Apache Spark Release 1.6

Patrick Wendell

About Me @pwendell

U.C. Berkeley PhD, left to co-found Databricks

Coordinate community roadmap

Frequent release manager for Spark

About Databricks

Founded by Spark team, donated Spark to Apache in 2013 and lead development today.

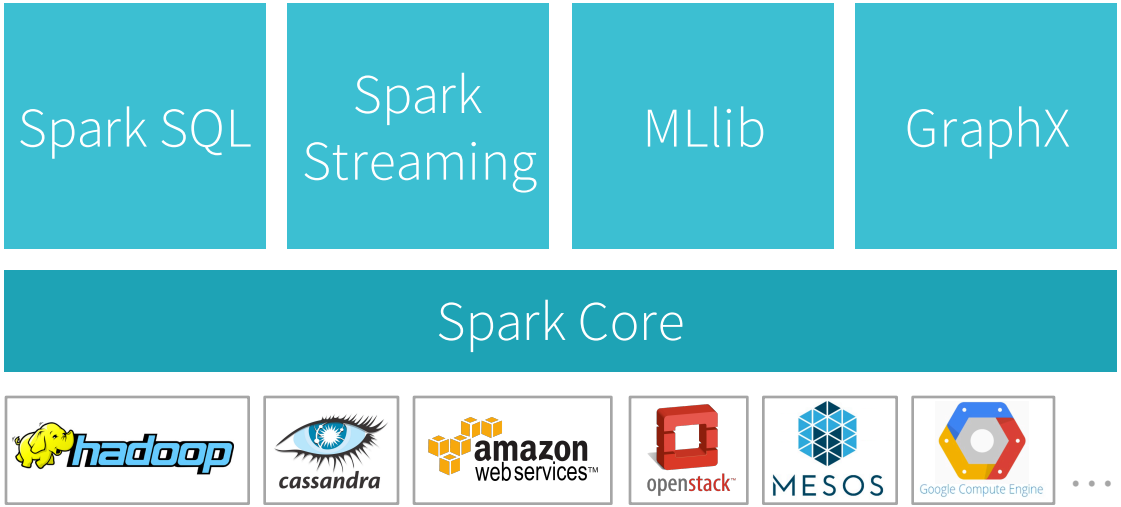
Collaborative, cloud-hosted data platform powered by Spark

Free trial to check it out

https://databricks.com/

We're hiring!

Apache Spark Engine



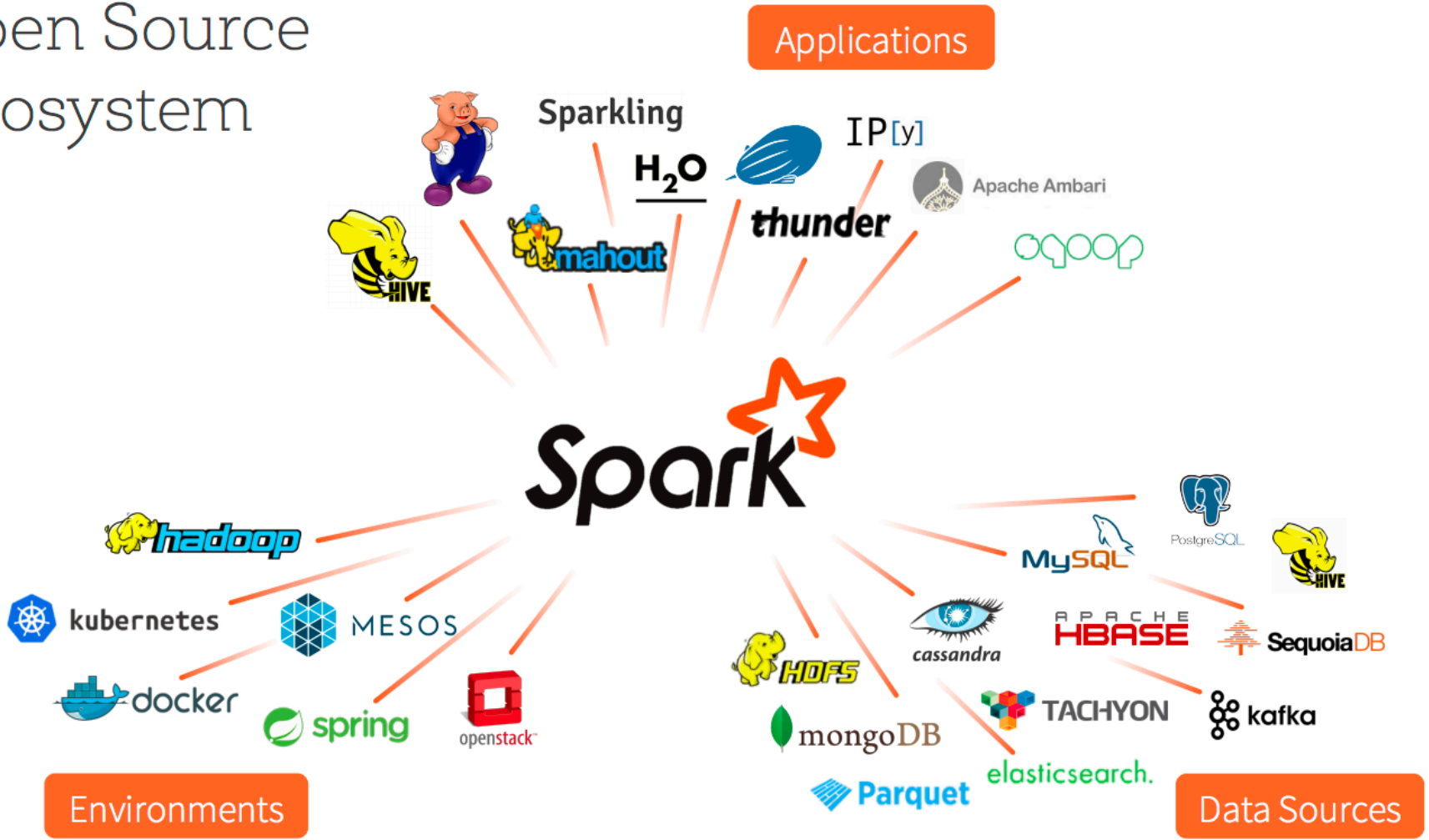
Scale out, fault tolerant

Python, Java, Scala, and R APIs

Standard libraries

Unified engine across diverse workloads & environments

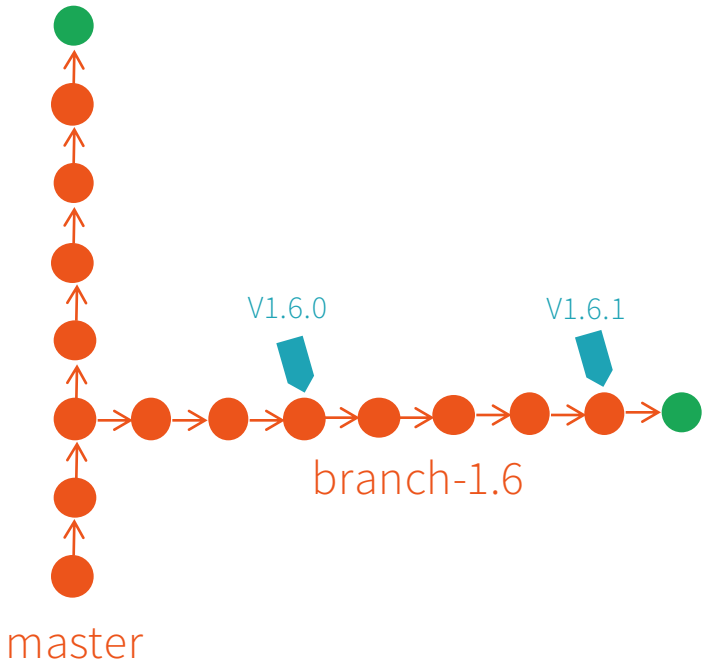
Open Source Ecosystem



Users

Distributors & Apps

Spark's 3 Month Release Cycle



For production jobs, use the latest release

To try out unreleased features or fixes, use nightly builds

people.apache.org/~pwendell/spark-nightly/

Spark 1.6

databricks™

Spark 1.6 Release

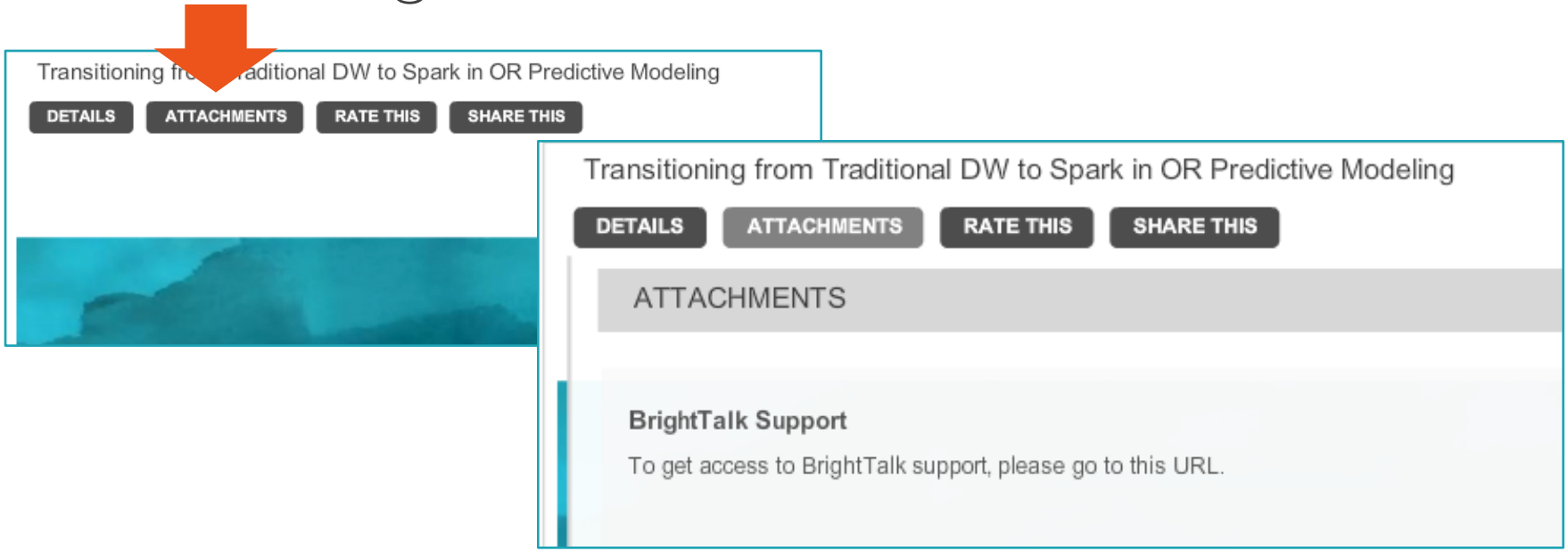
Will ship upstream through Apache foundation in December (likely)

Key themes

- Out of the box performance
- Previews of key new API's

Follow along with me at http://bit.ly/1OBkjMM

Follow along



http://bit.ly/1lrvdLc

Memory Management in Spark: <= 1.5

- Two separate memory managers:
  - Execution memory: computation of shuffles, joins, sorts, aggregations
  - Storage memory: caching and propagating internal data sources across cluster
- Challenges with this:
  - Manual intervention to avoid unnecessary spilling
  - No good defaults for all workloads – meaning lost efficiency
- Goal: Allow memory regions to shrink/grow dynamically

Unified Memory Management in Spark 1.6

- Can cross between execution and storage memory
  - When execution memory exceeds its own region, it can borrow as much of the storage space as is free and vice versa
  - Borrowed storage memory can be evicted at any time
- Significantly reduces configuration
  - Can define low water mark for storage (below which we won't evict)
- Reference: [SPARK-10000]

History of Spark API's

RDD API (2011)

Distribute collection of JVM objects

Functional operators (map, filter, etc)

DataFrame API (2013)

Distribute collection of Row objects

Expression-based operations and UDF's

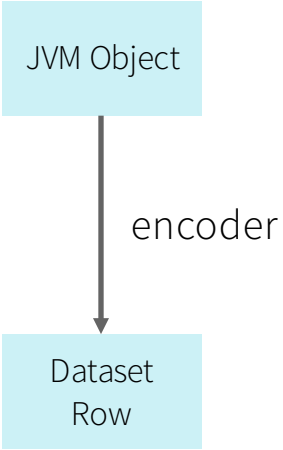
Logical plans and optimizer

Fast/efficient internal representations

Dataset

"Encoder" converts from JVM Object into a Dataset Row

Checkout [SPARK-9999]



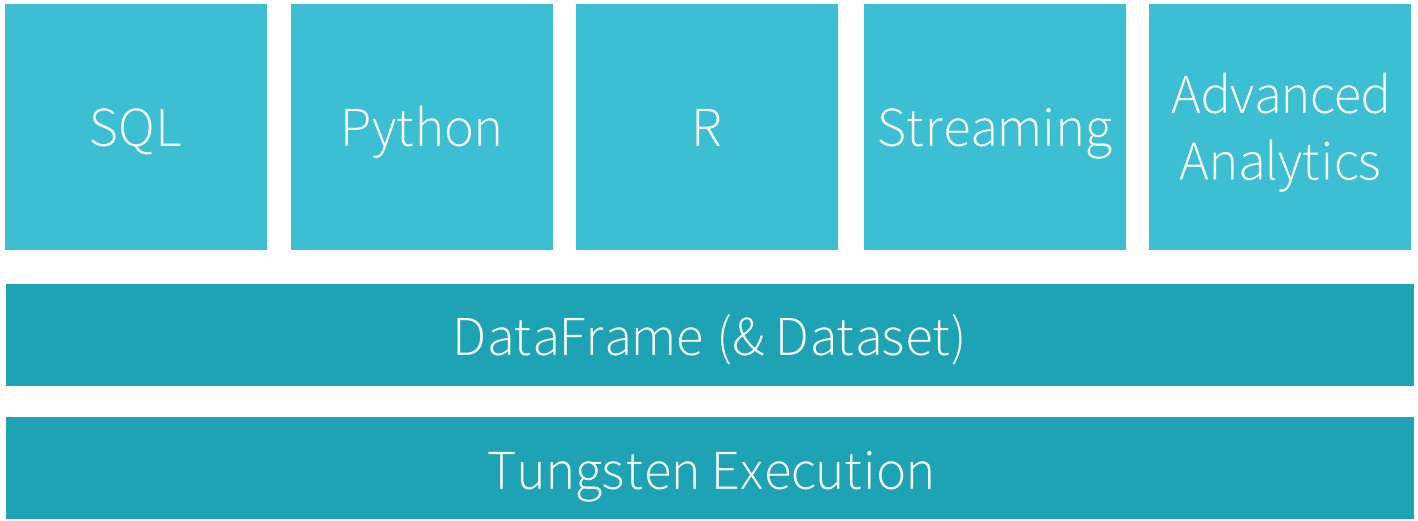
Dataset API in Spark 1.6

Typed interface over DataFrames / Tungsten

```
case class Person(name: String, age: Long)

val dataframe = read.json("people.json")
val ds: Dataset[Person] = dataframe.as[Person]

ds.filter(p => p.name.startsWith("M"))
  .groupBy($"name")
  .avg("age")
```


SQL
Python
R
Streaming
Advanced Analytics
DataFrame (& Dataset)
Tungsten Execution

Other Notable Core Engine Features

SQL directly over files

Advanced JSON parsing

Better instrumentation for SQL operators

Demos of What We Learned So Far

Advanced Layout of Cached Data

- Storing partitioning and ordering schemes in In-memory table scan
  - allows for performance improvements: e.g. in Joins, an extra partition step can be saved based on this information
- Adding `distributeBy` and `localSort` to DF API
  - Similar to HiveQL's `DISTRIBUTE BY`
  - allows the user to control the partitioning and ordering of a data set
- Check out [SPARK-4849]

[Streaming] New improved state management

Introducing a `DStream` transformation for stateful stream processing

- Does not scan every key
- Easier to implement common use cases
  - timeout of idle data
  - returning items other than state

Supercedes `updateStateByKey` in functionality and performance.

`trackStateByKey` (note, this name may change)

[Streaming] trackStateByKey example
(name may change)

```
// Initial RDD input
val initialRDD = ssc.sparkContext.parallelize(...)

// ReceiverInputDStream
val lines = ssc.socketTextStream(...)
val words = lines.flatMap(...)
val wordDStream = words.map(x => (x, 1))

// stateDStream using trackStateByKey
val trackStateFunc = (...) { ... }
val stateDStream =
   wordDStream.trackStateByKey(StateSpec.function(trackStateFunc).initialSta
   te(initialRDD))
```

[Streaming] Display the failed output op in Streaming

Check out:
[SPARK-10885] PR#8950

Duration	Status	Job Id	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total	Error
-	Succeeded	-	-	-	-	-

-	Failed due to error: java.lang.RuntimeException: xxx +details
	`java.lang.RuntimeException: xxx`
	`at StreamingApp$$anonfun$main$2.apply(StreamingApp.scala:31)`
	`at StreamingApp$$anonfun$main$2.apply(StreamingApp.scala:29)`
	`at org.apache.spark.streaming.dstream.DStream$$anonfun$foreachRDD$1$$anonfun$apply$mcV$sp$3.apply(DStream.scala:631)`
		-	-	-	-	-

s 17 ms	Succeeded	81	13 ms	2/2	9/9	
		82	4 ms	1/1 (1 skipped)	5/5 (4 skipped)	
s 8 ms	Failed due to Spark job error +details	83	4 ms	1/1 (1 skipped)	5/5 (4 skipped)	
		84	4 ms	0/1 (1 failed) (1 skipped)	0/5 (1 failed) (4	Job aborted due to stage failure: Task 2 in stage 168.0 failed 1 times, most recent failure: Lost task 2.0 in stage 168.0 (TID 517, localhost): java.lang.RuntimeException: xxx +details

[MLlib]: Pipeline persistence

Persist ML Pipelines to:

- Save models in the spark.ml API
- Re-run workflows in a reproducible manner
- Export models to non-Spark apps (e.g., model server)

This is more complex than ML model persistence because:

- Must persist Transformers and Estimators, not just Models.
- We need a standard way to persist Params.
- Pipelines and other meta-algorithms can contain other Transformers and Estimators, including as Params.
- We should save feature metadata with Models

[MLlib]: Pipeline persistence

Reference [SPARK-6725]

Adding model export/import to the spark.ml API.

Adding the internal Saveable/Loadable API and Parquet-based format

Sub-Tasks

1.	Model export/import for spark.ml: LogisticRegression	RESOLVED	Joseph K. Bradley
2.	Model export/import for spark.ml: HashingTF	CLOSED	*Unassigned*
3.	Model export/import for spark.ml: Normalizer	CLOSED	*Unassigned*
4.	Model export/import for spark.ml: estimators under ml.feature (I)	RESOLVED	Xiangrui Meng
5.	Model export/import for spark.ml: Tokenizer	CLOSED	*Unassigned*
6.	Model export/import for spark.ml: ALS	RESOLVED	Joseph K. Bradley
7.	Model export/import for spark.ml: LinearRegression	RESOLVED	Wenjian Huang
8.	Model export/import for spark.ml: CrossValidator	RESOLVED	Joseph K. Bradley
9.	JSON serialization of standard params	RESOLVED	Xiangrui Meng
10.	Model import/export for non-meta estimators and transformers	RESOLVED	Xiangrui Meng
11.	Model export/import for spark.ml: Pipeline and PipelineModel	RESOLVED	Joseph K. Bradley
12.	Refactoring of basic ML import/export	RESOLVED	Joseph K. Bradley
13.	Refactoring to create template for Estimator, Model pairs	RESOLVED	Joseph K. Bradley
14.	JSON serialization of Param[Vector]	RESOLVED	Xiangrui Meng
15.	Model export/import for spark.ml: all basic Transformers	RESOLVED	Joseph K. Bradley
16.	Model export/import for spark.ml: estimators under ml.feature (II)	RESOLVED	Yanbo Liang
17.	Renames traits to avoid collision with java.util.* and add use default traits to simplify the impl	RESOLVED	Xiangrui Meng
18.	Cleanups to existing Readers and Writers	RESOLVED	Joseph K. Bradley
19.	Model export/import for spark.ml: AFTSurvivalRegression and IsotonicRegression	RESOLVED	Xusen Yin
20.	Model export/import for spark.ml: LDA	RESOLVED	yuhao yang
21.	Model export/import for spark.ml: k-means & naive Bayes	RESOLVED	Xusen Yin
22.	Model export/import for spark.ml: Multilayer Perceptron	IN PROGRESS	Xusen Yin
23.	Model export/import for spark.ml: DecisionTreeClassifier,Regressor	IN PROGRESS	Joseph K. Bradley
24.	Model export/import for RFormula and RFormulaModel	IN PROGRESS	*Unassigned*
25.	Model export/import for spark.ml: OneVsRest	IN PROGRESS	*Unassigned*
26.	Model export/import for spark.ml: TrainValidationSplit	IN PROGRESS	*Unassigned*
27.	Create user guide section explaining export/import	OPEN	*Unassigned*

R-like statistics for GLMs

```
> # Model summary are returned in a similar format to R's native glm().
  summary(model)
```

▸ (1) Spark Jobs

```
$devianceResiduals
 Min       Max
 -1.307112 1.412532

$coefficients
                   Estimate  Std. Error t value  Pr(>|t|)
(Intercept)        2.251393  0.3697543  6.08889  9.568102e-09
Sepal_Width        0.8035609 0.106339   7.556598 4.187317e-12
Species_versicolor 1.458743  0.1121079  13.01195 0
Species_virginica  1.946817  0.100015   19.46525 0
```

Command took 0.90s

Provide R-like summary statistics for ordinary least squares via normal equation solver

Check out [SPARK-9836]

Performance

SPARK-10000 **Unified Memory Management** - Shared memory for execution and caching instead of exclusive division of the regions.

SPARK-10917, SPARK-11149 **In-memory Columnar Cache Performance** - Significant (up to 14x) speed up when caching data that contains complex types in DataFrames or SQL.

SPARK-11389 **SQL Execution Using Off-Heap Memory** - Support for configuring query execution to occur using off-heap memory to avoid GC overhead

Performance (continued)

SPARK-4849 **Advanced Layout of Cached Data** - storing partitioning and ordering schemes in In-memory table scan, and adding distributeBy and localSort to DF API

SPARK-9858 **Adaptive query execution** - Initial support for automatically selecting the number of reducers for joins and aggregations.

Spark SQL

SPARK-9999 Dataset API

SPARK-11197 SQL Queries on Files

SPARK-11745 Reading non-standard JSON files

SPARK-10412 Per-operator Metrics for SQL Execution

SPARK-11329 Star (*) expansion for StructTypes

SPARK-11111 Fast null-safe joins

SPARK-10978 Datasource API Avoid Double Filter

Spark Streaming

API Updates

- SPARK-2629 New improved state management
- SPARK-11198 Kinesis record deaggregation
- SPARK-10891 Kinesis message handler function
- SPARK-6328 Python Streaming Listener API

UI Improvements

- Made failures visible in the streaming tab, in the timelines, batch list, and batch details page.
- Made output operations visible in the streaming tab as progress bars

MLlib: New algorithms / models

SPARK-8518 **Survival analysis** - Log-linear model for survival analysis

SPARK-9834 **Normal equation for least squares** - Normal equation solver, providing R-like model summary statistics

SPARK-3147 **Online hypothesis testing** - A/B testing in the Spark Streaming framework

SPARK-9930 **New feature transformers** - ChiSqSelector, QuantileDiscretizer, SQL transformer

SPARK-6517 **Bisecting K-Means clustering** - Fast top-down clustering variant of K-Means

MLlib: API Improvements

ML Pipelines

- SPARK-6725 **Pipeline persistence** - Save/load for ML Pipelines, with partial coverage of spark.ml algorithms
- SPARK-5565 **LDA in ML Pipelines** - API for Latent Dirichlet Allocation in ML Pipelines

R API

- SPARK-9836 **R-like statistics for GLMs** - (Partial) R-like stats for ordinary least squares via summary(model)
- SPARK-9681 **Feature interactions in R formula** - Interaction operator ":" in R formula

Python API - Many improvements to Python API to approach feature parity

MLlib: Miscellaneous Improvements

SPARK-7685, SPARK-9642 **Instance weights for GLMs** - Logistic and Linear Regression can take instance weights

SPARK-10384, SPARK-10385 **Univariate and bivariate statistics in DataFrames** - Variance, stddev, correlations, etc.

SPARK-10117 **LIBSVM data source** - LIBSVM as a SQL data source

For More Information

Apache Spark 1.6.0 Release Preview: http://apache-spark-developers-list.1001551.n3.nabble.com/ANNOUNCE-Spark-1-6-0-Release-Preview-td15314.html

Spark 1.6 Preview available in Databricks: https://databricks.com/blog/2015/11/20/announcing-spark-1-6-preview-in-databricks.html

Notebooks

Spark 1.6 Improvements Notebook:

http://cdn2.hubspot.net/hubfs/438089/notebooks/Spark_1.6_Improvements.html?t=1448929686268

Spark 1.6 R Improvements Notebook:

http://cdn2.hubspot.net/hubfs/438089/notebooks/Spark_1.6_R_Improvements.html?t=1448946977231

Join us at Spark Summit East

February 16-18, 2016 | New York City

Thanks!

databricks™