Introduction to Spark 1.4

Improvements and New Features

Shixiong Zhu - Spark Meetup Beijing, June 2015

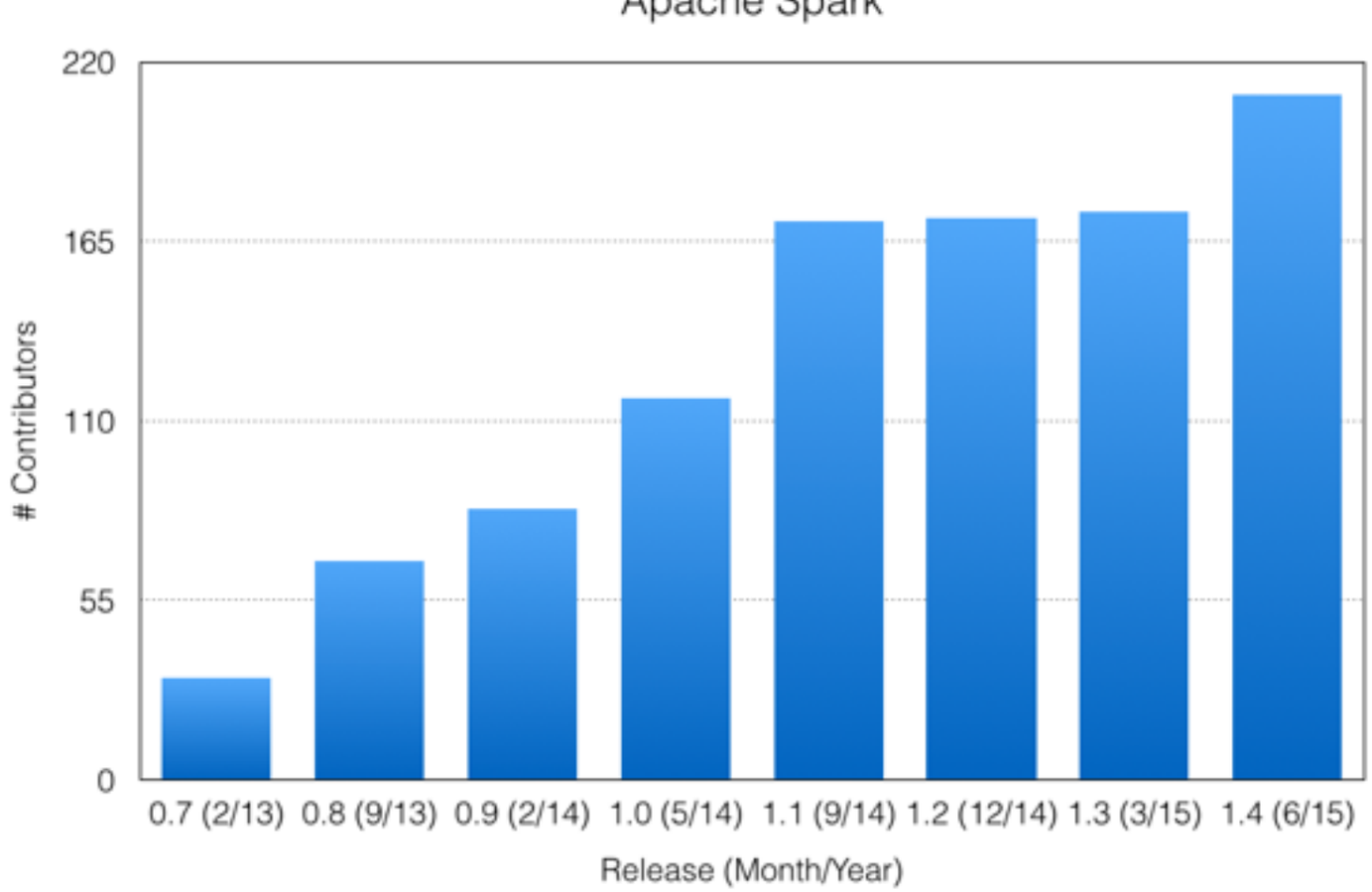
databricks™

Agenda

- Core
- Dataframes and SQL
- Streaming
- MLLib
- SparkR
- UI Improvements (Demo)

Spark 1.4

June 11th, 210+ contributors, 1000+ commits



Core

- Serialized shuffle map output
- Python 3 Support
- Rest APIs for all application information
- Project Tungsten

Project Tungsten Initial Release

Goals Substantially improve the memory and CPU efficiency of Spark applications; Push performance closer to the limits of modern hardware

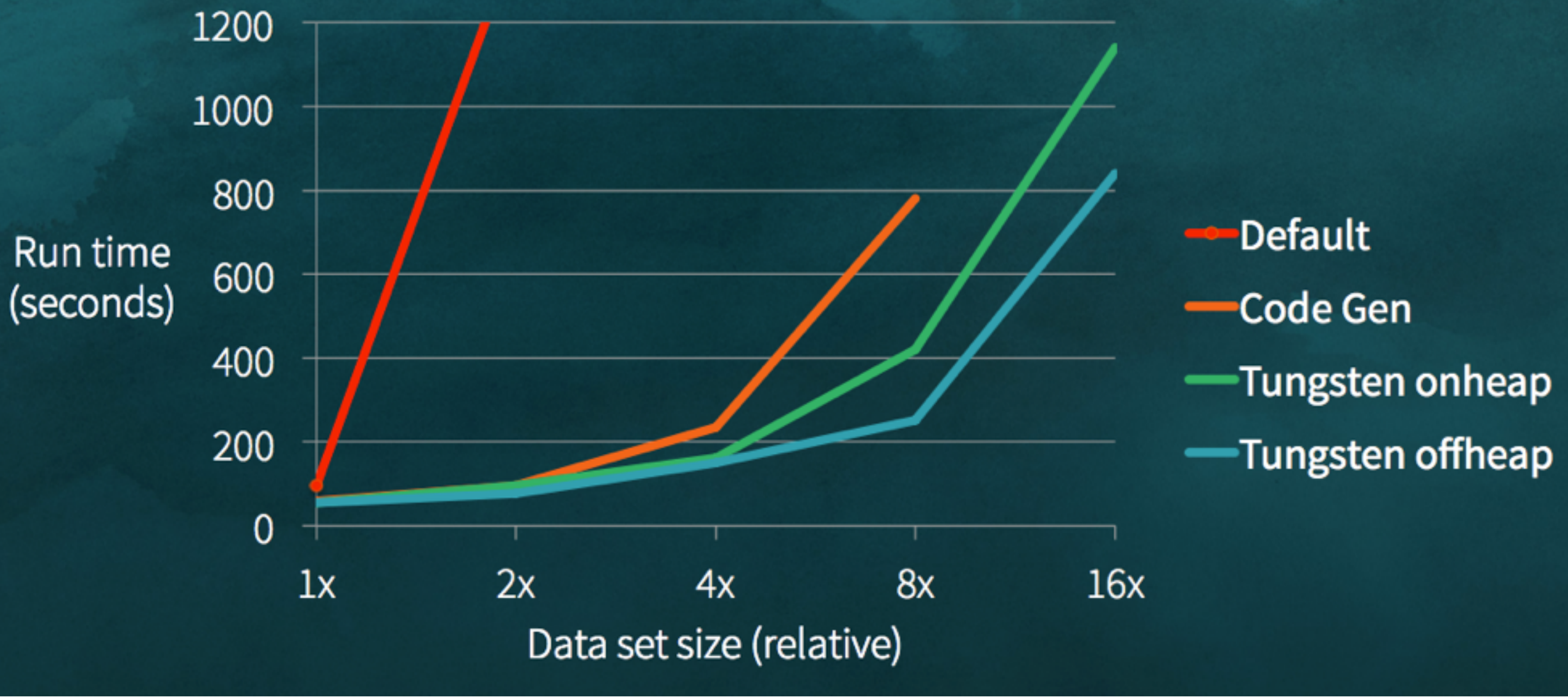
Why is CPU the new bottleneck?

- Hardware has improved
- Spark's IO has been optimized
- Data formats have improved
  - Parquet
- Serialization and hashing are CPU-bound bottlenecks

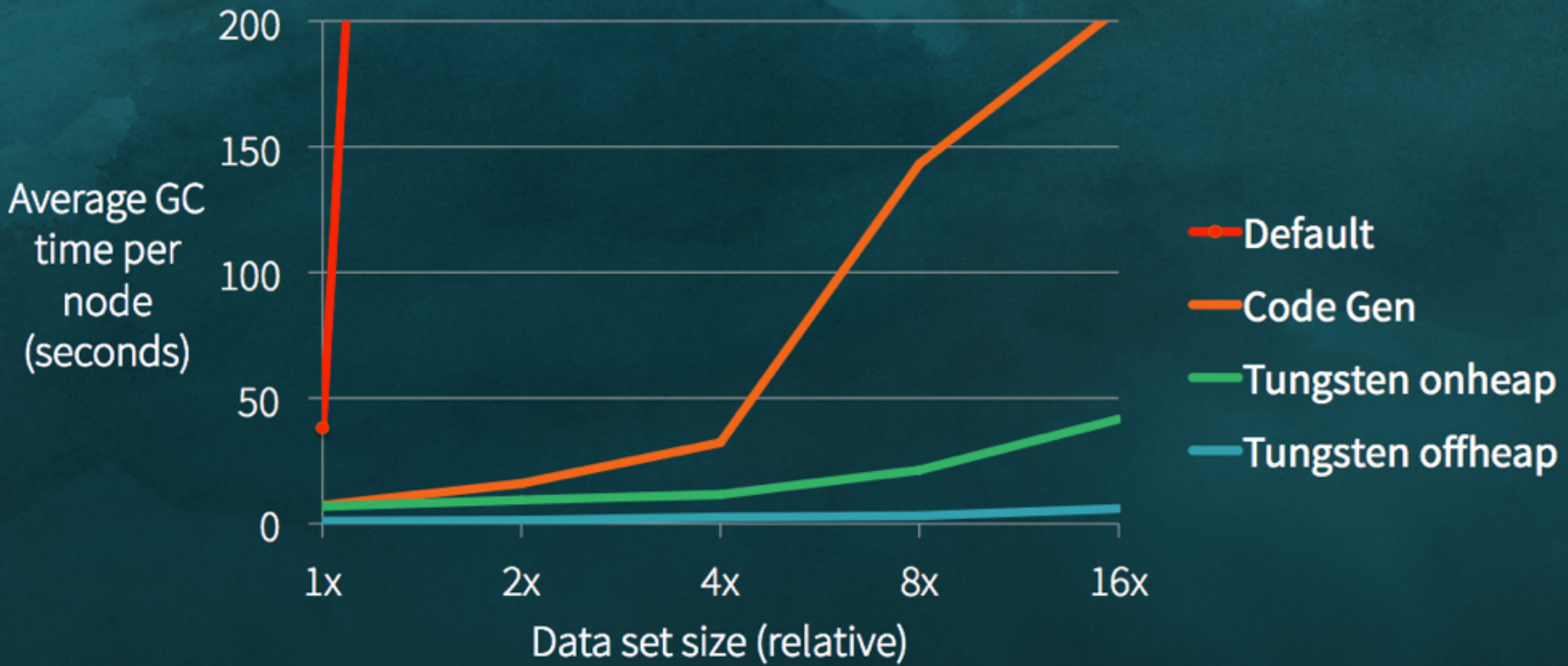
How Tungsten improves CPU & memory efficiency

- Memory Management and Binary Processing: leverage application semantics to manage memory explicitly and eliminate the overhead of JVM object model and garbage collection
- Cache-aware computation: algorithms and data structures to exploit memory hierarchy
- Code generation: exploit modern compilers and CPUs; allow efficient operation directly on binary data

Initial performance results for agg. query



Initial performance results for agg. query



Spark SQL and Dataframes

Datasource API: ORC file data source, partitioning as first-class

Analytic functions: Window functions, statistics and mathematical functions for DataFrame, missing data, rollup/cube

Sort-merge join: MR style sort-merge

Streaming

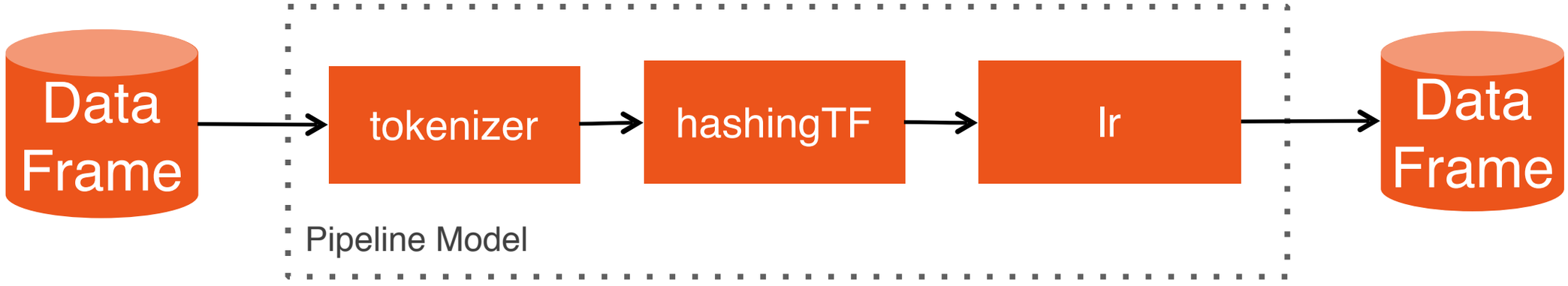
- Enhanced support for Kafka
  - Kafka With Scala 2.11
  - Python API for Kafka Direct mode
- Input Rate Tracking
- Pluggable interface for write ahead logs
- Enhanced support for Kinesis

ML Pipelines Graduates

```
// create pipeline
tok = Tokenizer(in="text", out="words”)
tf = HashingTF(in=“words”, out="features”)
lr = LogisticRegression(maxIter=10, regParam=0.01)
pipeline = Pipeline(stages=[tok, tf, lr])
```

```
// train pipeline
df = sqlCtx.table(”training”)
model = pipeline.fit(df)

// make predictions
df = sqlCtx.read.json("/path/to/test”)
model.transform(df)
  .select(“id”, “text”, “prediction”)
```



MLLib

- Feature transformers
  - VectorAssembler, String/VectorIndexer, OneHotEncoder, PolynomialExpansion, ….
- New algorithms
  - GLM with elastic-net, Tree classifiers, Tree regressors, OneVsRest, …
- Many Python APIs

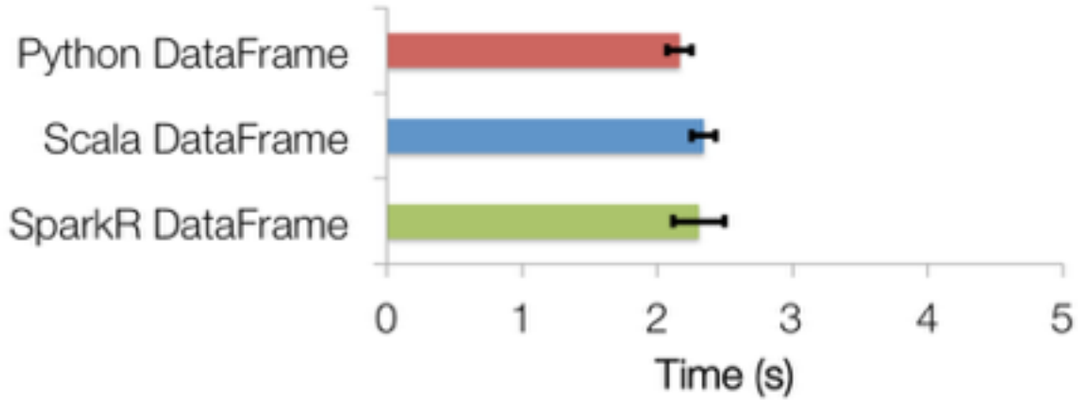
SparkR based on DataFrame

Data Sources API

From “local” R data frames, or from any Spark data source such as Hive, HDFS, Parquet or JSON

Spark’s performance

Runtime optimizer, code generation, memory management



Spark’s scale

Thousands of machines and cores

SparkR Example

```
people <- read.df(sqlContext, "./examples/src/main/resources/people.json", "json")
head(people)
##  age    name
##1  NA Michael
##2  30    Andy
##3  19  Justin

# SparkR automatically infers the schema from the JSON file
printSchema(people)
# root
#  |-- age: integer (nullable = true)
#  |-- name: string (nullable = true)
```

Deployment

Spark on YARN

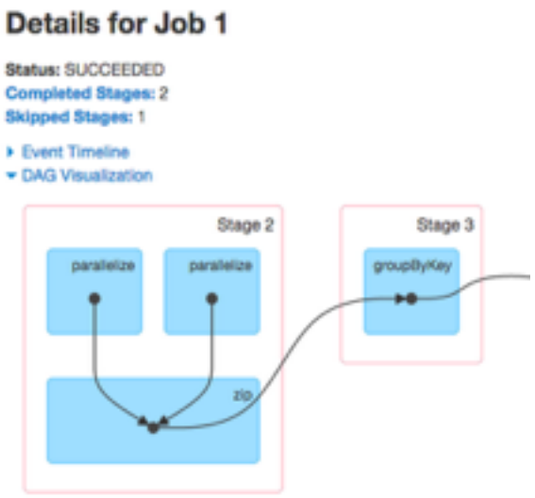
- YARN supported on Spark EC2
- Kerberos for long running (e.g. Streaming) apps

Spark on Mesos

- Cluster mode on Mesos
- Docker image support

Spark UI

RDD DAG Viz.



Job + Stage Timeline

Demo 1: WordCount

```
val lines = sc.textFile("README.md")
val words = lines.flatMap(_.split(" "))
val wordCounts = words.map(x => (x, 1)).reduceByKey(_ + _)
wordCounts.collect().foreach(println)
```

Demo2: ALS

```
bin/run-example mllib.MovieLensALS --rank 5 --
numIterations 5 --lambda 1.0 data/mllib/
sample_movielens_data.txt
```

Streaming UI

Demo 3:StatefulNetworkWordCount

```
Server:

$ nc -lk 9999

Application:

$ bin/run-example streaming.StatefulNetworkWordCount
localhost 9999
```

Thrift Server UI

JDBC/ODBC Server

Started at: 2015/06/26 22:27:26
Time since start: 2 minutes 21 seconds

1 session(s) are online, running 0 SQL statement(s)

Session Statistics

User	IP	Session ID	Start Time	Finish Time	Duration	Total Execute
anonymous	/127.0.0.1	b0b2bc4e-058c-4a80-a182-41f0c3bfbbe7	2015/06/26 22:27:55		1 minute 52 seconds	3

SQL Statistics

User	JobID	GroupID	Start Time	Finish Time	Duration	Statement	State	Detail
anonymous		90fca285-3307-4e2d-ae9d-7ab63366c8c2	2015/06/26 22:28:25	2015/06/26 22:28:26	1 second 355 ms	CREATE TABLE IF NOT EXISTS src (key INT, value STRING)	FINISHED	== Parsed Logical Plan == + details
anonymous		44384501-dd62-4c38-be20-b281c9165bec	2015/06/26 22:28:26	2015/06/26 22:28:27	226 ms	LOAD DATA LOCAL INPATH 'examples/src/main/resources/kv1.txt' INTO TABLE src	FINISHED	== Parsed Logical Plan == + details
anonymous	[0]	8456f7e2-0a7b-4a4f-9eec-0b3d06bd207e	2015/06/26 22:28:27	2015/06/26 22:28:27	818 ms	SELECT count(*) FROM src WHERE key > 200	FINISHED	== Parsed Logical Plan == + details

Demo 4: Thrift Server

```
Start Thrift Server:

sbin/start-thriftserver.sh

Run SQLs in beeline:

bin/beeline

beeline> !connect jdbc:hive2://localhost:10000
beeline> CREATE TABLE IF NOT EXISTS src (key INT, value STRING);
beeline> LOAD DATA LOCAL INPATH 'examples/src/main/resources/
kv1.txt' INTO TABLE src;
beeline> SELECT count(*) FROM src WHERE key > 200;
```

What's Coming in Spark 1.5+?

Project Tungsten Code generation, sort and aggregation

Spark Streaming Flow control, optimized state management

ML Calling single machine solvers, scalability to many features

SparkR Integration with Spark's machine learning API's

More Information About Spark 1.4

Databrick's blog
https://databricks.com/blog/2015/06/11/announcing-apache-spark-1-4.html
https://databricks.com/blog/2015/06/22/understanding-your-spark-application-through-visualization.html
https://databricks.com/blog/2015/06/09/announcing-sparkr-r-on-spark.html
https://databricks.com/blog/2015/04/28/project-tungsten-bringing-spark-closer-to-bare-metal.html

Release note:
http://spark.apache.org/releases/spark-release-1-4-0.html

Thomas Dinsmore's notes:
http://thomaswdinsmore.com/2015/06/12/spark-1-4-released/

Thanks!

databricks™