Hortonworks

Hortonworks Data Platform

November 2015

Guilherme Braccialli
Solutions Engineer - LATAM
gbraccialli@hortonworks.com
Skype: gbraccialli
+55 11 9 8134 1618

Agenda – Meetup Machine Learning – 05/11/2015

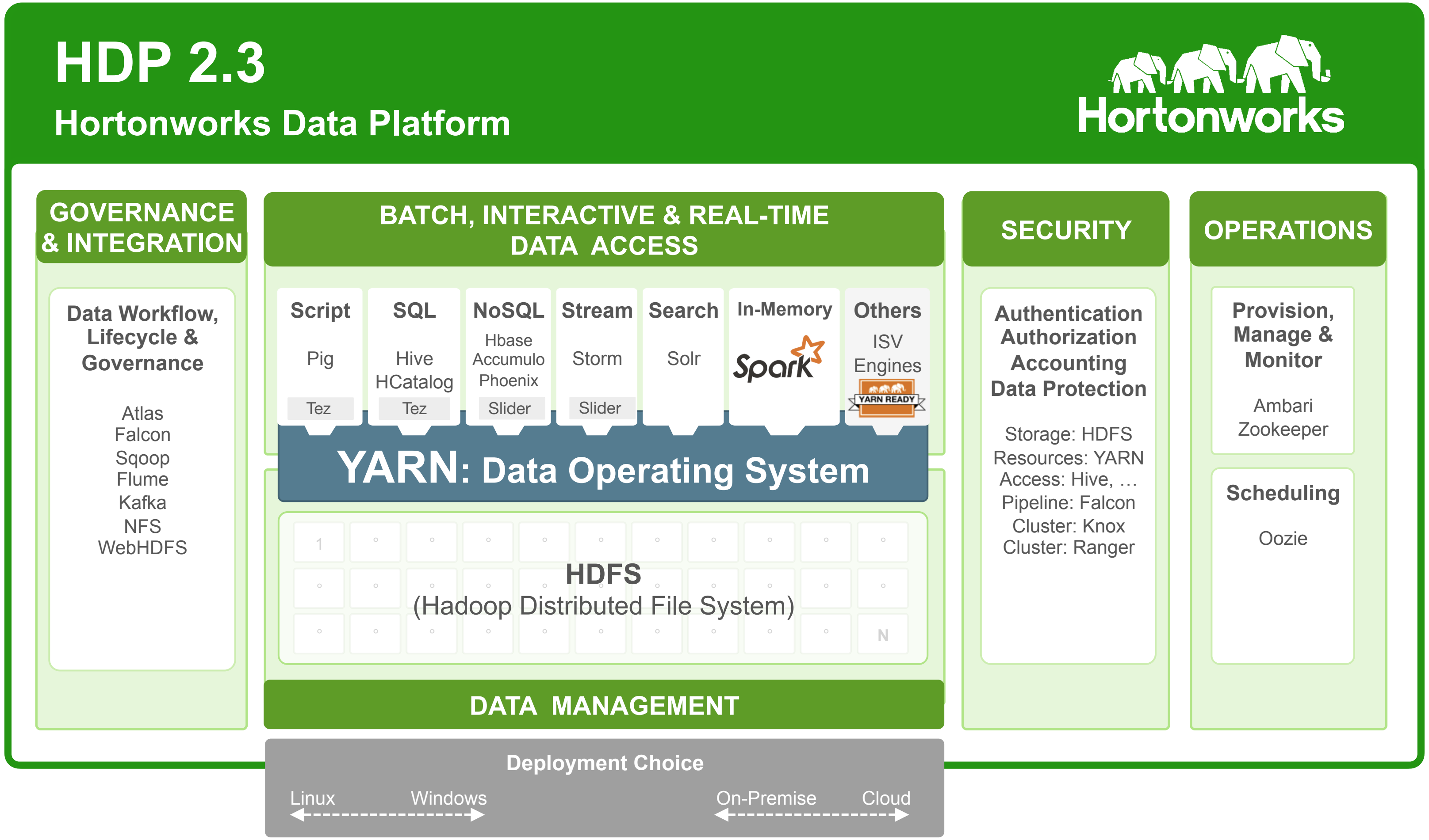
- ✓ Indrodução – 20 minutos
  - ✓ Hortonworks
  - ✓ Spark
  - ✓ Zeppelin
- ✓ Demo – Parte 1 – 10 minutos
- ✓ Spark Streaming – 10 minutos
- ✓ Demo – Parte 2 – 10 minutos
- ✓ Apoio – 30 minutos
- ✓ Dúvidas – 10 minutos

Hortonworks®

Hortonworks

HDP delivers a completely open data platform

Hortonworks Data Platform provides Hadoop for the Enterprise: a centralized architecture of core enterprise services, for any application and any data.

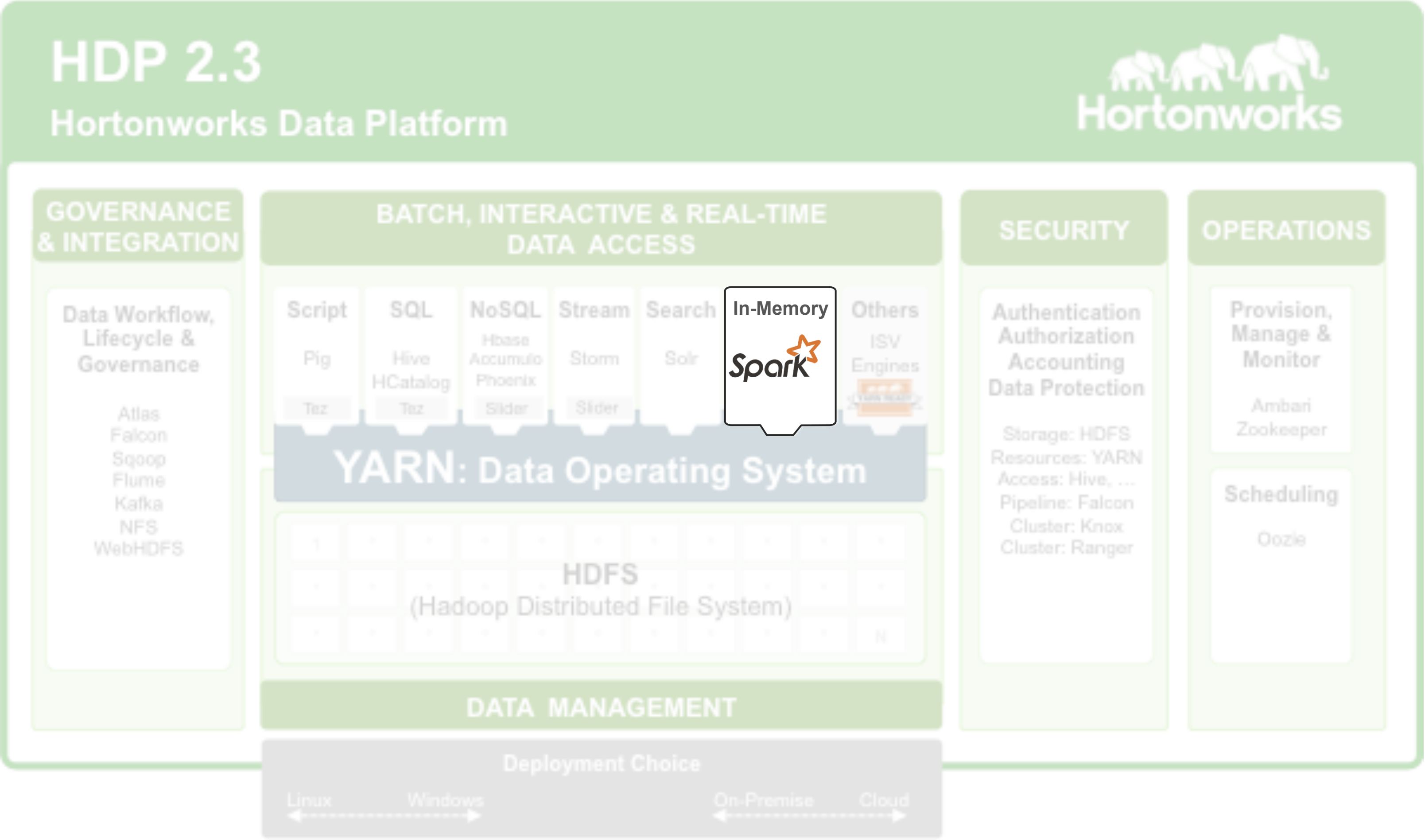


Completely Open

- HDP incorporates every element required of an enterprise data platform: data storage, data access, governance, security, operations
- All components are developed in open source and then rigorously tested, certified, and delivered as an integrated open source platform that's easy to consume and use by the enterprise and ecosystem.

Hortonworks®

Spark


HDP 2.3
Hortonworks Data Platform
Hortonworks
GOVERNANCE & INTEGRATION
BATCH, INTERACTIVE & REAL-TIME DATA ACCESS
SECURITY
OPERATIONS
Data Workflow, Lifecycle & Governance
Atlas
Falcon
Sqoop
Flume
Kafka
NFS
WebHDFS
Script
Pig
Tez
SQL
Hive
HCatalog
Tez
NoSQL
Hbase
Accumulo
Phoenix
Slider
Stream
Storm
Slider
Search
Solr
In-Memory
Spark
Others
ISV
Engines
YARN: Data Operating System
HDFS
(Hadoop Distributed File System)
DATA MANAGEMENT
Authentication
Authorization
Accounting
Data Protection
Storage: HDFS
Resources: YARN
Access: Hive, …
Pipeline: Falcon
Cluster: Knox
Cluster: Ranger
Provision, Manage & Monitor
Ambari
Zookeeper
Scheduling
Oozie
Deployment Choice
Linux
Windows
On-Premise
Cloud

Spark

What is Apache Spark?

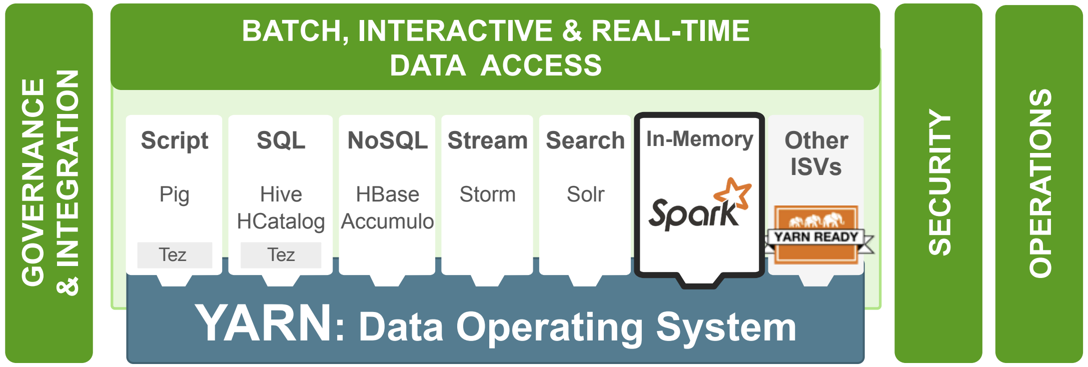
- Spark is top level Apache Project since February 2014, originally a graduate project at UC Berkeley's AMPLab
- Spark is a general-purpose engine for ad-hoc interactive analytics, iterative machine-learning, and other use cases well-suited to interactive, in-memory data processing of GB to TB sized datasets.
- Spark loads data into memory so it can be queried repeatedly. It can create a “shadow” of data that can be used in the next iteration of a query
- Spark provides simple APIs for data scientists and engineers familiar with Scala (programming language) to build applications

- Spark is built on HDFS
- **Spark is YARN enabled**

Hortonworks®

Hortonworks Commitment to Spark

Hortonworks is focused on making Apache Spark enterprise ready so you can depend on it for mission critical applications



1. **YARN enable Spark to co-exist with other engines**
   We have already declared it "YARN Ready" so its memory & CPU intensive apps can work with predictable performance along side other engines all on the same set(s) of data.

2. **Extend Spark with enterprise capabilities**
   Ensure Spark can be managed, secured and governed all via a single set of frameworks to ensure consistency. Ensure reliability and quality of service of Spark along side other engines.

3. **Actively contribute within the open community**
   As with everything we do at Hortonworks we work entirely within the open community across Spark and all related projects to improve this key Hadoop technology.

Why We Love Spark at Hortonworks

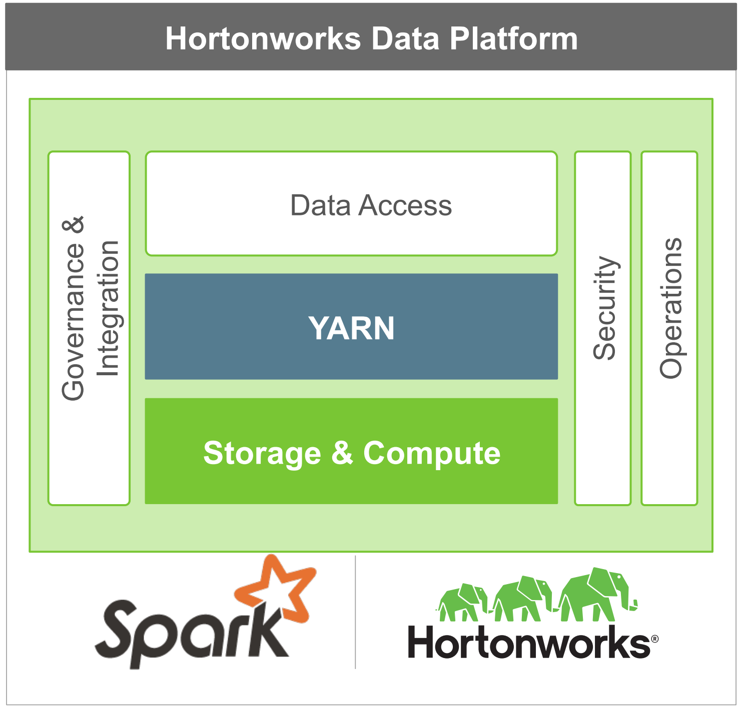
Made for Data Science
All apps need to get predictive at scale and fine granularity

Democratizes Machine Learning
Spark is doing to ML on Hadoop what Hive did for SQL on Hadoop

Elegant Developer APIs
DataFrames, Machine Learning and SQL

Realize Value of Data Operating System
A key tool in the Hadoop toolbox

Community
Broad developer, customer and partner interest



Hortonworks

Spark in Hadoop® with HDP 2.3

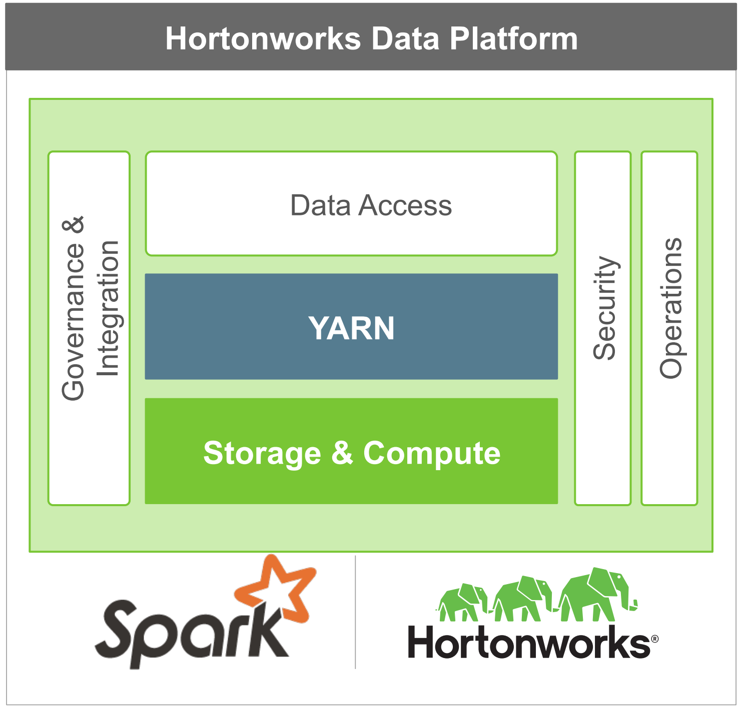
Resource Management
YARN for multi-tenant, diverse workloads with predictable SLAs

Tiered Memory Storage
HDFS in-memory tier—External BlockStore for RDD Cache

Deployable Anywhere
Linux, Windows and on-premises or cloud

Self-Service Spark in the Cloud
Easy launch of Data Science clusters via Cloudbreak and Ambari—for Azure, AWS, GCP, OpenStack and Docker

Operations
Deployment / management via Apache Ambari



Spark is Integrated into HDP's Centralized Architecture

HDP 2.3 Ships with Apache Spark 1.3.1

Production-ready

Centralized Resource Management

Run other workloads along with Spark

YARN provides capacity guarantees via Capacity Scheduler

Consistent Operations

Deployable anywhere

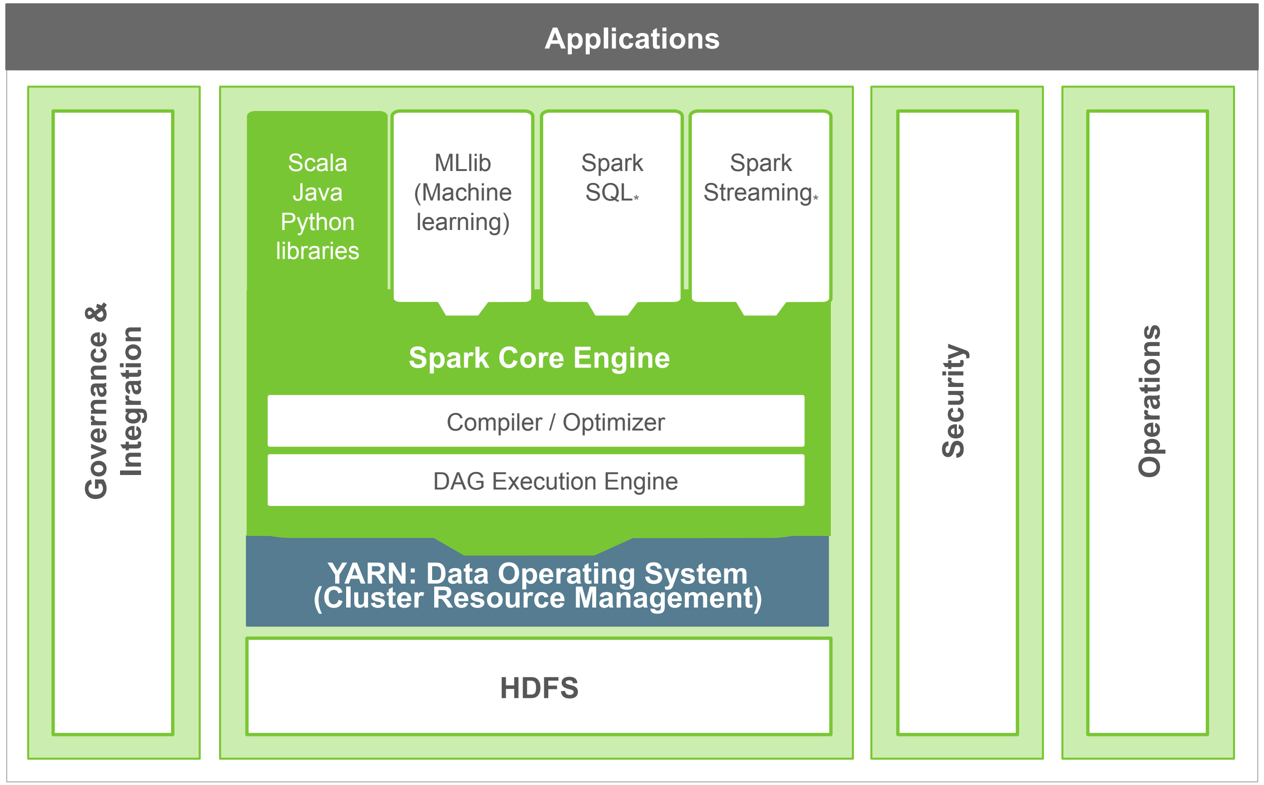
Ambari deploys and manages

Comprehensive Security

Improved Authentication—only way to run in a kerberized environment

With Speed and at Scale

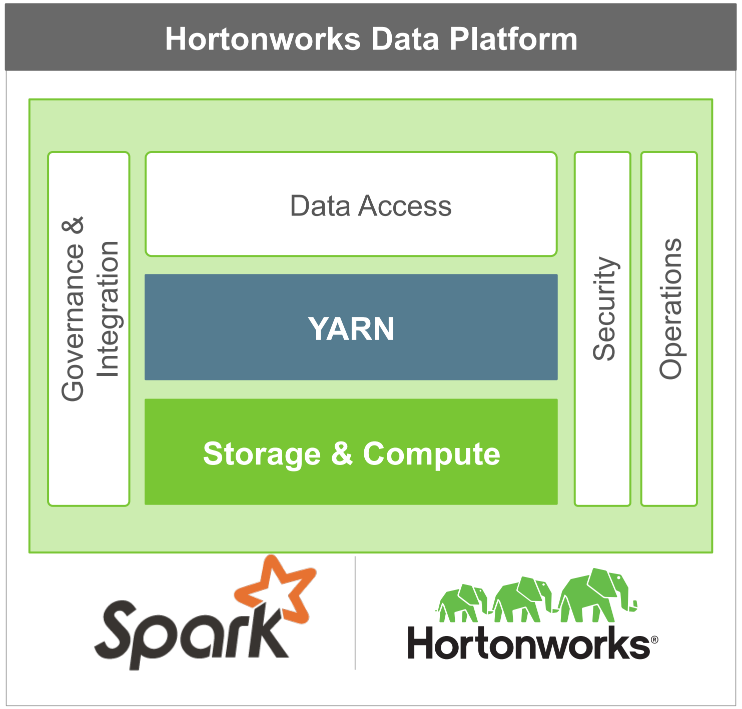
Vertical integration of Spark with YARN, HDFS and ORC



* Tech Preview

Hortonworks®

Hortonworks Focus for Spark



Easy of Use

Apache Zeppelin for interactive notebooks

Metadata and Governance

Apache Atlas for metadata & Apache Falcon support for Spark pipelines

Security

Apache Ranger managed authorization

Spark SQL and Hive for SQL

Interop with modern Metastore / HS2, optimized ORC support, advanced analytics—e.g., Geospatial

Spark and NoSQL

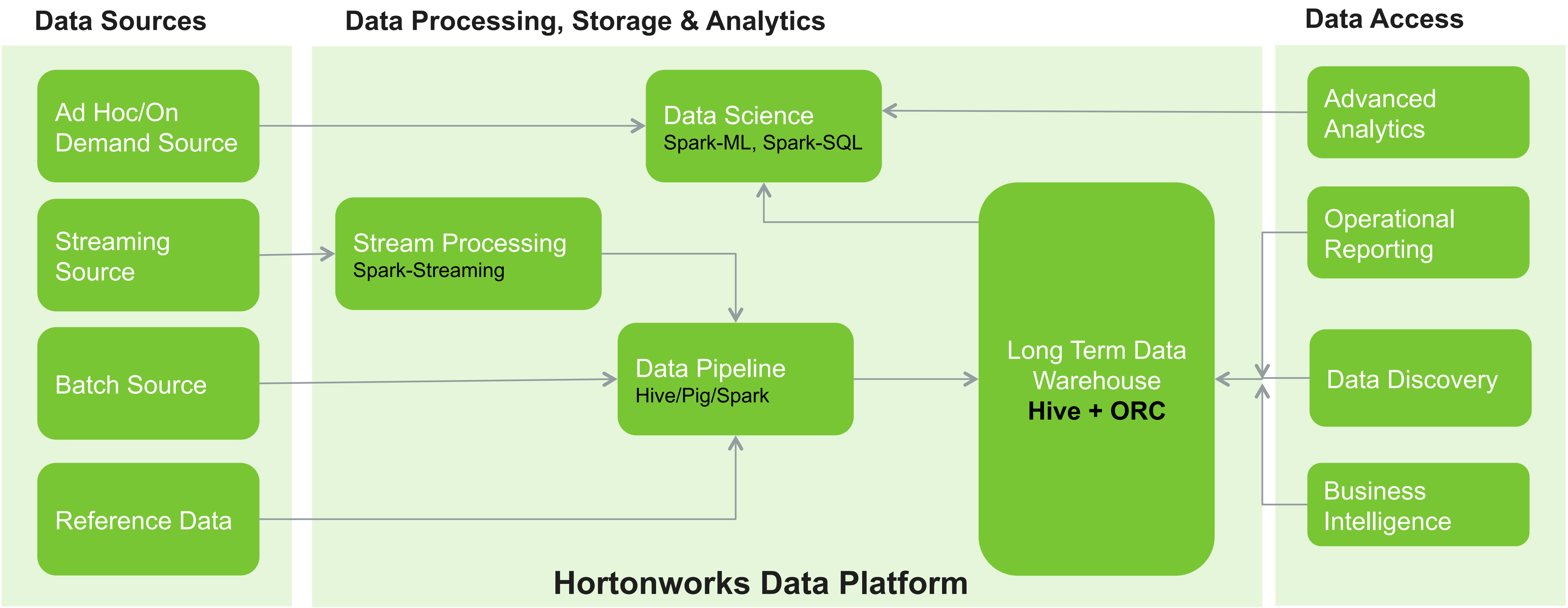
Deep integration with HBase via DataSources / Catalyst for Predicate / Aggregate Pushdown

Connect the Dots—Algorithms to Use-Cases

Higher-level ML Abstractions—e.g., OneVsRest

Validation, tuning, pipeline assembly—e.g., GeoSpatial

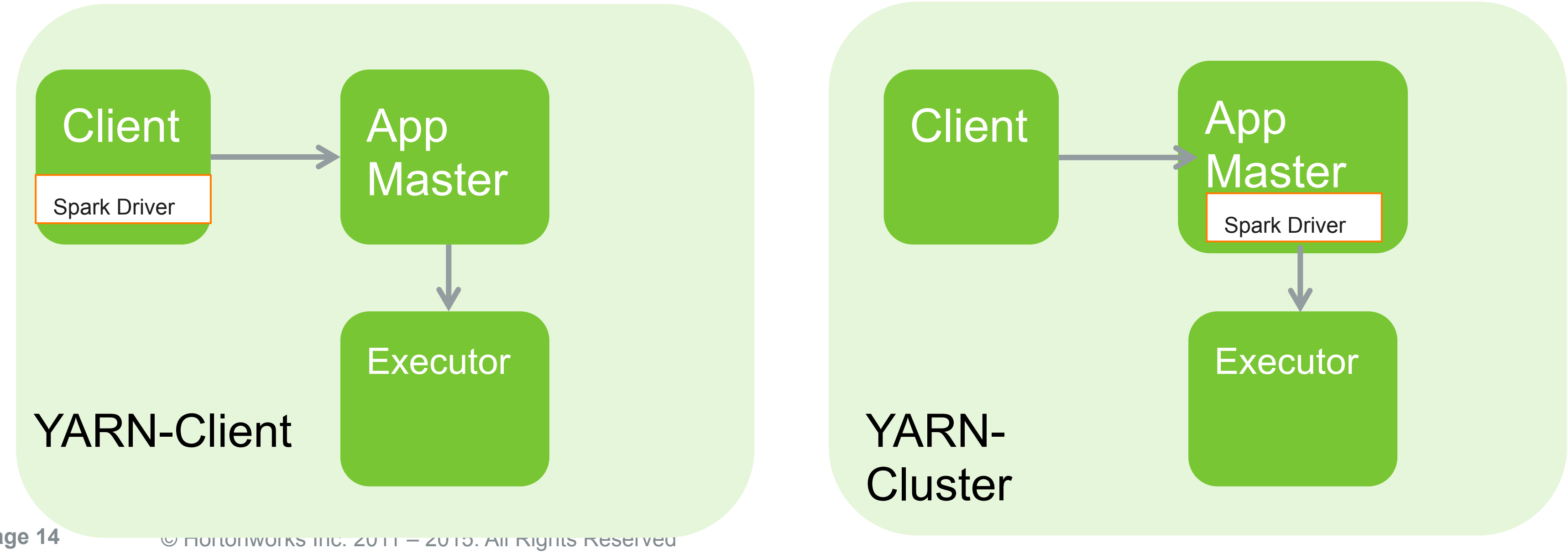
Reference Deployment Architecture



Spark Deployment Modes

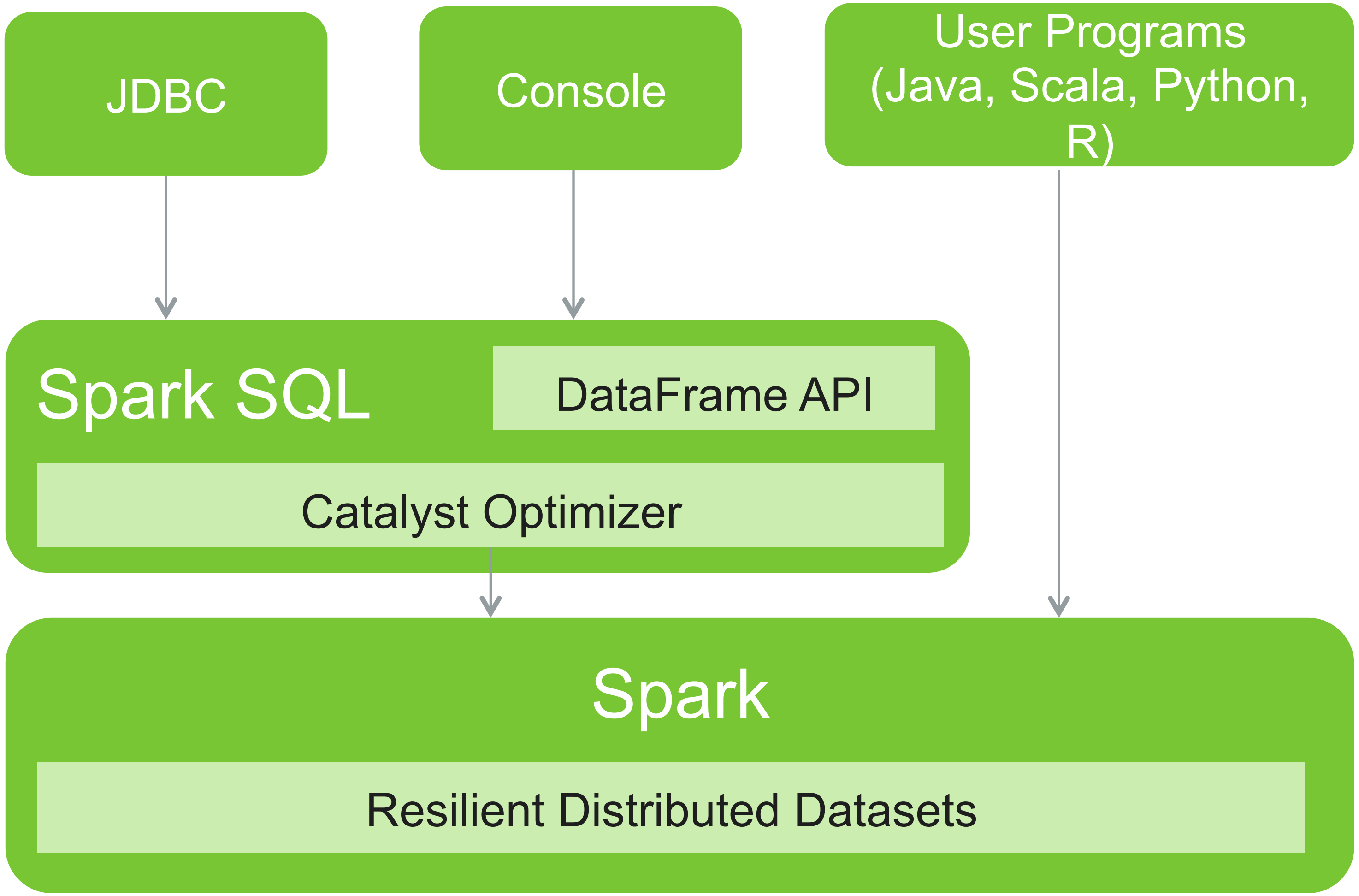
- Spark Standalone Cluster
  - For developing Spark apps against a local Spark (similar to develop/deploying in IDE)
- Spark on YARN
  - Spark driver (SparkContext) in YARN AM(yarn-cluster)
  - Spark driver (SparkContext) in local (yarn-client)
    - Spark Shell runs in yarn-client only

Mode setup with Ambari



Hortonworks®

Interfaces to Spark SQL



Hortonworks®

Current State of Security in Spark

HDP 2.3

Only Spark on YARN supports Kerberos today

Leverage Kerberos for authentication

Spark reads data from HDFS and ORC

HDFS file permissions (and Ranger integration) applicable to Spark jobs

Spark submits job to YARN queue

YARN queue ACL (and Ranger integration) applicable to Spark jobs

Wire Encryption

Spark has some coverage, not all channels are covered

LDAP Authentication

No authentication in Spark UI OOB, supports filter for hooking in LDAP

Hortonworks®

spark-shell and pyspark shell

```
15/07/03 19:05:19 INFO AbstractConnector: Started SocketConnector@0.0.0.0:49270
15/07/03 19:05:19 INFO Utils: Successfully started service 'HTTP class server' on port 49270.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.3.1
      /_/

Using Scala version 2.10.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_45)
Type in expressions to have ther
Type :help for more information.
15/07/03 19:05:21 INFO SparkCont
15/07/03 19:05:21 INFO SecurityM
15/07/03 19:05:21 INFO SecurityM
15/07/03 19:05:21 INFO SecurityM
fy permissions: Set(mlong)
```

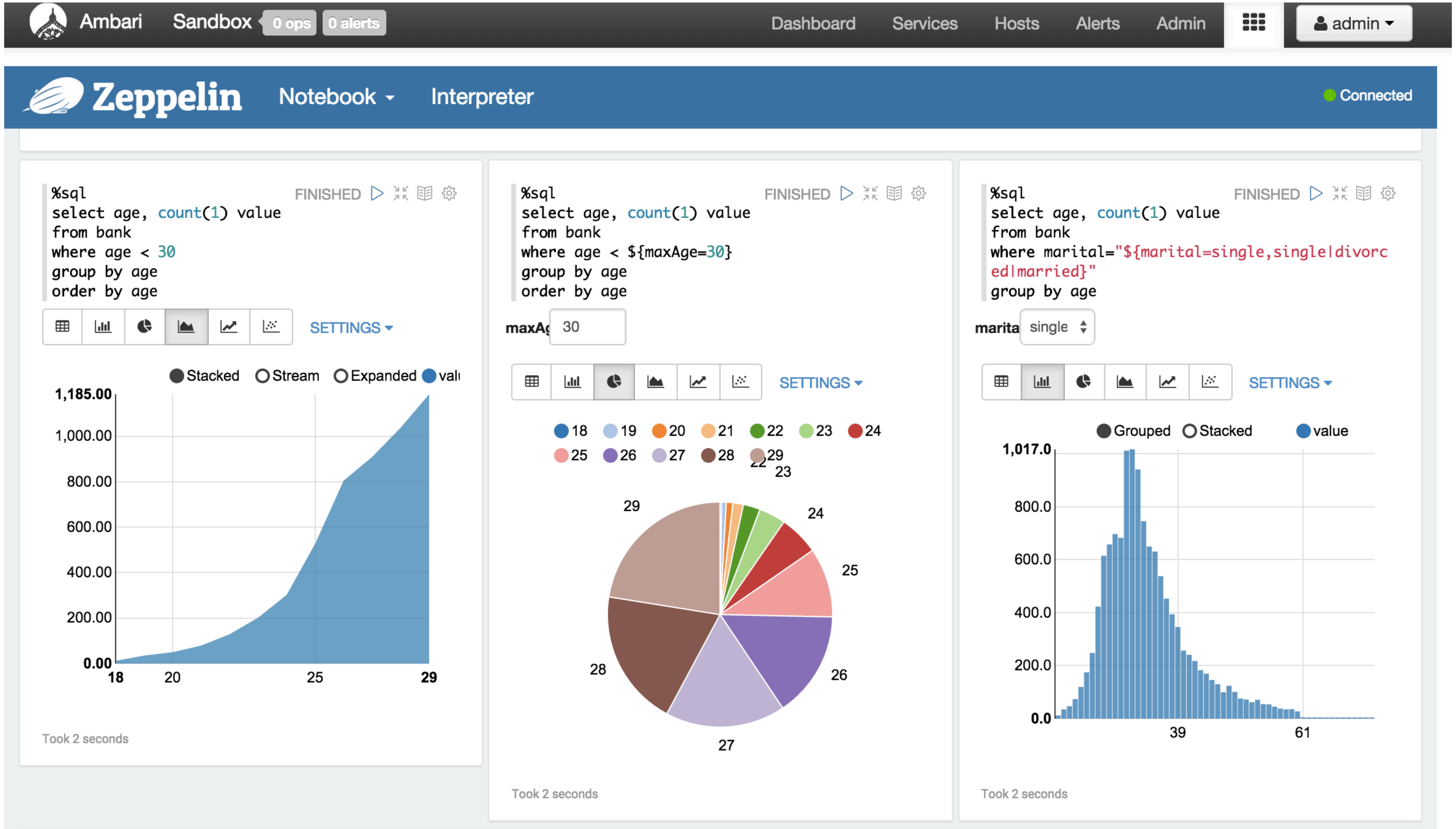
```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 1.3.1
      /_/

Using Python version 2.7.6 (default, Sep  9 2014 15:04:36)
SparkContext available as sc, HiveContext available as sqlContext.
>>>
```


Hortonworks®

Zeppelin

Introducing Apache Zeppelin



Hortonworks

Apache Zeppelin

Features	Use Cases
A web-based notebook for interactive analytics —Ad-hoc experimentation with Spark, Hive, Shell, Flink, Tajo, Ignite, Lens, etc	Data exploration and discovery
Deeply integrated with Spark and Hadoop —Can be managed via Ambari Stacks	Visualization—tables, graphs and charts
Supports multiple language backends —Pluggable "Interpreters"	Interactive snippet-at-a-time experience
Incubating at Apache —100% open source and open community	Collaboration and publishing
	"Modern Data Science Studio"

Hortonworks®

Apache Zeppelin

```
HW11718:bin mlong$ ./zeppelin-daemon.sh start
Zeppelin start                                             [  OK  ]
HW11718:bin mlong$
```

Zeppelin Notebook Interpreter

salary demo

```scala
val salaryData = sc.textFile("data/salarydata.txt")
```

```
salaryData: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[58] at textFile at <console>:24
```

Took 0 seconds

```scala
val genderSalaryData = salaryData.map(line => line.split(',')).map(line => (line(0), line(2).toInt))
genderSalaryData.cache()
genderSalaryData.collect()
```

```
genderSalaryData: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[60] at map at <console>:26
res25: genderSalaryData.type = MapPartitionsRDD[60] at map at <console>:26
res26: Array[(String, Int)] = Array((M,39000), (F,41000), (M,99000), (M,58000), (M,43000), (M,11000), (M
0), (F,96000), (M,37000), (F,53000), (F,27000), (F,0), (M,54000), (F,0), (F,45000), (M,57000), (M,16000)
M,0), (M,0), (M,75000), (F,0), (F,42000), (F,48000), (F,16000), (F,85000), (F,72000), (M,18000), (M,8100
0), (F,69000), (F,57000), (M,76000), (M,12000), (M,58000), (F,96000), (F,37000), (F,0), (F,20000), (M,0)
F,0), (M,99000), (F,68000), (F,0)...
```

Took 0 seconds

Hortonworks®

PySpark / Spark SQL

Zeppelin Notebook Interpreter

```
%pyspark
py_val=sc.parallelize(["M",14,0,95102])
print py_val.collect()
```

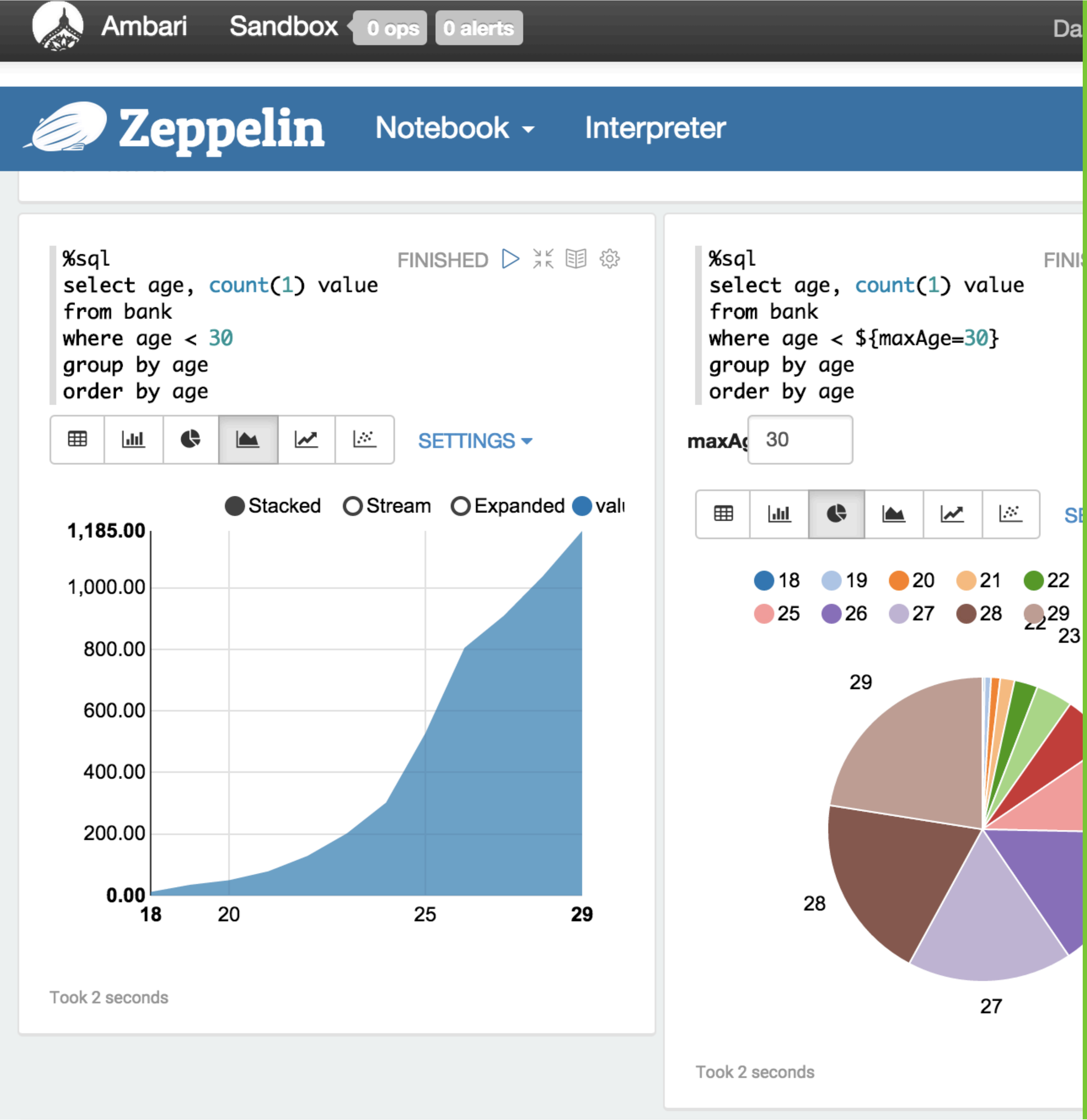
```
['M', 14, 0, 95102]
```

Took 0 seconds

Zeppelin Notebook Interpreter

```
%sql
select s.gender, s.salary from
(select d.gender, d.salary, row_number() over (partition by d.gender order by d.salary desc) rn
  from (select distinct gender, salary from salaries) d) s
where s.rn <= 10 order by gender, salary desc
```


Hortonworks



Web-based Notebook for interactive analytics

Features

Ad-hoc experimentation

Spark, Hive, Shell, Flink, Tajo, Ignite, Lens, etc

Deeply integrated with Spark + Hadoop

Can be managed via Ambari Stacks

Supports multiple language backends

Pluggable "Interpreters"

Incubating at Apache

100% open source and open community

Use Case

Data exploration and discovery

Visualization

tables, graphs and charts

Interactive snippet-at-a-time experience

Collaboration and publishing

"Modern Data Science Studio"

Demo – Parte 1

Hortonworks®

Spark Streaming

What is Spark Streaming?

- Extends Spark for doing large scale stream processing
  - Scales to 100s of nodes and achieves second scale latencies
- Efficient and fault-tolerant stateful stream processing
  - Simple batch-like API for implementing complex algorithms

Hortonworks

Motivation

- Many important applications must process large streams of live data and provide results in near-real-time
  - Social network trends
  - Website statistics
  - Ad impressions
- Distributed stream processing framework is required to
  - Scale to large clusters (100s of machines)
  - Achieve low latency (few seconds)

Integration with Batch Processing

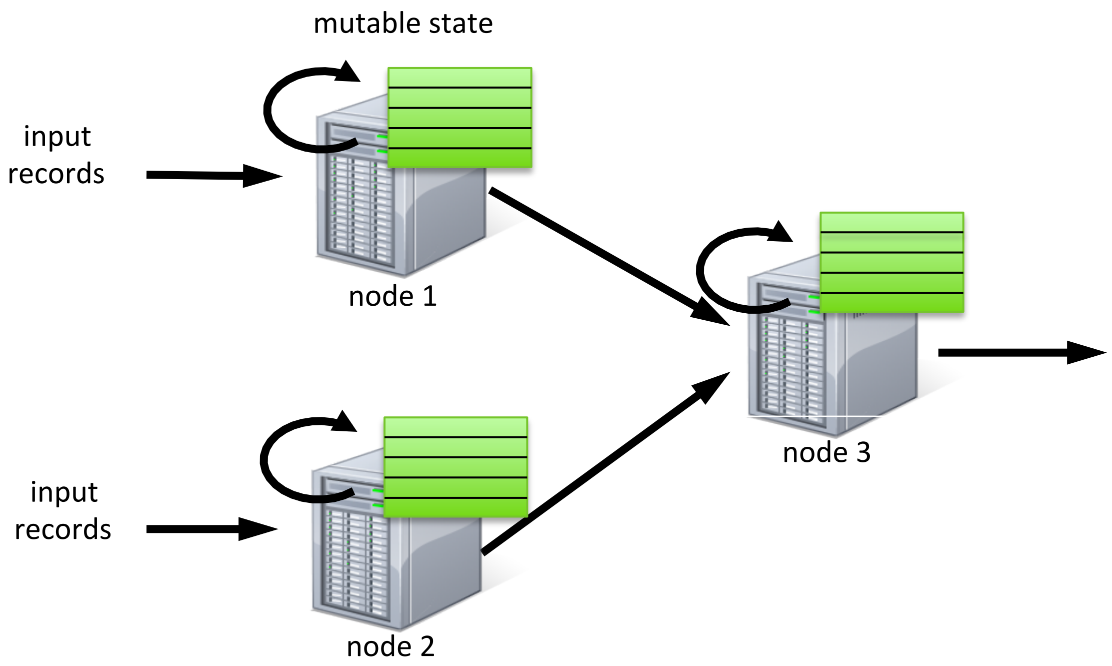
- Many environments require processing same data in live streaming as well as batch post processing
- Existing framework cannot do both
  - Either do stream processing of 100s of MB/s with low latency
  - Or do batch processing of TBs / PBs of data with high latency
- Extremely painful to maintain two different stacks
  - Different programming models
  - Double the implementation effort
  - Double the number of bugs

Stateful Stream Processing

- Traditional streaming systems have a record-at-a-time processing model
  – Each node has mutable state
  – For each record, update state and send new records

- State is lost if node dies.

- Making stateful stream processing be fault-tolerant is challenging



Hortonworks

Comparison with Storm

- Storm
  - Replays record if not processed by a node
  - Processes each record *at least once*
  - May update mutable state twice
  - Mutable state can be lost due to failure

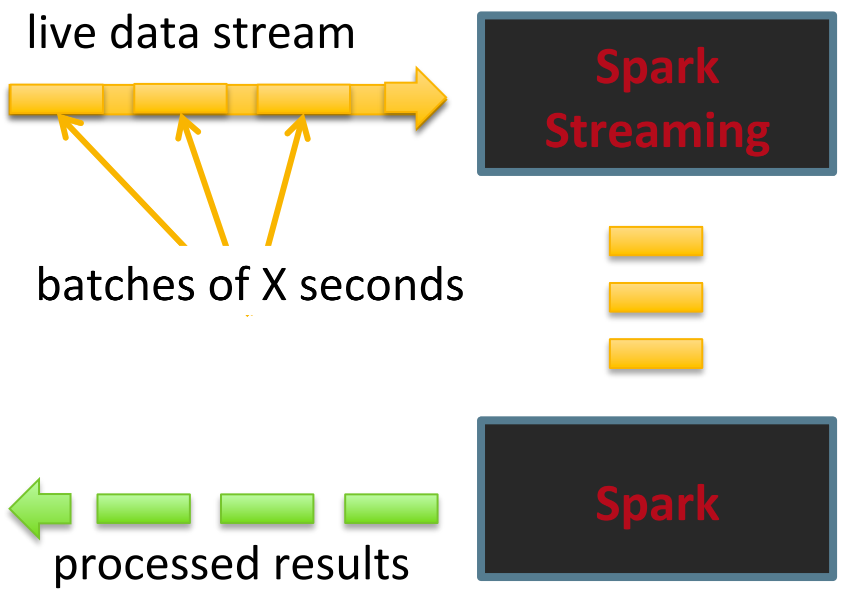
- Spark vs Storm: Use case dictates which one to use. Use Storm for at least once, and low latency SLAs. Can use Spark for SLAs > 500ms. Officially HWX doesn't support Spark Streaming and supports Storm.

Hortonworks®

Discretized Stream Processing

Run a streaming computation as a series of very small, deterministic batch jobs

- Chop up the live stream into batches of X seconds
- Spark treats each batch of data as RDDs and processes them using RDD operations
- Finally, the processed results of the RDD operations are returned in batches

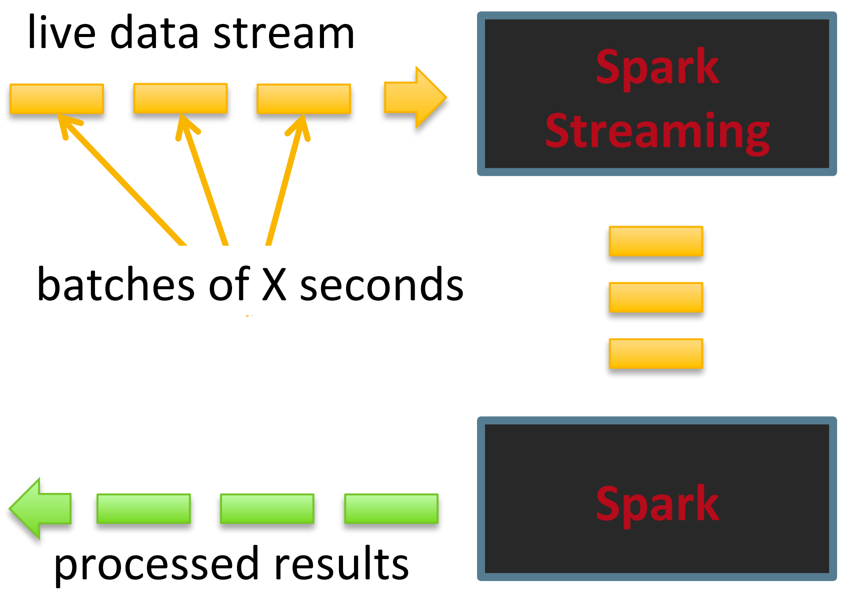


Hortonworks®

Discretized Stream Processing

Run a streaming computation as a series of very small, deterministic batch jobs

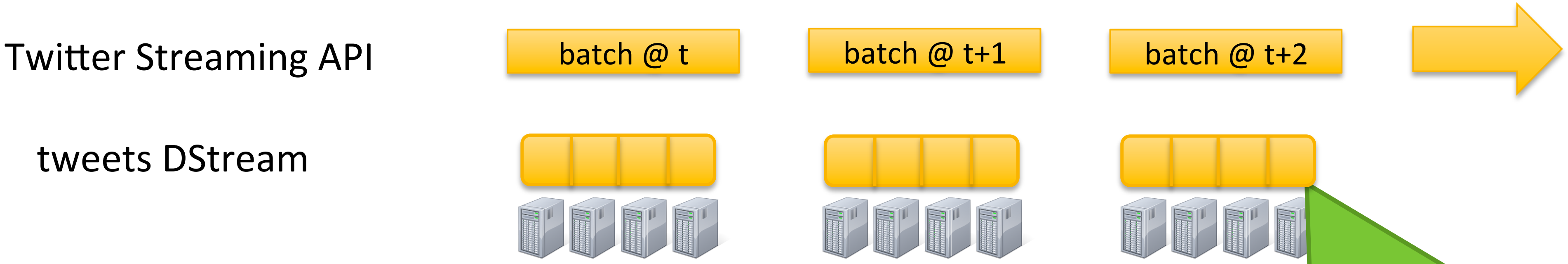
- Batch sizes as low as ½ second, latency of about 1 second
- Potential for combining batch processing and streaming processing in the same system



Example – Get hashtags from Twitter

```
val tweets = ssc.twitterStream()
```

DStream: a sequence of RDDs representing a stream of data



stored in memory as an RDD (immutable, distributed)

Hortonworks®

Example – Get hashtags from Twitter

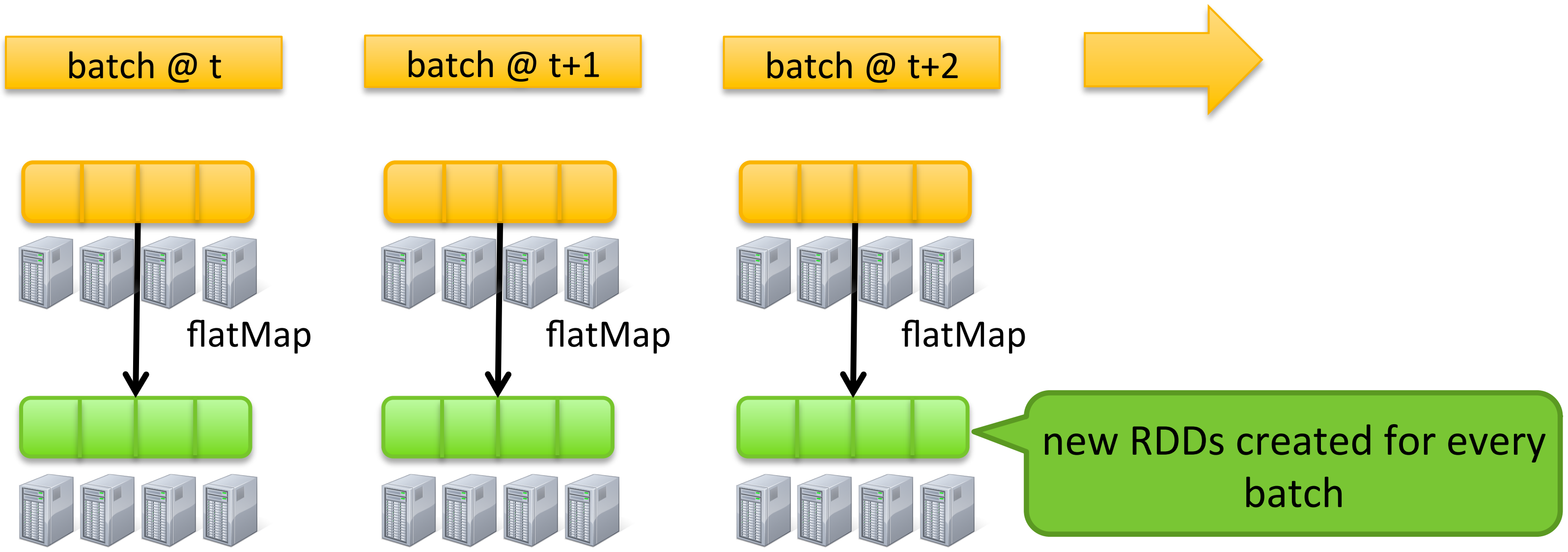
```
val tweets = ssc.twitterStream()
val hashTags = tweets.flatMap (status => getTags(status))
```

new DStream

transformation: modify data in one DStream to create another DStream

tweets DStream

hashTags Dstream
[#cat, #dog, …]



Hortonworks®

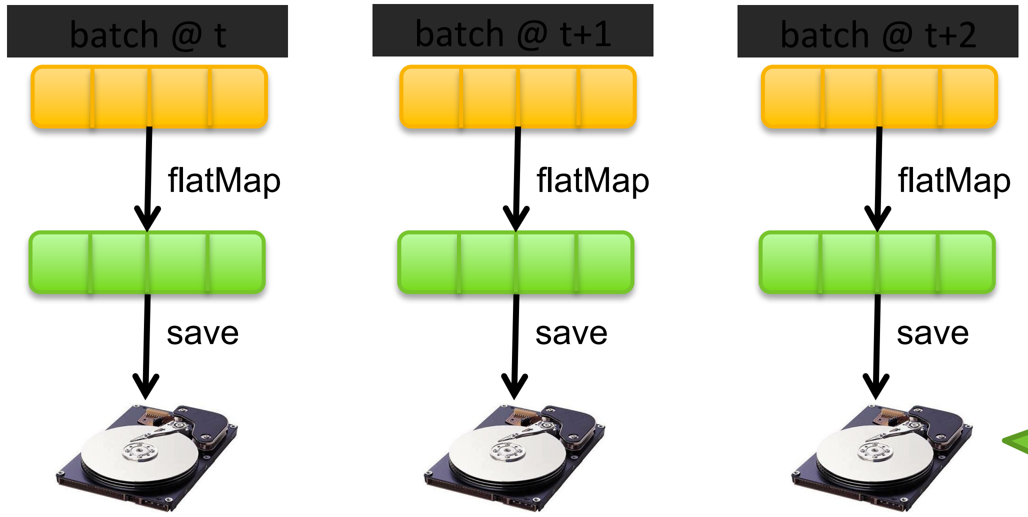
Example – Get hashtags from Twitter

```
val tweets = ssc.twitterStream()
val hashTags = tweets.flatMap (status => getTags(status))
hashTags.saveAsHadoopFiles("hdfs://...")
```

output operation: to push data to external storage

tweets DStream

hashTags DStream

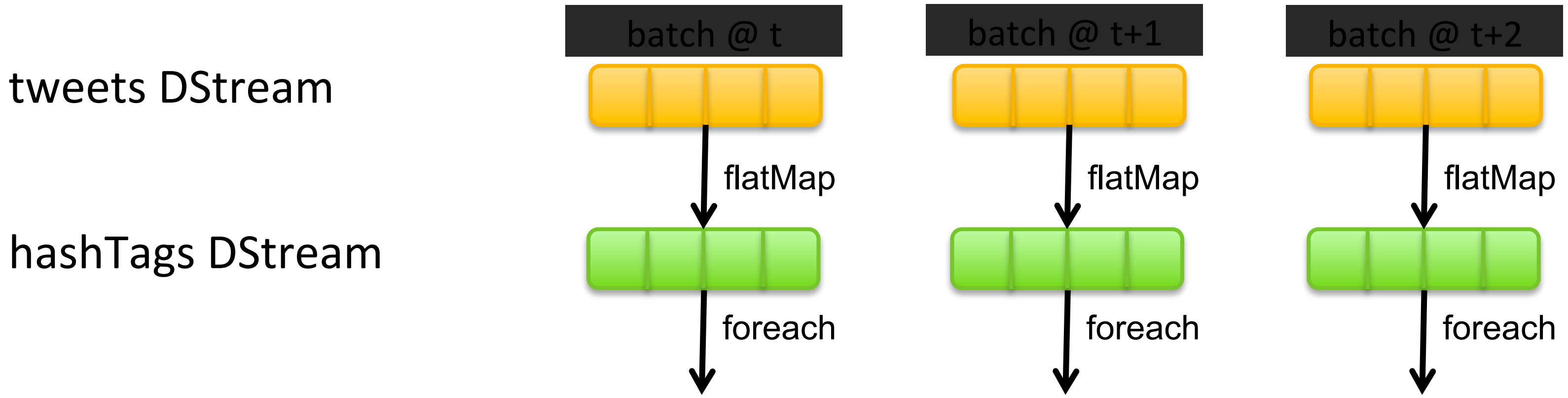


every batch saved to HDFS

Example – Get hashtags from Twitter

```
val tweets = ssc.twitterStream()
val hashTags = tweets.flatMap (status => getTags(status))
hashTags.foreach(hashTagRDD => { ... })
```

foreach: do whatever you want with the processed data



Write to database, update analytics UI, do whatever you want

Java Example

Scala

```
val tweets = ssc.twitterStream()
val hashTags = tweets.flatMap (status => getTags(status))
hashTags.saveAsHadoopFiles("hdfs://...")
```

Java

```
JavaDStream<Status> tweets = ssc.twitterStream()
JavaDstream<String> hashTags = tweets.flatMap(new Function<...> {  })
hashTags.saveAsHadoopFiles("hdfs://...")
```

Function object

Hortonworks®

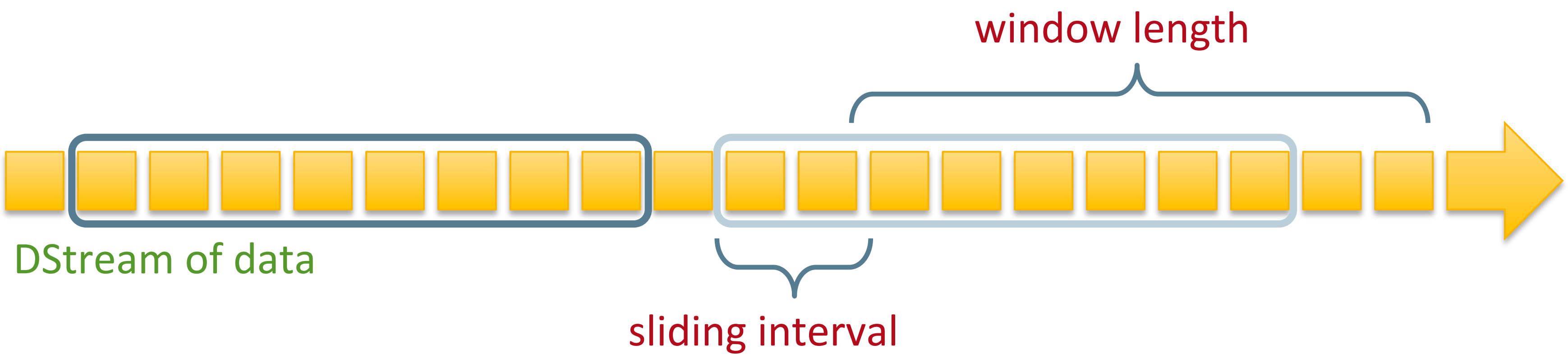
Window-based Transformations

```
val tweets = ssc.twitterStream()
val hashTags = tweets.flatMap (status => getTags(status))
val tagCounts = hashTags.window(Minutes(1), Seconds(5)).countByValue()
```

sliding window operation

window length

sliding interval



Hortonworks®

Arbitrary Stateful Computations

Specify function to generate new state based on previous state and new data

- Example: Maintain per-user mood as state, and update it with their tweets

```
updateMood(newTweets, lastMood) => newMood
moods = tweets.updateStateByKey(updateMood _)
```


Hortonworks

Arbitrary Combinations of Batch and Streaming Computations

Inter-mix RDD and DStream operations!

– Example: Join incoming tweets with a spam HDFS file to filter out bad tweets

```
tweets.transform(tweetsRDD => {
      tweetsRDD.join(spamHDFSFile).filter(...)
})
```


Hortonworks®

DStream Input Sources

- Out of the box we provide
  - Kafka
  - HDFS
  - Flume
  - Akka Actors
  - Raw TCP sockets

- Very easy to write a *receiver* for your own data source

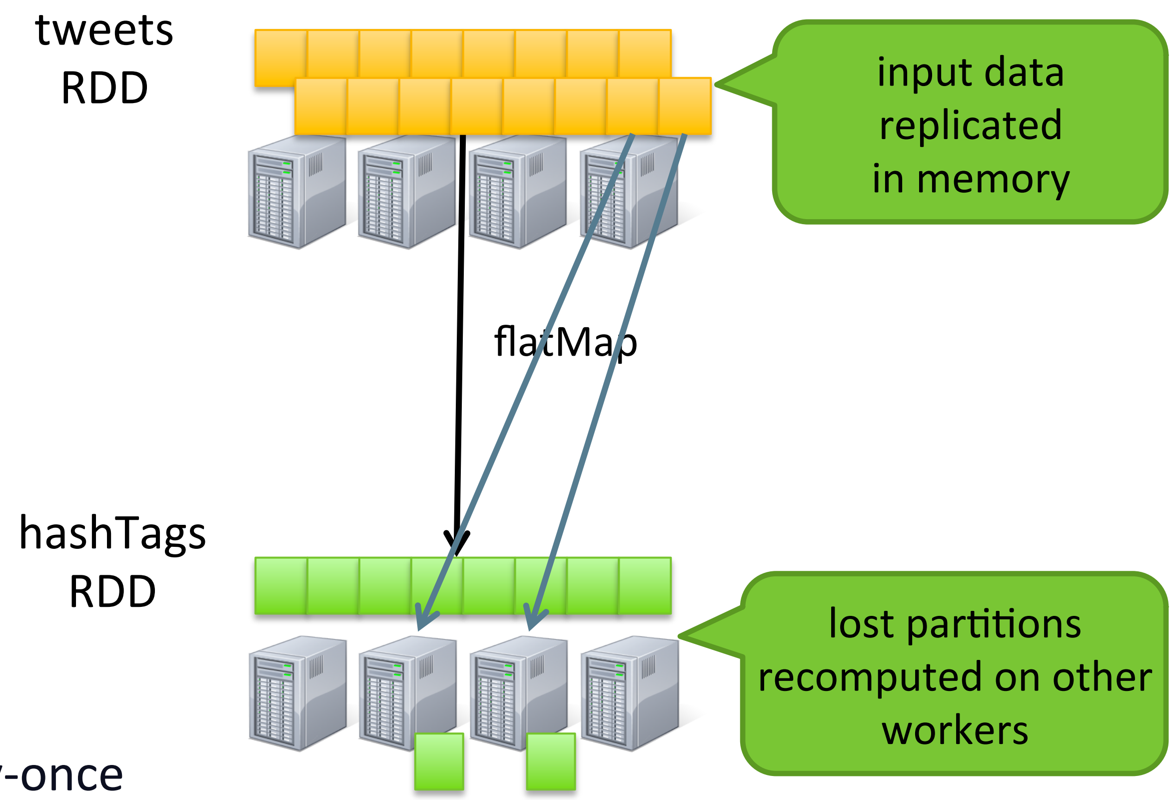
Hortonworks®

Fault-tolerance: Worker

- RDDs remember the operations that created them

- Batches of input data are replicated in memory for fault-tolerance

- Data lost due to worker failure, can be recomputed from replicated input data



- All transformed data is fault-tolerant, and exactly-once transformations

Fault-tolerance: Master

- Master saves the state of the DStreams to a checkpoint file
  - Checkpoint file saved to HDFS periodically

- If master fails, it can be restarted using the checkpoint file

Unifying Batch and Stream Processing Models

Spark program on Twitter log file using RDDs

```
val tweets = sc.hadoopFile("hdfs://...")
val hashTags = tweets.flatMap (status => getTags(status))
hashTags.saveAsHadoopFile("hdfs://...")
```

Spark Streaming program on Twitter stream using DStreams

```
val tweets = ssc.twitterStream()
val hashTags = tweets.flatMap (status => getTags(status))
hashTags.saveAsHadoopFiles("hdfs://...")
```

Demo – Parte 2

Apoio

Apoio

✓ Download Hortonworks Sandbox

http://hortonworks.com/sandbox

✓ Update Zeppelin Gallery

https://github.com/hortonworks-gallery/zeppelin-notebooks

Hortonworks®

Apoio

- ✓ Acessar Zeppelin

http://localhost:9995

- ✓ Acesso Shell

http://localhost:4200

- ✓ Outros Exemplos

https://github.com/DhruvKumar/spark-twitter-sentiment

https://github.com/DhruvKumar/spark-workshop

Hortonworks®

Apoio

- ✓ Twitter Apps

https://apps.twitter.com/

Perguntas?

Obrigado