark's core API
- Extends Spark concepts to fault-tolerant, transformable streams
- Adds "rolling window" operations
  - Example: Compute rolling averages or counts for data over last five minutes

Common Use Cases:

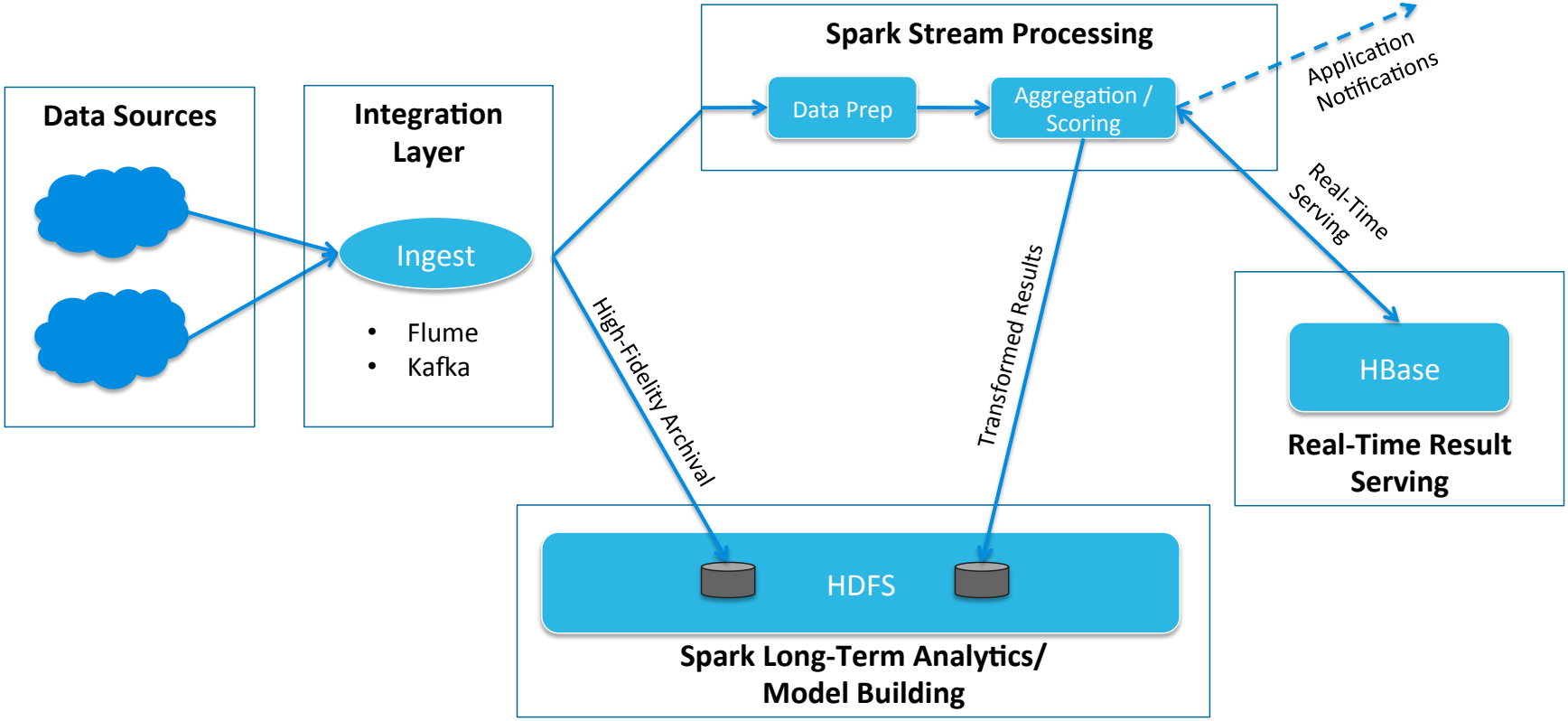
- "On-the-fly" ETL as data is ingested into Hadoop/HDFS
- Detect anomalous behavior and trigger alerts
- Continuous reporting of summary metrics for incoming data

Benefits:

- Same programming paradigm for streaming and batch
- Excellent throughput
  - Scale easily to support large volumes of data ingest

cloudera

Spark Streaming Architectures



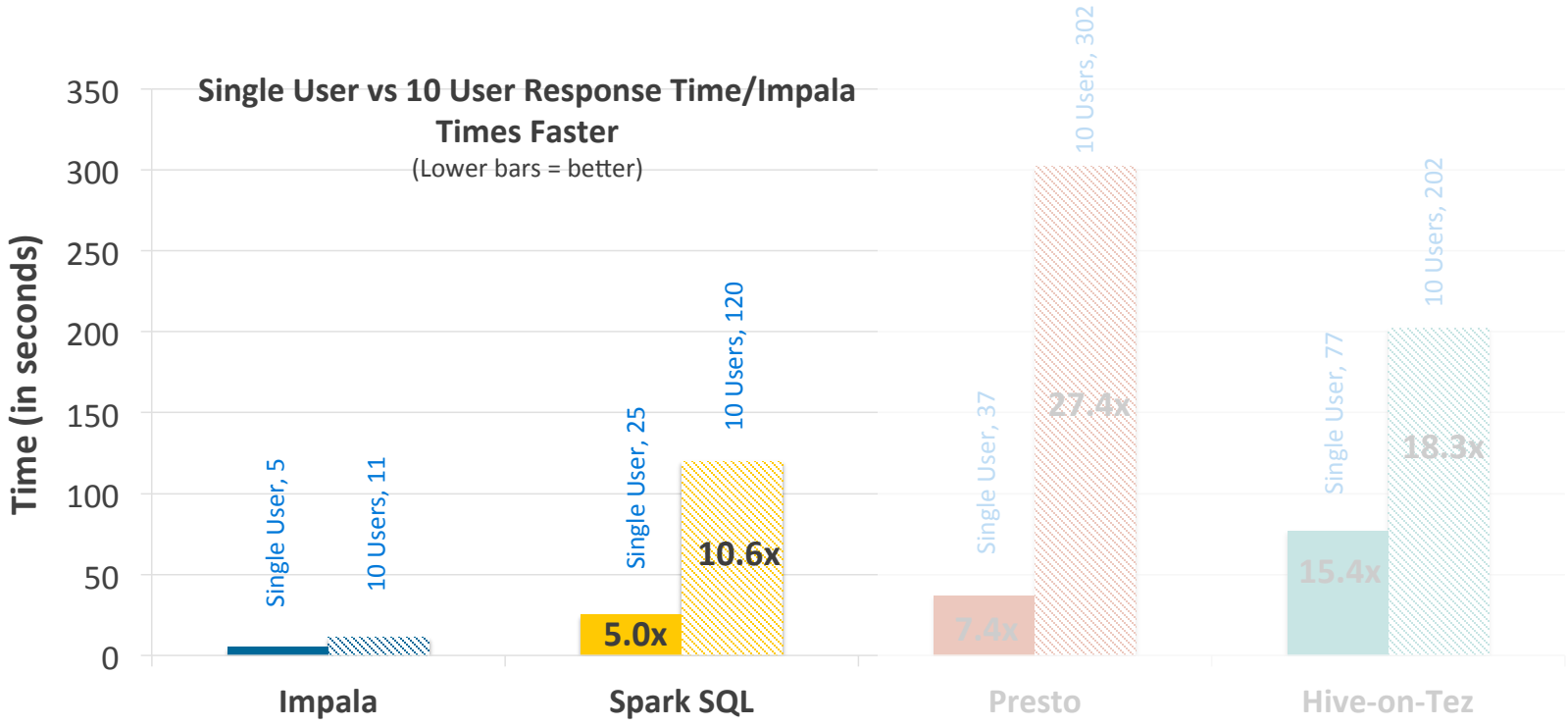
cloudera

SparkSQL
Machine Learning Applications

- Goal:
  - Spark/Java Developers and Data Scientists can inline SQL into Spark apps
- Designed for:
  - Ease of development for Spark developers
  - Handful of concurrent Spark jobs
- Strengths:
  - Ease of embedding SQL into Java or Scala applications
  - SQL for common functionality in developer flow (eg. aggregations, filters, samples)

cloudera

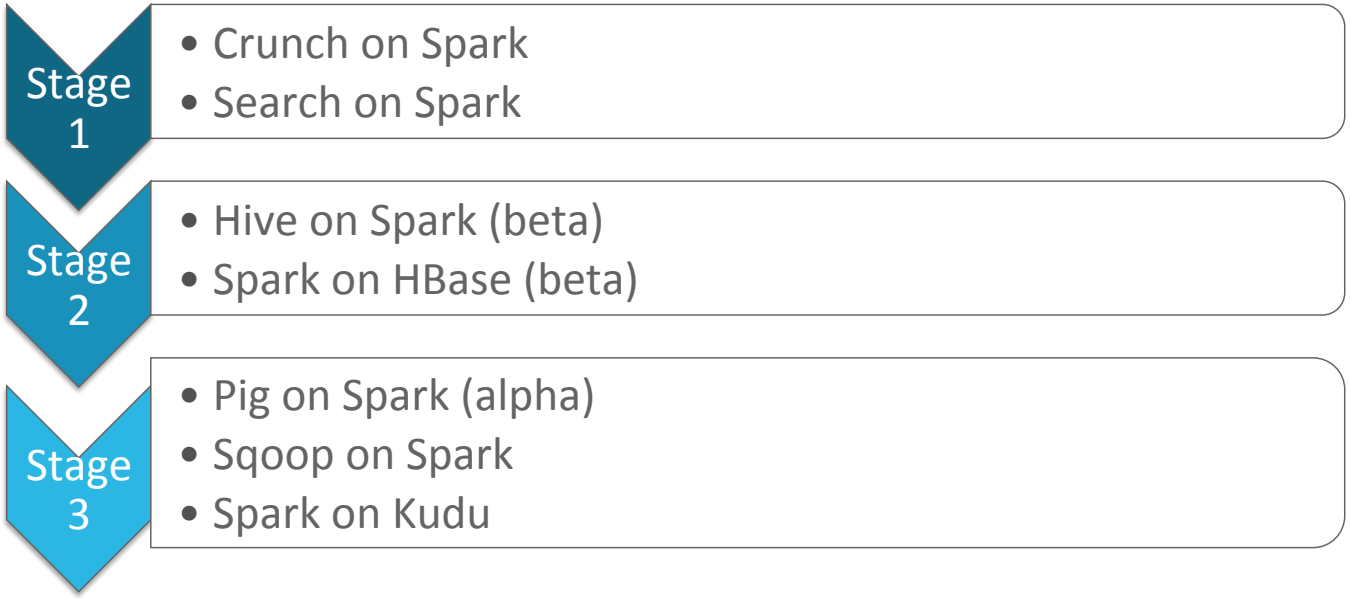
Impala Remains Tool of Choice for Interactive SQL



cloudera

Pluggable Spark – replace MapReduce

Cloudera is leading community development to port components to Spark:



cloudera

Spark Customer Use Cases

Core Spark

Financial Services

- Portfolio Risk Analysis
- ETL Pipeline Speed-Up
- 20+ years of stock data

Health

- Identify disease-causing genes in the full human genome
- Calculate Jaccard scores on health care data sets

ERP

- Optical Character Recognition and Bill Classification

Data Services

- Trend analysis
- Document classification (LDA)
- Fraud analytics

Spark Streaming

Financial Services

- Online Fraud Detection

Health

- Incident Prediction for Sepsis

Retail

- Online Recommendation Systems
- Real-Time Inventory Management

Ad Tech

- Real-Time Ad Performance Analysis

cloudera

Doing the Math – Executors and Cores

Determine the optimal resource allocation for the Spark job

16 Total Cores in Cluster

C 1 Core for Application Master

1 15 Cores for Executors

Core Allocation

$$\frac{1 \text{ Executor}}{4 \text{ Cores}} \times 15 \text{ Cores} =$$

3 Executors with 4 Cores Each

(Leaves 3 Cores un-utilized)

Other Ratios may lead to better resource utilization

$$\frac{1 \text{ Executor}}{2 \text{ Cores}} \times 15 \text{ Cores} =$$

7 Executors with 2 Cores Each

(Leaves 1 Core un-utilized)

Don't exceed= 5 Cores per Executor

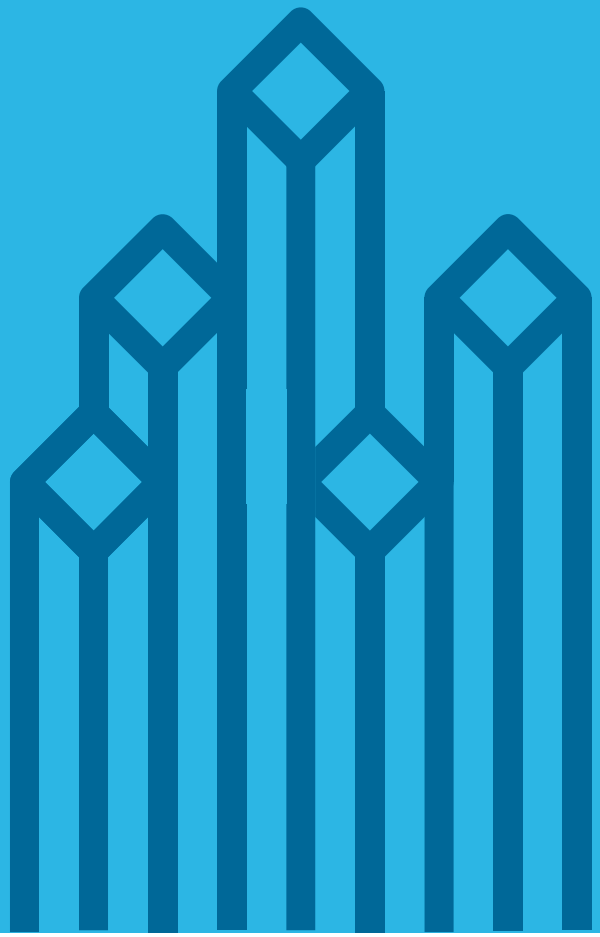
http://blog.cloudera.com/blog/2015/03/how-to-tune-your-apache-spark-jobs-part-2/

Allocate Executors

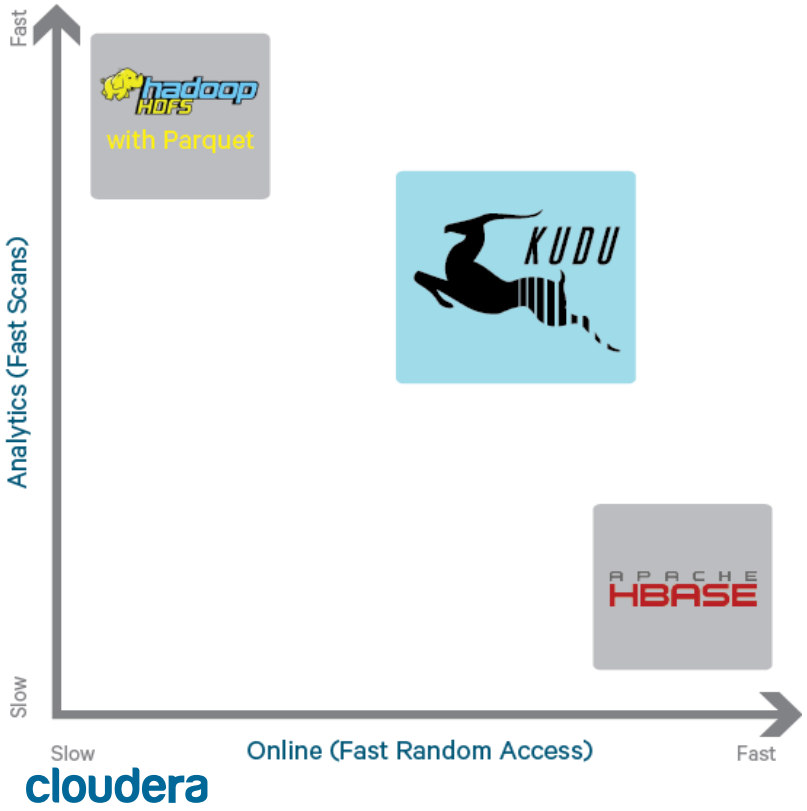
cloudera

cloudera

Kudu

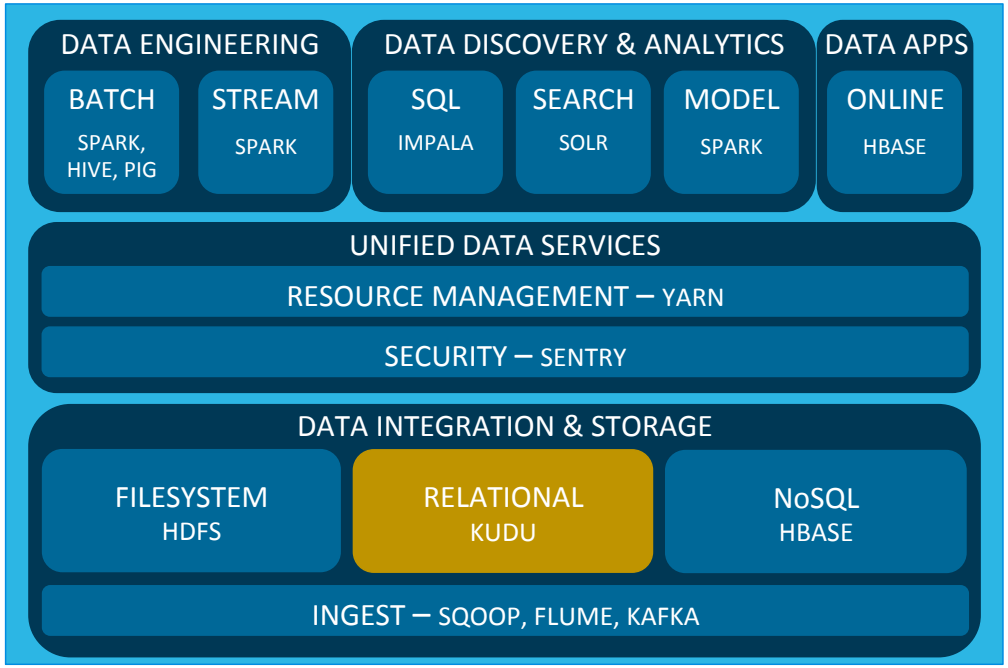


Kudu Design Goals



- **High throughput** for big scans (columnar storage and replication)
  Goal: Within 2x of Parquet
- **Low-latency** for short accesses (primary key indexes and quorum design)
  Goal: 1ms read/write on SSD
- **Database-like** semantics (initially single-row ACID)
- **Relational data model**
  - SQL query
  - "NoSQL" style scan/insert/update (Java client)

Kudu
Storage for Fast Analytics on Fast Data



- New updating column store for Hadoop
  - Simplifies the architecture for building analytic applications on changing data
  - Designed for fast analytic performance
  - Natively integrated with Hadoop
- Apache-licensed open source (intent to donate to ASF)
- Beta now available

cloudera

Kudu Trade-Offs

- Random updates will be slower
  - HBase model allows random updates without incurring a disk seek
  - Kudu requires a key lookup before update, Bloom lookup before insert

- Single-row reads may be slower
  - Columnar design is optimized for scans
  - Future: may introduce “column groups” for applications where single-row access is more important

cloudera

Resources

Join the community

http://getkudu.io

Download the Beta

cloudera.com/downloads

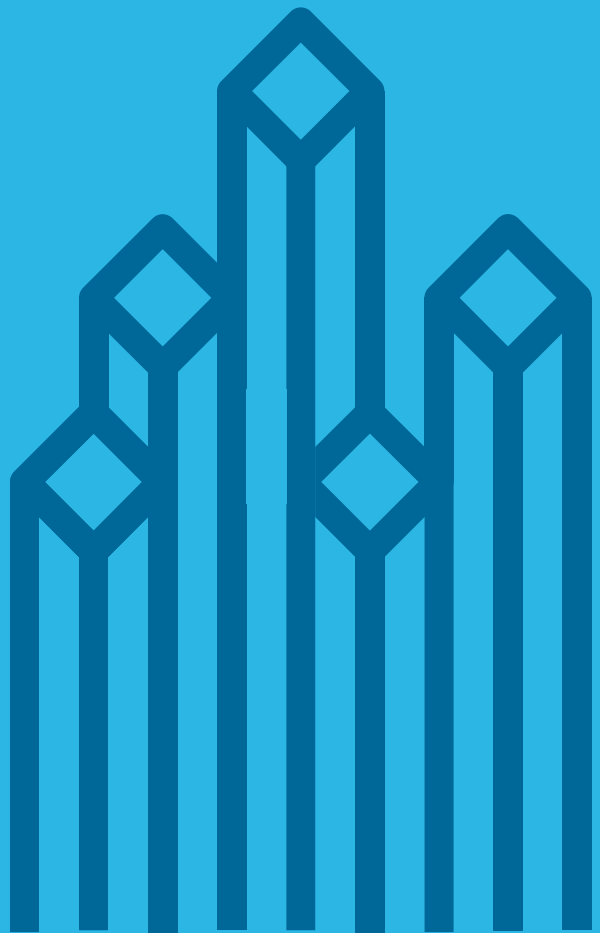
Read the Whitepaper

getkudu.io/kudu.pdf

cloudera

cloudera

RecordService



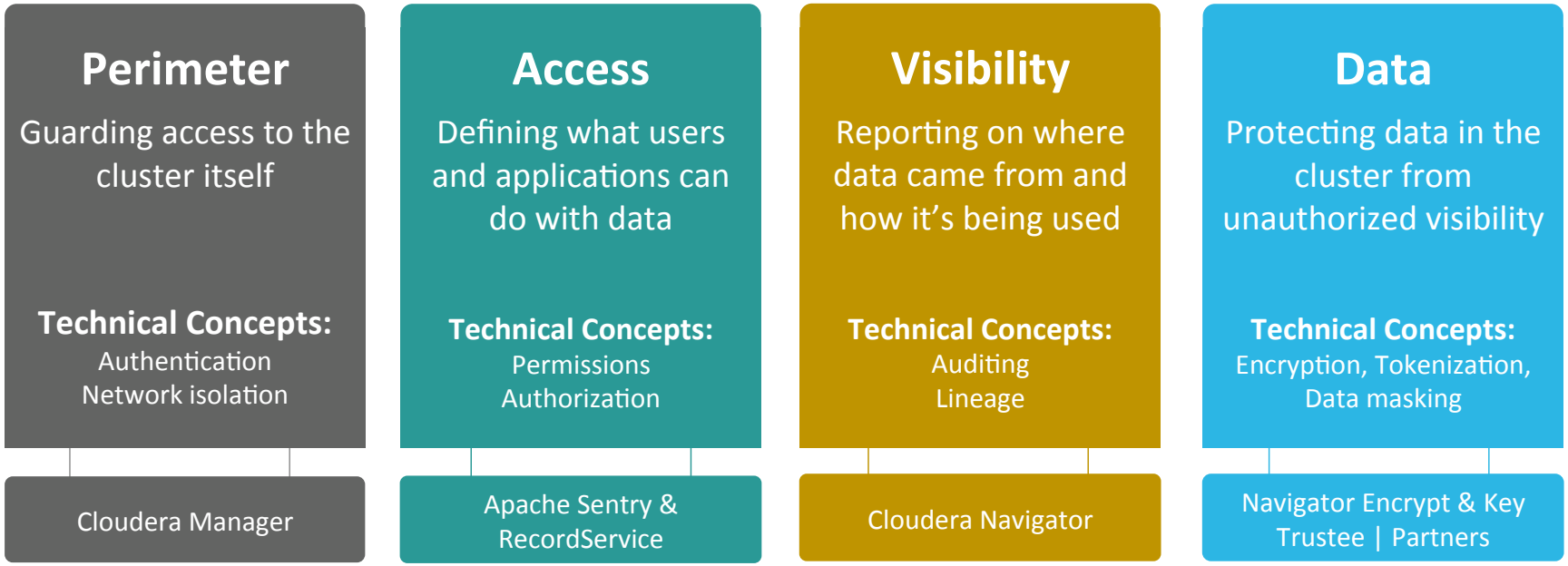
Hadoop started out with zero security

- Didn’t need it for the Silicon Valley applications
- Does need it for Corporate applications
- Cloudera is working on providing full featured Spark Security

cloudera

Comprehensive, Compliance-Ready Security

Authentication, Authorization, Audit, and Compliance



cloudera

Active Directory and Kerberos

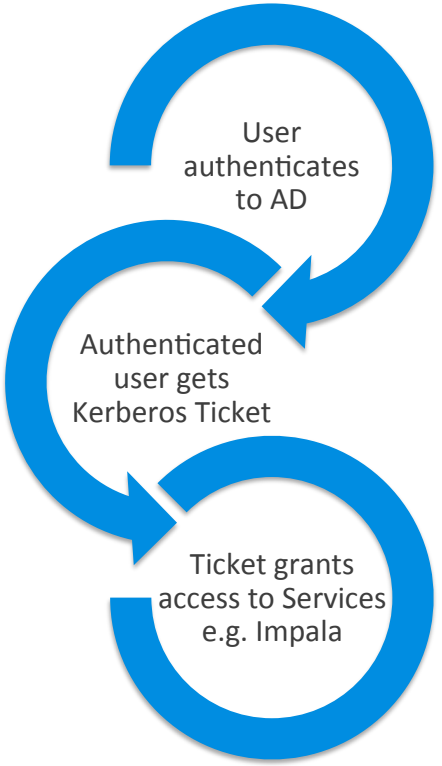
Active Directory

- Manages Users, Groups, and Services
- Provides username / password authentication
- Group membership determines Service access

Kerberos

- Trusted and standard third-party
- Authenticated users receive “Tickets”
- “Tickets” gain access to Services

User [ssmith]
Password[*****]



cloudera

Fine-Grained Access Control in HDFS

Across All Hadoop Paths

Columns:

Sensitive column visibility varies by role (Ex. credit card numbers)

- Managers: 1234 5678 1234 5678
- Call Center: XXXX XXXX XXXX 5678
- Analysts: XXXX XXXX XXXX XXXX
- Others: No access to credit card column

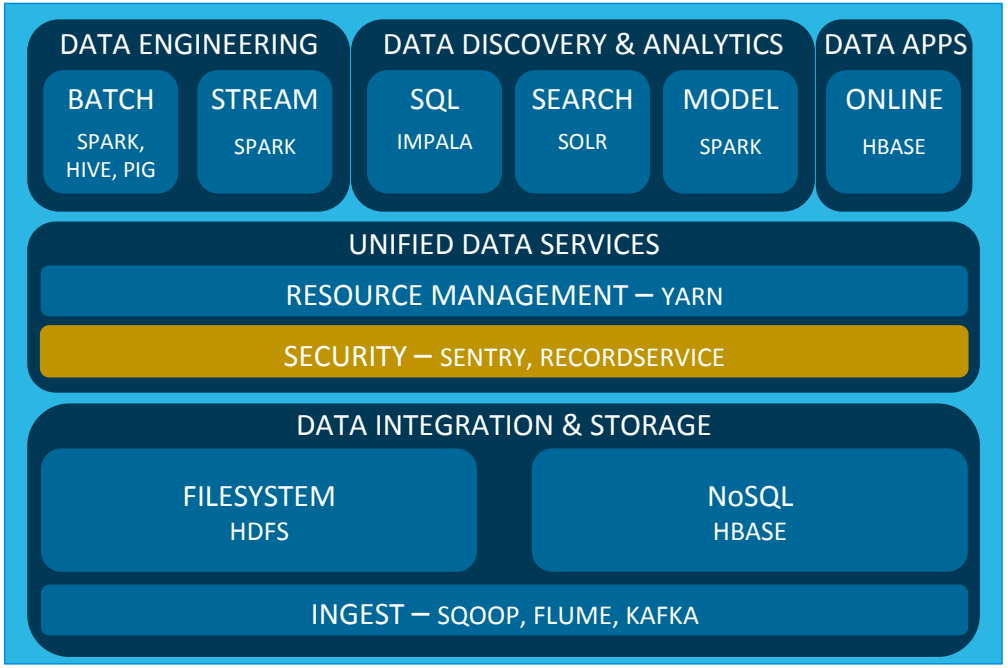
Rows:

Different user groups need access to different records

- European privacy laws
- Government security clearance
- Financial information restrictions

cloudera

RecordService
Unified Access Control Enforcement



- New high performance security layer that centrally enforces access control policies across Hadoop
  - Complements Apache Sentry's unified policy definition
  - Row- and column-based security
  - Dynamic data masking
- Apache-licensed open source
- Beta now available

cloudera

Fine-Grained HDFS Access without RecordService

Split the original file
Use HDFS permissions to limit access

Date/time	Accnt #	SSN	Asset	Trade	Country
09:33:11 16-Feb-2015	0234837823	238-23-9876	AAPL	Sell	US
11:33:01 16-Feb-2015	3947848494	329-44-9847	TBT	Buy	EU
14:12:34 16-Feb-2015	4848367383	123-56-2345	IBM	Sell	UK
09:22:03 16-Feb-2015	3485739384	585-11-2345	INTC	Buy	US
11:55:33 16-Feb-2015	3847598390	234-11-8765	F	Buy	US
10:22:55 16-Feb-2015	8765432176	344-22-9876	UA	Buy	UK
13:45:24 16-Feb-2015	3456789012	412-22-8765	AMZN	Sell	EU
09:03:44 16-Feb-2015	4857389329	123-44-5678	TMV	Buy	US
15:55:55 16-Feb-2015	4756983234	234-76-9274	MA	Buy	UK

Date/time	Accnt #	SSN	Asset	Trade	Country
09:33:11 16-Feb-2015	0234837823	238-23-9876	AAPL	Sell	US
09:22:03 16-Feb-2015	3485739384	585-11-2345	INTC	Buy	US
11:55:33 16-Feb-2015	3847598390	234-11-8765	F	Buy	US
09:03:44 16-Feb-2015	4857389329	123-44-5678	TMV	Buy	US

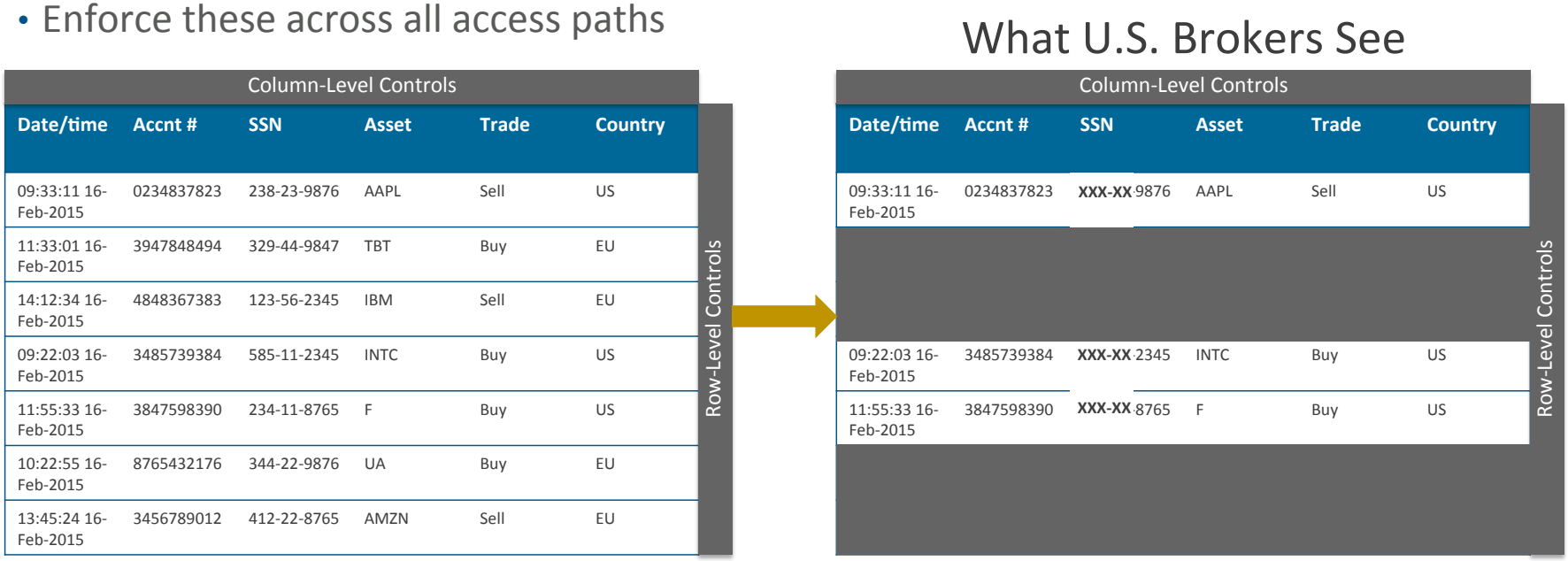
Date/time	Accnt #	SSN	Asset	Trade	Country
11:33:01 16-Feb-2015	3947848494	329-44-9847	TBT	Buy	EU
13:45:24 16-Feb-2015	3456789012	412-22-8765	AMZN	Sell	EU

Date/time	Accnt #	SSN	Asset	Trade	Country
14:12:34 16-Feb-2015	4848367383	123-56-2345	IBM	Sell	UK
10:22:55 16-Feb-2015	8765432176	344-22-9876	UA	Buy	UK
15:55:55 16-Feb-2015	4756983234	234-76-9274	MA	Buy	UK

cloudera

Fine-Grained HDFS Access Control with RecordService

- Apply controls to the master data file
- Row, column, and sub-column (masking) controls
- Enforce these across all access paths



cloudera

Spark Resources

- Learn Spark
  - Spark Cookbook – by Rishi Yadav
  - O'Reilly Advanced Analytics with Spark eBook (written by Clouderans)
  - Cloudera Developer Blog
  - cloudera.com/spark
- Get Trained
  - Cloudera Spark Training

cloudera