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 company snapshot

<table>
<thead>
<tr>
<th>Founded</th>
<th>2008, by former employees of <strong>Oracle</strong>, <strong>Yahoo!</strong>, <strong>Facebook</strong>, <strong>Google</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Employees Today</td>
<td>900+ worldwide</td>
</tr>
<tr>
<td>World Class Support</td>
<td>More than 75 24x7 global staff</td>
</tr>
<tr>
<td>Cloudera University</td>
<td>Over 40,000 trained</td>
</tr>
<tr>
<td>We help code Hadoop</td>
<td>Cloudera employees are leading developers &amp; contributors to the complete Apache Hadoop ecosystem of projects</td>
</tr>
<tr>
<td>We help fix Hadoop</td>
<td>Cloudera fixed 60% of all Hadoop JIRA bugs</td>
</tr>
</tbody>
</table>
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
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’s Engineering Commitment to Spark

Spark Committers by Hadoop Distribution*

- Cloudera: 67%
- Intel: 17%
- Hortonworks: 17%

Spark Patches by Hadoop Distribution

- Cloudera: 370
- Hortonworks: 4
- IBM: 12
- MapR: 1
- Intel: 400

* IBM and MapR have 0 committers
Spark will replace MapReduce
To become the standard execution engine for Hadoop
The Future of Data Processing on Hadoop
Spark complemented by specialized fit-for-purpose engines

General Data Processing w/ Spark
Fast Batch Processing, Machine Learning, and Stream Processing

Full-Text Search w/ Solr
Querying textual data

Analytic Database w/ Impala
Low-Latency Massively Concurrent Queries

On-Disk Processing w/ MapReduce
Jobs at extreme scale and extremely disk IO intensive

Shared:
- Data Storage
- Metadata
- Resource Management
- Administration
- Security
- Governance
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
Spark Overview and Improvements
Apache Spark
Flexible, in-memory data processing for Hadoop

- Easy Development
  - Rich APIs for Scala, Java, and Python
  - Interactive shell

- Flexible Extensible API
  - APIs for different types of workloads:
    - Batch
    - Streaming
    - Machine Learning
    - Graph

- Fast Batch & Stream Processing
  - In-Memory processing and caching
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
Python Or Scala?

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

• Use Scala for development
  • Steep learning curve for functional programming
You can develop interactively with the Spark shell and get results immediately.

```
scala>
val words = sc.textFile("file:/usr/share/dict/words")

scala> words.count
res0: Long = 235886

scala>
```

For example:

```
percolateur:spark srowen$ ./bin/spark-shell --master local[*]
...Welcome to ...

// 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.
...
```

- Interactive exploration of data for data scientists
- No need to develop “applications”
- Developers can prototype application on live system
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
- …
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
Spark Concepts

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

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

Diagram:
- HDFS File → Filtered RDD
  - `filter`
    (func = `startsWith(...)`) 
- Filtered RDD → Mapped RDD
  - `map`
    (func = `split(...)`)
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...
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.
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

Benefits:

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

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
Spark Streaming Architectures

Data Sources
- Flume
- Kafka

Integration Layer
- Ingest

Spark Stream Processing
- Data Prep
- Aggregation / Scoring

HDFS
- High-Fidelity Archival
- Transformed Results

HBase
- Real-Time Result Serving

Spark Long-Term Analytics/Model Building
- Application Notifications
- Real-Time Serving
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)
Impala Remains Tool of Choice for Interactive SQL

![Chart showing performance comparison between Impala, Spark SQL, Presto, and Hive-on-Tez. The chart indicates that Impala is consistently faster, with response times being significantly lower than other tools.](chart.png)

- **Impala**
  - Single User: 5.0x faster
  - 10 Users: 10.6x faster

- **Spark SQL**
  - Single User: 10.6x faster
  - 10 Users: 27.4x faster

- **Presto**
  - Single User: 7.4x faster
  - 10 Users: 15.4x faster

- **Hive-on-Tez**
  - Single User: 18.3x faster
  - 10 Users: 20.2x faster

*(Lower bars = better)*
Pluggable Spark – replace MapReduce

Cloudera is leading community development to port components to Spark:

- **Stage 1**
  - Crunch on Spark
  - Search on Spark

- **Stage 2**
  - Hive on Spark (beta)
  - Spark on HBase (beta)

- **Stage 3**
  - Pig on Spark (alpha)
  - Sqoop on Spark
  - Spark on Kudu
Spark Customer Use Cases

Core Spark

- Portfolio Risk Analysis
- ETL Pipeline Speed-Up
- 20+ years of stock data
- Identify disease-causing genes in the full human genome
- Calculate Jaccard scores on health care data sets
- Optical Character Recognition and Bill Classification
- Trend analysis
- Document classification (LDA)
- Fraud analytics

Spark Streaming

- Online Fraud Detection
- Incident Prediction for Sepsis
- Online Recommendation Systems
- Real-Time Inventory Management
- Real-Time Ad Performance Analysis
Doing the Math – Executors and Cores

Determine the optimal resource allocation for the Spark job

1 Core for Application Master
1 Core for Executors

Allocate Executors

16 Total Cores in Cluster
Core Allocation

1 Executor x 15 Cores = 3 Executors with 4 Cores Each
(Leaves 3 Cores un-utilized)

1 Executor x 15 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/
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
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
Resources

Join the community
http://getkudu.io

Download the Beta
cloudera.com/downloads

Read the Whitepaper
getkudu.io/kudu.pdf
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
Comprehensive, Compliance-Ready Security

Authentication, Authorization, Audit, and Compliance

**Perimeter**
Guarding access to the cluster itself

**Technical Concepts:**
- Authentication
- Network isolation

**Access**
Defining what users and applications can do with data

**Technical Concepts:**
- Permissions
- Authorization

**Visibility**
Reporting on where data came from and how it’s being used

**Technical Concepts:**
- Auditing
- Lineage

**Data**
Protecting data in the cluster from unauthorized visibility

**Technical Concepts:**
- Encryption, Tokenization, Data masking

**Cloudera Manager**

**Apache Sentry & RecordService**

**Cloudera Navigator**

**Navigator Encrypt & Key Trustee | Partners**
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 authenticates to AD
Authenticated user gets Kerberos Ticket
Ticket grants access to Services e.g. Impala

User [ssmith]
Password[*****]
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
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
Fine-Grained HDFS Access without RecordService

Split the original file
Use HDFS permissions to limit access

<table>
<thead>
<tr>
<th>Date/time</th>
<th>Acnt #</th>
<th>SSN</th>
<th>Asset</th>
<th>Trade</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:33:11 16-Feb-2015</td>
<td>0234837823</td>
<td>238-23-9876</td>
<td>AAPL</td>
<td>Sell</td>
<td>US</td>
</tr>
<tr>
<td>11:33:01 16-Feb-2015</td>
<td>3947848494</td>
<td>329-44-9847</td>
<td>TBT</td>
<td>Buy</td>
<td>EU</td>
</tr>
<tr>
<td>14:12:34 16-Feb-2015</td>
<td>4848367383</td>
<td>123-56-2345</td>
<td>IBM</td>
<td>Sell</td>
<td>UK</td>
</tr>
<tr>
<td>09:22:03 16-Feb-2015</td>
<td>3485739384</td>
<td>585-11-2345</td>
<td>INTC</td>
<td>Buy</td>
<td>US</td>
</tr>
<tr>
<td>09:03:44 16-Feb-2015</td>
<td>4857389329</td>
<td>123-44-5678</td>
<td>TMV</td>
<td>Buy</td>
<td>US</td>
</tr>
<tr>
<td>09:33:11 16-Feb-2015</td>
<td>0234837823</td>
<td>238-23-9876</td>
<td>AAPL</td>
<td>Sell</td>
<td>US</td>
</tr>
<tr>
<td>09:22:03 16-Feb-2015</td>
<td>3485739384</td>
<td>585-11-2345</td>
<td>INTC</td>
<td>Buy</td>
<td>US</td>
</tr>
<tr>
<td>09:03:44 16-Feb-2015</td>
<td>4857389329</td>
<td>123-44-5678</td>
<td>TMV</td>
<td>Buy</td>
<td>US</td>
</tr>
</tbody>
</table>
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

### What U.S. Brokers See

<table>
<thead>
<tr>
<th>Date/Time</th>
<th>Account #</th>
<th>SSN</th>
<th>Asset</th>
<th>Trade</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:33:11 16-Feb-2015</td>
<td>0234837823</td>
<td>238-23-9876</td>
<td>AAPL</td>
<td>Sell</td>
<td>US</td>
</tr>
<tr>
<td>11:33:01 16-Feb-2015</td>
<td>3947848494</td>
<td>329-44-9847</td>
<td>TBT</td>
<td>Buy</td>
<td>EU</td>
</tr>
<tr>
<td>14:12:34 16-Feb-2015</td>
<td>4848367383</td>
<td>123-56-2345</td>
<td>IBM</td>
<td>Sell</td>
<td>EU</td>
</tr>
<tr>
<td>09:22:03 16-Feb-2015</td>
<td>3485739384</td>
<td>585-11-2345</td>
<td>INTC</td>
<td>Buy</td>
<td>US</td>
</tr>
<tr>
<td>13:45:24 16-Feb-2015</td>
<td>3456789012</td>
<td>412-22-8765</td>
<td>AMZN</td>
<td>Sell</td>
<td>EU</td>
</tr>
</tbody>
</table>

© Cloudera, Inc. All rights reserved.
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