Recent Developments in SparkR for Advanced Analytics

Xiangrui Meng
meng@databricks.com

2016/06/07 - Spark Summit 2016

About Me

- Software Engineer at Databricks
  - tech lead of machine learning and data science
- Committer and PMC member of Apache Spark
- Ph.D. from Stanford in computational mathematics

Outline

- Introduction to SparkR
- Descriptive analytics in SparkR
- Predictive analytics in SparkR
- Future directions

Introduction to SparkR

Bridging the gap between R and Big Data

SparkR

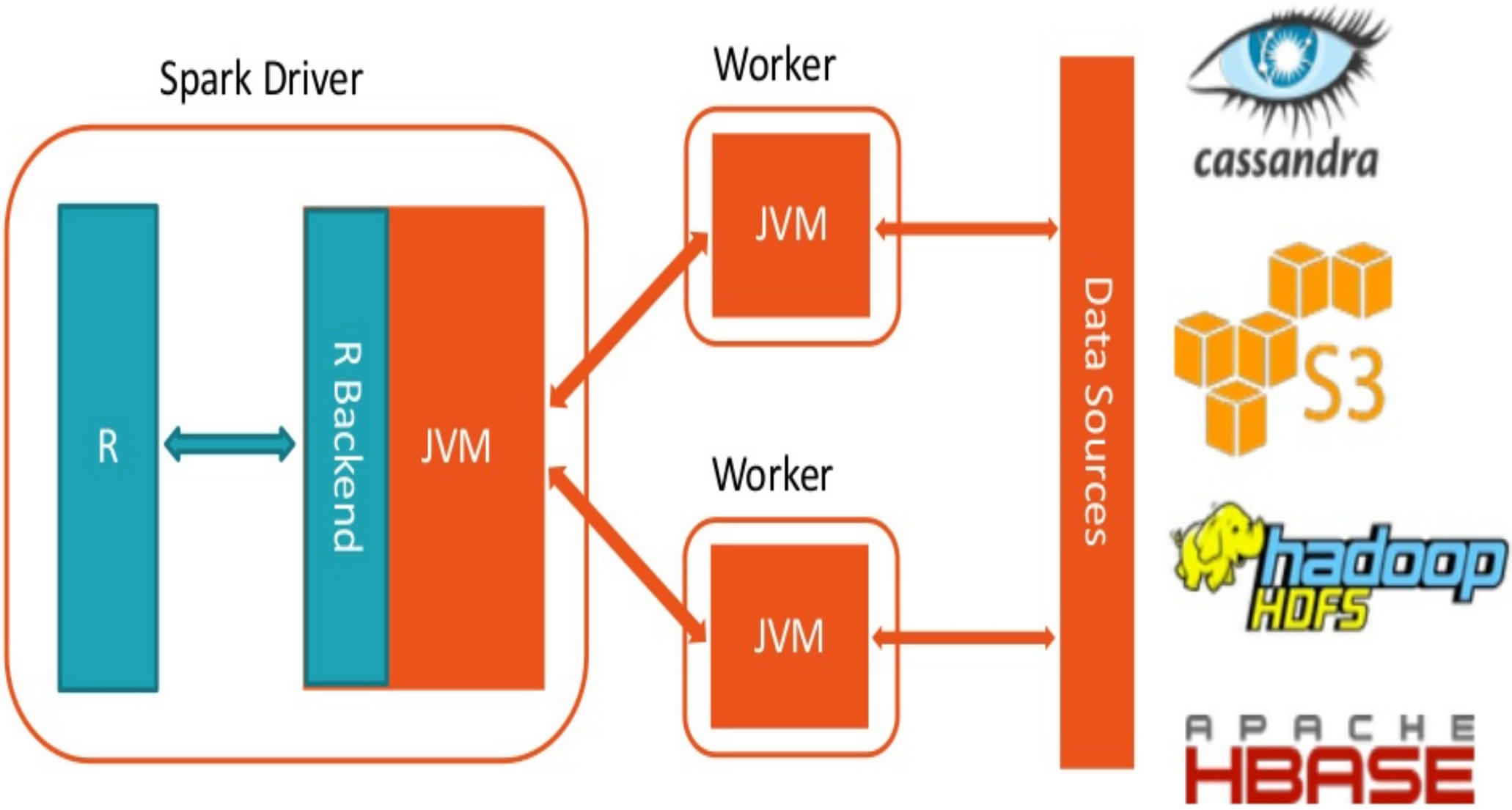
- Introduced to Spark since 1.4
- Wrappers over DataFrames and DataFrame-based APIs

- In SparkR, we make the APIs similar to existing ones in R (or R packages), rather than Python/Java/Scala APIs.
  - R is very convenient for analytics and users love it.
  - Scalability is the main issue, not the API.

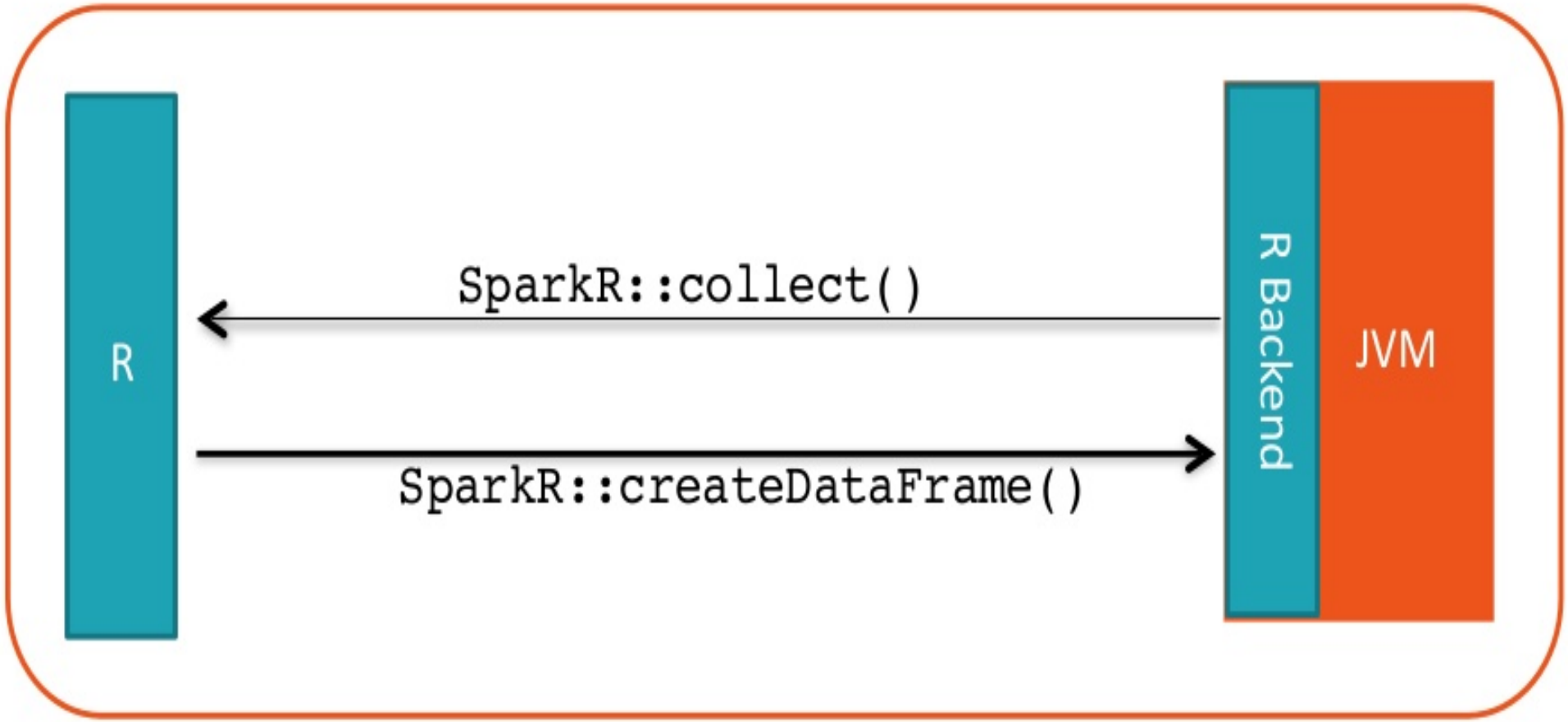
DataFrame-based APIs

- Storage: s3 / HDFS / local / …
- Data sources: csv / parquet / json / …
- DataFrame operations:
  - select / subset / groupBy / agg / collect / …
  - rand / sample / avg / var / …
- Conversion to/from R data.frame

SparkR Architecture



Data Conversion between R and SparkR



Descriptive Analytics

Big Data at a glimpse in SparkR

Summary Statistics

- count, min, max, mean, standard deviation, variance

```
describe(df)
df %>% groupBy("dept", avgAge = avg(df$age))
```

- covariance, correlation

```
df %>% select(var_samp(df$x, df$y))
```

- skewness, kurtosis

```
df %>% select(skewness(df$x), kurtosis(df$x))
```

Sampling Algorithms

- Bernoulli sampling (without replacement)

```
df %>% sample(FALSE, 0.01)
```

- Poisson sampling (with replacement)

```
df %>% sample(TRUE, 0.01)
```

- stratified sampling

```
df %>% sampleBy("key", c(positive = 1.0, negative = 0.1))
```

Approximate Algorithms

- frequent items [Karp03]

```
df %>% freqItems(c("title", "gender"), support = 0.01)
```

- approximate quantiles [Greenwald01]

```
df %>% approxQuantile("value", c(0.1, 0.5, 0.9), relErr = 0.01)
```

- single pass with aggregate pattern
- trade-off between accuracy and space

Implementation: Aggregation Pattern

split + aggregate + combine in a single pass

- split data into multiple partitions
- calculate partially aggregated result on each partition
- combine partial results into final result

Implementation: High-Performance

- new online update formulas of summary statistics
- code generation to achieve high performance

kurtosis of 1 billion values on a Macbook Pro (2 cores):

scipy.stats	250s
octave	120s
CRAN::moments	70s
SparkR / Spark / PySpark	5.5s

Predictive Analytics

Enabling large-scale machine learning in SparkR

MLlib + SparkR

MLlib and SparkR integration started in Spark 1.5.

API design choices:

1. mimic the methods implemented in R or R packages
    - no new method to learn
    - similar but not the same / shadows existing methods
    - inconsistent APIs
2. create a new set of APIs

Generalized Linear Models (GLMs)

- Linear models are simple but extremely popular.
- A GLM is specified by the following:
  - a distribution of the response (from the exponential family),
  - a link function g such that $\mathbf{E}(y) = g^{-1}(x^T\beta)$
- maximizes the sum of log-likelihoods

$$\text{maximize}_\beta \quad \sum_{i=1}^{m} \log p(y_i|x_i;\beta)$$

Distributions and Link Functions

Model	Distribution	Link
linear least squares	normal	identity
logistic regression	binomial	logit
Poisson regression	Poisson	log
gamma regression	gamma	inverse
...	...	...

SparkR supports all families supported by R in Spark 2.0.

GLMs in SparkR

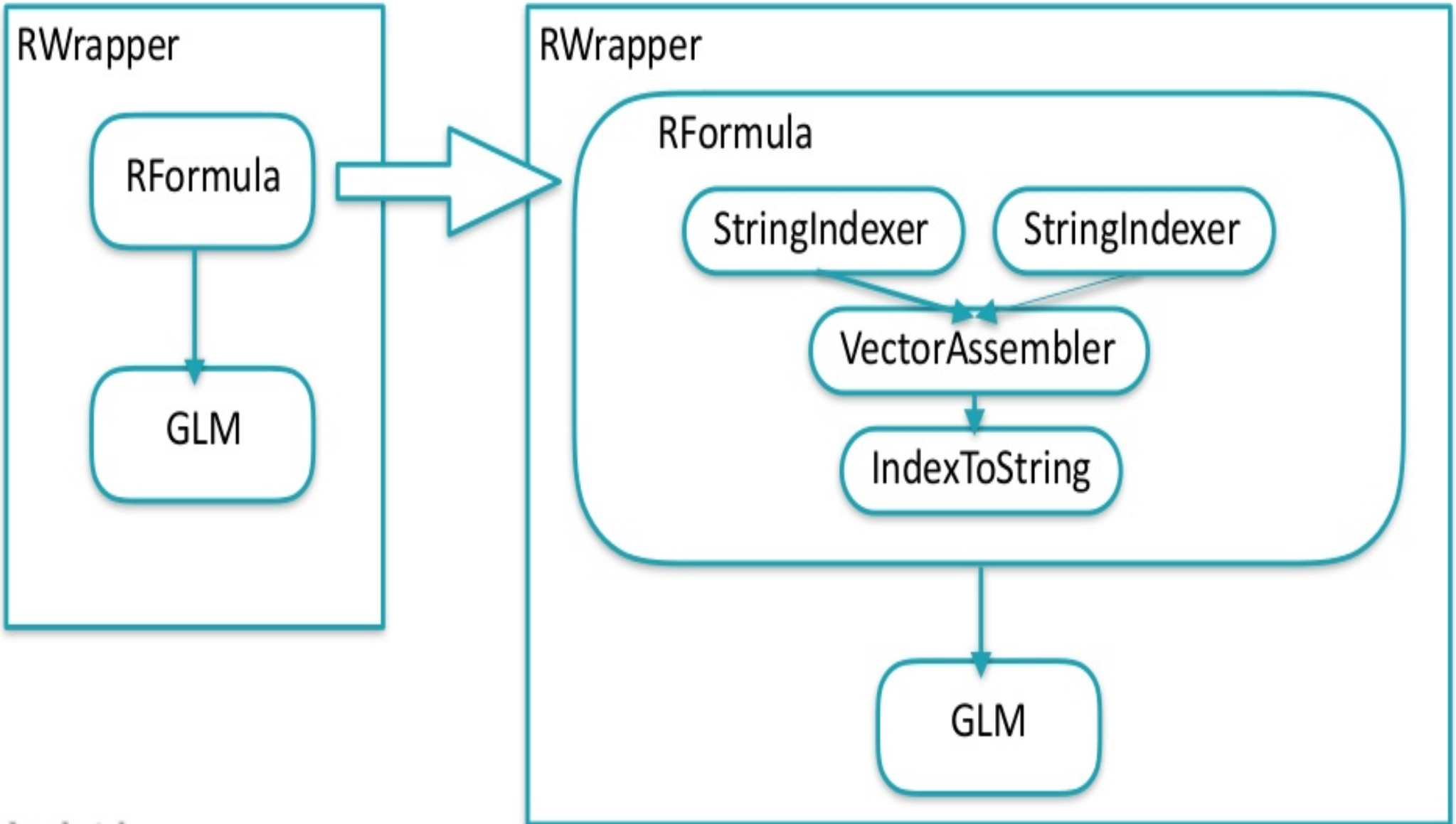
```
# Create the DataFrame for training
df <- read.df(sqlContext, “path/to/training”)

# Fit a Gaussian linear model
model <- glm(y ~ x1 + x2, data = df, family = “gaussian”) # mimic R
model <- spark.glm(df, y ~ x1 + x2, family = “gaussian”)

# Get the model summary
summary(model)

# Make predictions
predict(model, newDF)
```

Implementation: `SparkR::glm`

The `SparkR::glm` is a simple wrapper over an ML pipeline that consists of the following stages:

- RFormula, which itself embeds an ML pipeline for feature preprocessing and encoding,
- an estimator (GeneralizedLinearRegression).

Implementation: SparkR::glm



Implementation: R Formula

- R provides model formula to express models.
- We support the following R formula operators in SparkR:
  - `~` separate target and terms
  - `+` concat terms, "+ 0" means removing intercept
  - `-` remove a term, "- 1" means removing intercept
  - `:` interaction (multiplication for numeric values, or binarized categorical values)
  - `.` all columns except target
- The implementation is in Scala.

Implementation: Test against R

Besides normal tests, we also verify our implementation using R.

```
/*
   df <- as.data.frame(cbind(A, b))
   for (formula in c(b ~ . -1, b ~ .)) {
     model <- lm(formula, data=df, weights=w)
     print(as.vector(coef(model)))
   }

   [1] -3.727121  3.009983
   [1] 18.08  6.08 -0.60
 */
val expected = Seq(Vectors.dense(0.0, -3.727121, 3.009983),
                   Vectors.dense(18.08, 6.08, -0.60))
```

ML Models in SparkR

- generalized linear models (GLMs)
  - `glm / spark.glm (stats::glm)`
- accelerated failure time (AFT) model for survival analysis
  - `spark.survreg (survival)`
- k-means clustering
  - `spark.kmeans (stats:kmeans)`
- Bernoulli naive Bayes
  - `spark.naiveBayes (e1071)`

Model Persistence in SparkR

- model persistence supported for all ML models in SparkR
- thin wrappers over pipeline persistence from MLlib

```
model <- spark.glm(df, x ~ y + z, family = “gaussian”)
write.ml(model, path)
model <- read.ml(path)
summary(model)
```

- feasible to pass saved models to Scala/Java engineers

Work with R Packages in SparkR

- There are ~8500 community packages on CRAN.
  - It is impossible for SparkR to match all existing features.
- Not every dataset is large.
  - Many people work with small/medium datasets.

- SparkR helps in those scenarios by:
  - connecting to different data sources,
  - filtering or downsampling big datasets,
  - parallelizing training/tuning tasks.

Work with R Packages in SparkR

```
df <- sqlContext %>% read.df(…) %>% collect()
points <- data.matrix(df)

run_kmeans <- function(k) {
  kmeans(points, centers=k)
}
kk <- 1:6

lapply(kk, run_kmeans) # R's apply
spark.lapply(sc, kk, run_kmeans) # parallelize the tasks
```

summary(this.talk)

- SparkR enables big data analytics on R
  - descriptive analytics on top of DataFrames
  - predictive analytics from MLlib integration
- SparkR works well with existing R packages

Thanks to the Apache Spark community for developing and maintaining SparkR: Alteryx, Berkeley AMPLab, Databricks, Hortonworks, IBM, Intel, etc, and individual contributors!!

Future Directions

- CRAN release of SparkR
- more consistent APIs with existing R packages: dplyr, etc
- better R formula support
- more algorithms from MLlib: decision trees, ALS, etc
- better integration with existing R packages: gapply / UDFs
- integration with Spark packages: GraphFrames, CoreNLP, etc

We'd greatly appreciate feedback from the R community!

Try Apache Spark with Databricks

- Download a companion notebook of this talk at: http://dbricks.co/1rbujoD
- Try latest version of Apache Spark and preview of Spark 2.0

http://databricks.com/try

Thank you.

- SparkR user guide on Apache Spark website
- MLlib roadmap for Spark 2.1
- Office hours:
  - 2-3:30pm at Expo Hall Theater; 3:45-6pm at Databricks booth
- Databricks Community Edition and blog posts