

SCALABLE DATA SCIENCE WITH SPARKR

Felix Cheung

Principal Engineer - *Spark @ Microsoft* & Apache Spark Committer



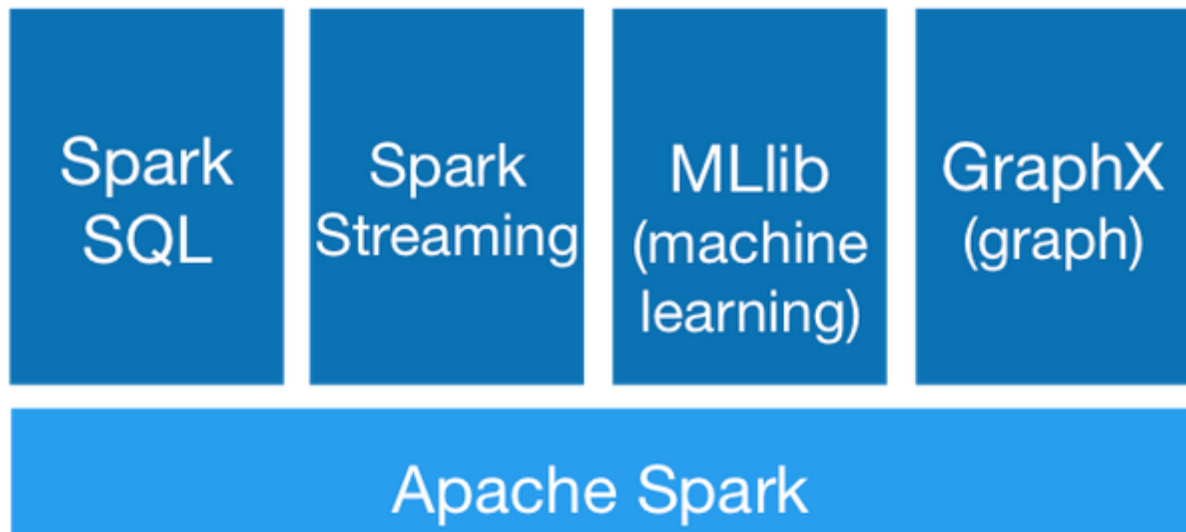


Disclaimer:
Apache Spark community contributions



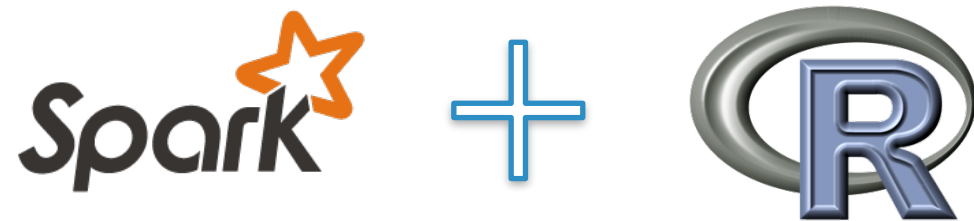
Spark in 5 seconds

- General-purpose cluster computing system
- Spark SQL + DataFrame/Dataset + data sources
- Streaming/Structured Streaming
- ML
- GraphX



R

- A programming language for statistical computing and graphics
- S – 1975
- S4 - advanced object-oriented features
- R – 1993
- S + lexical scoping
- Interpreted
- Matrix arithmetic
- Comprehensive R Archive Network ([CRAN](https://cran.r-project.org/)) – 10k+ packages

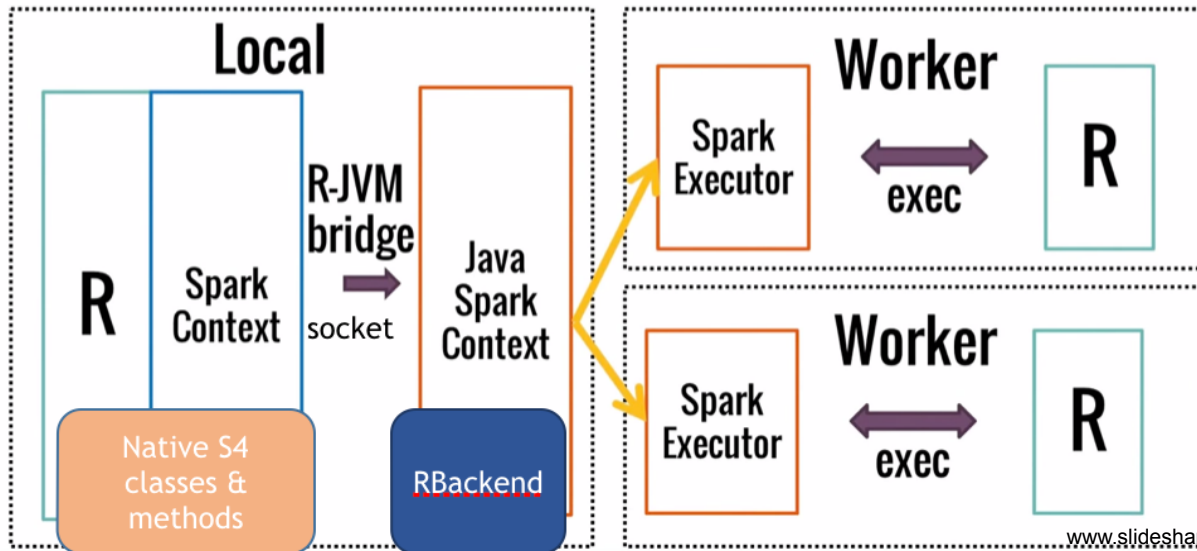


SparkR

- R language APIs for Spark and Spark SQL
- Exposes Spark functionality in an R-friendly DataFrame APIs
- Runs as its own REPL `sparkR`
- or as a R package loaded in IDEs like RStudio
`library(SparkR)`
`sparkR.session()`

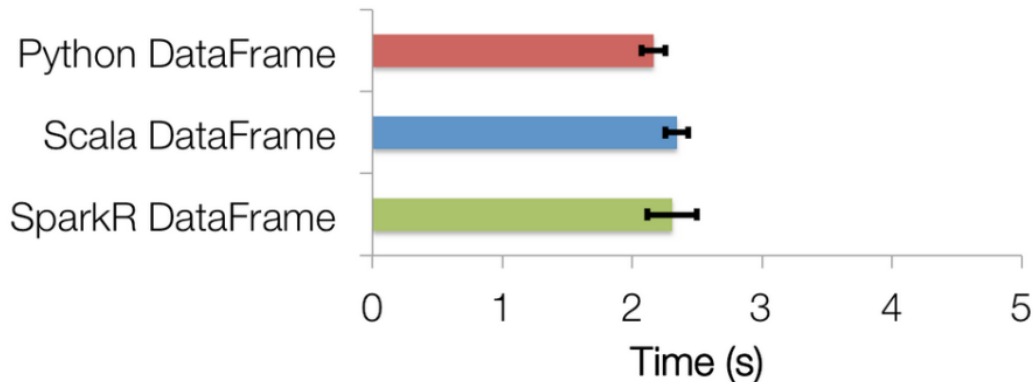
Architecture

- Native R classes and methods
- RBackend
- Scala “helper” methods (ML pipeline etc.)



Advantages

- JVM processing, full access to DAG capabilities and Catalyst optimizer, predicate pushdown, code generation, etc.

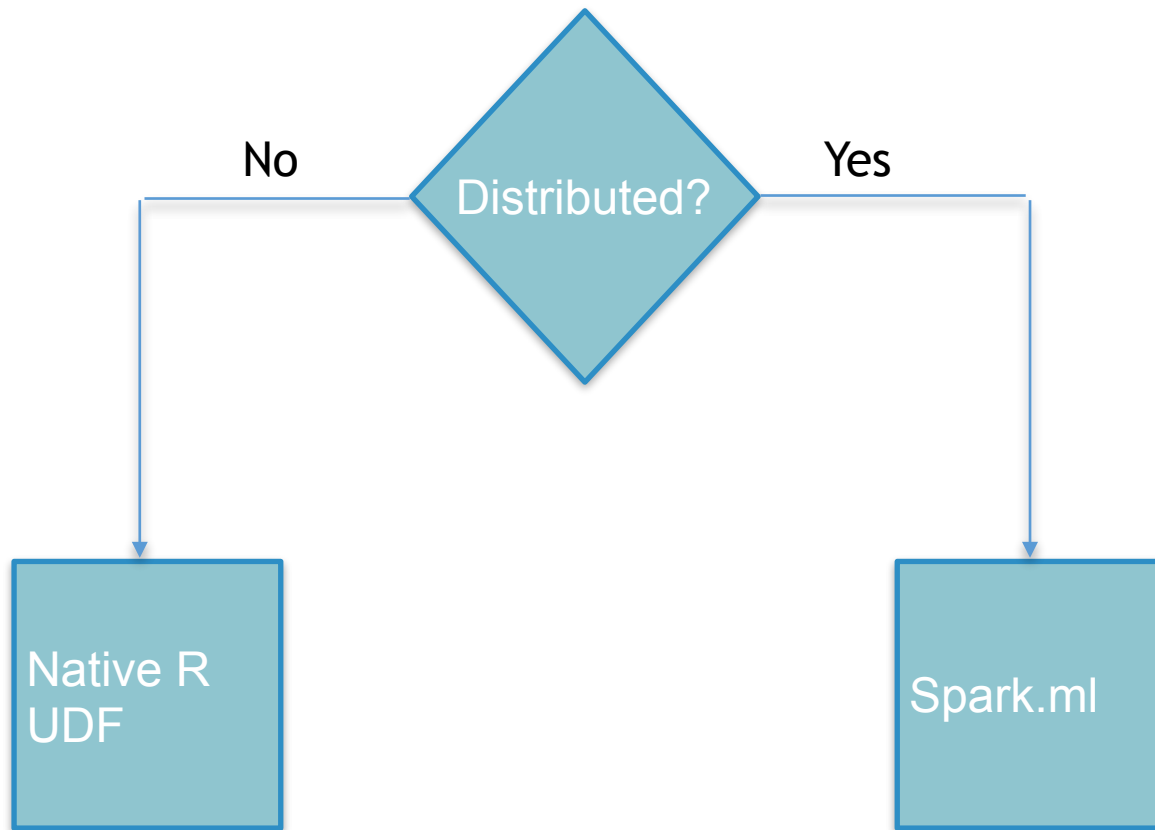


Features - What's new in SparkR

- SQL
- Data source (JSON, csv, PostgreSQL, libsvm)
- SparkSession & default session (streamlined parameter)
`as.DataFrame(iris)`
- Catalog (external data table management)
- [Spark packages](#), `spark.addFiles()`
- ML
- R-native UDF
- Cluster support (YARN, mesos, standalone)

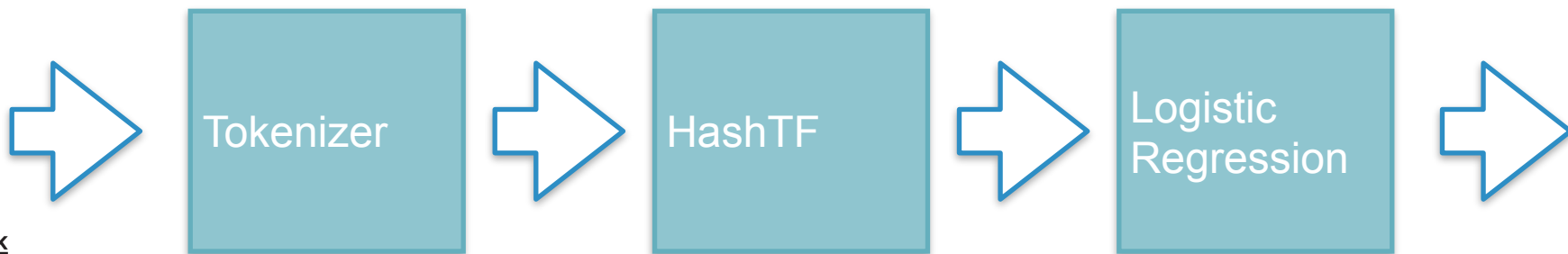
SparkR for Data Science

Decisions, decisions?

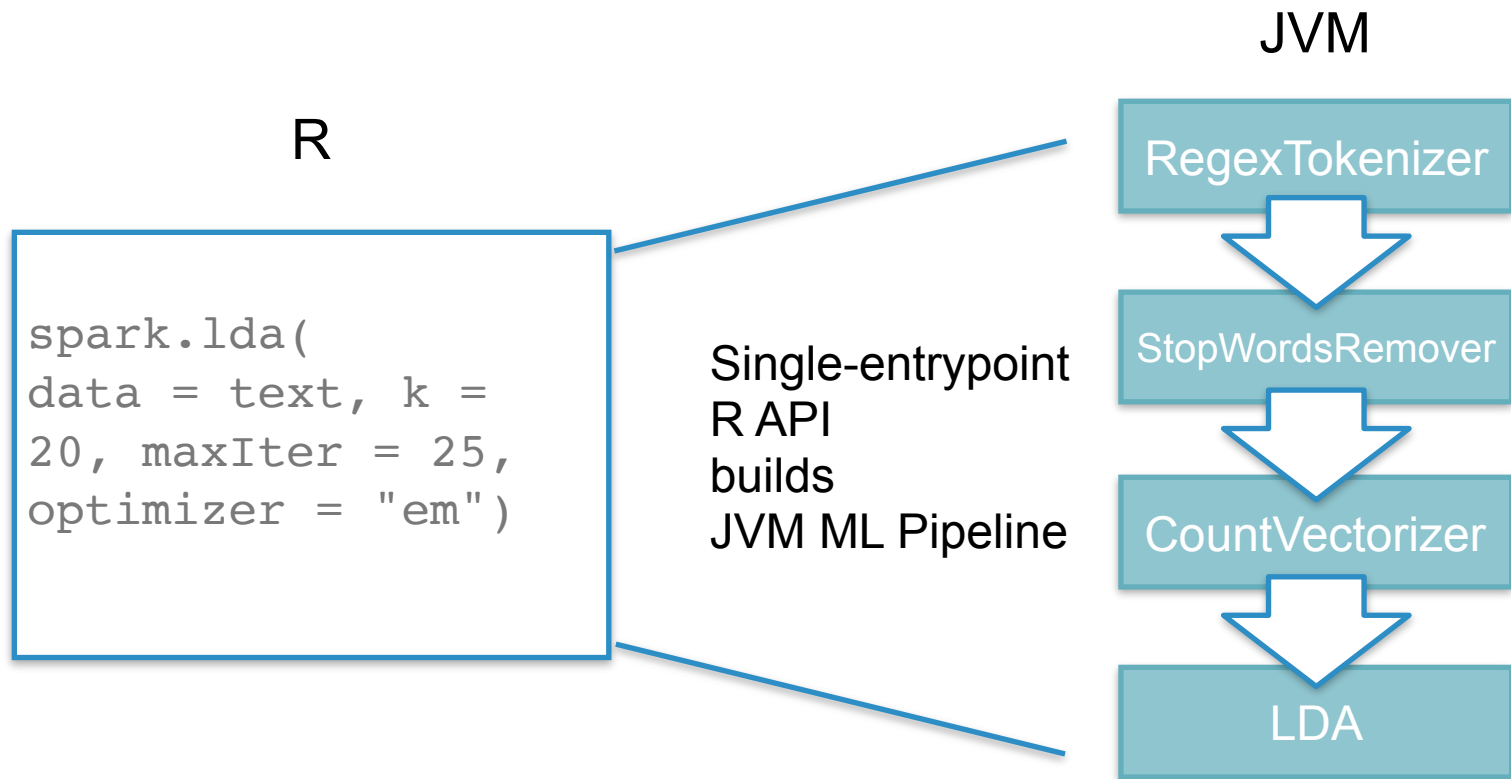


Spark ML Pipeline

- Pre-processing, feature extraction, model fitting, validation stages
- Transformer
- Estimator
- Cross-validation/hyperparameter tuning



SparkR API for ML Pipeline



Model Operations

- `summary` - print a summary of the fitted model
- `predict` - make predictions on new data
- `write.ml/read.ml` - save/load fitted models
(slight layout difference: pipeline model plus R metadata)

Spark.ml in SparkR 2.0.0

- Generalized Linear Model (GLM)
- Naive Bayes Model
- k-means Clustering
- Accelerated Failure Time (AFT) Survival Model

Spark.ml in SparkR 2.1.0

- Generalized Linear Model (GLM)
- Naive Bayes Model
- k-means Clustering
- Accelerated Failure Time (AFT) Survival Model
- Isotonic Regression Model
- Gaussian Mixture Model (GMM)
- Latent Dirichlet Allocation (LDA)
- Alternating Least Squares (ALS)
- Multilayer Perceptron Model (MLP)
- Kolmogorov-Smirnov Test (K-S test)
- Multiclass Logistic Regression
- Random Forest
- Gradient Boosted Tree (GBT)

RFormula

- Specify modeling in symbolic form

$y \sim f0 + f1$

response y is modeled linearly by $f0$ and $f1$

- Support a subset of [R formula](#) operators

\sim , \cdot , $:$, $+$, $-$

- Implemented as feature transformer in core Spark, available to Scala/Java, Python
- String label column is indexed
- String term columns are one-hot encoded

Generalized Linear Model

```
# R-like
```

```
glm(Sepal_Length ~ Sepal_Width + Species,  
    gaussianDF, family = "gaussian")
```

```
spark.glm(binomialDF, Species ~  
Sepal_Length + Sepal_Width, family =  
"binomial")
```

- “binomial” output string label, prediction

Multilayer Perceptron Model

```
spark.mlp(df, label ~ features,  
blockSize = 128, layers = c(4, 5, 4,  
3), solver = "l-bfgs", maxIter = 100,  
tol = 0.5, stepSize = 1)
```

Multiclass Logistic Regression

```
spark.logit(df, label ~ ., regParam =  
0.3, elasticNetParam = 0.8, family =  
"multinomial", thresholds = c(0, 1,  
1) )
```

- binary or multiclass

Random Forest

```
spark.randomForest(df, Employed ~ ., type  
= "regression", maxDepth = 5, maxBins =  
16)
```

```
spark.randomForest(df, Species ~  
Petal_Length + Petal_Width,  
"classification", numTree = 30)
```

- “classification” index label, predicted label to string

Gradient Boosted Tree

```
spark.gbt(df, Employed ~ ., type =  
"regression", maxDepth = 5, maxBins = 16)
```

```
spark.gbt(df, IndexedSpecies ~ ., type =  
"classification", stepSize = 0.1)
```

- “classification” index label, predicted label to string
- Binary classification

Modeling Parameters

```
spark.randomForest
```

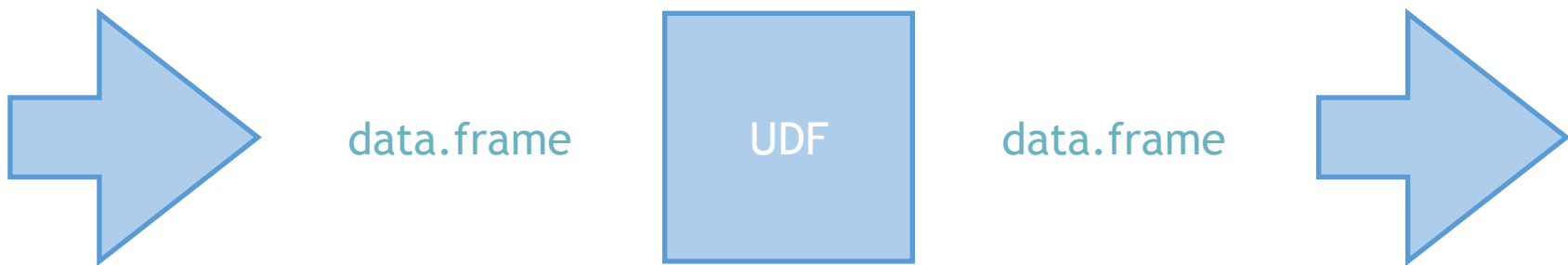
```
function(data, formula, type = c("regression", "classification"),  
        maxDepth = 5, maxBins = 32, numTrees = 20, impurity = NULL,  
        featureSubsetStrategy = "auto", seed = NULL,  
        subsamplingRate = 1.0,  
        minInstancesPerNode = 1, minInfoGain = 0.0,  
        checkpointInterval = 10,  
        maxMemoryInMB = 256, cacheNodeIds = FALSE)
```


Spark.ml Challenges

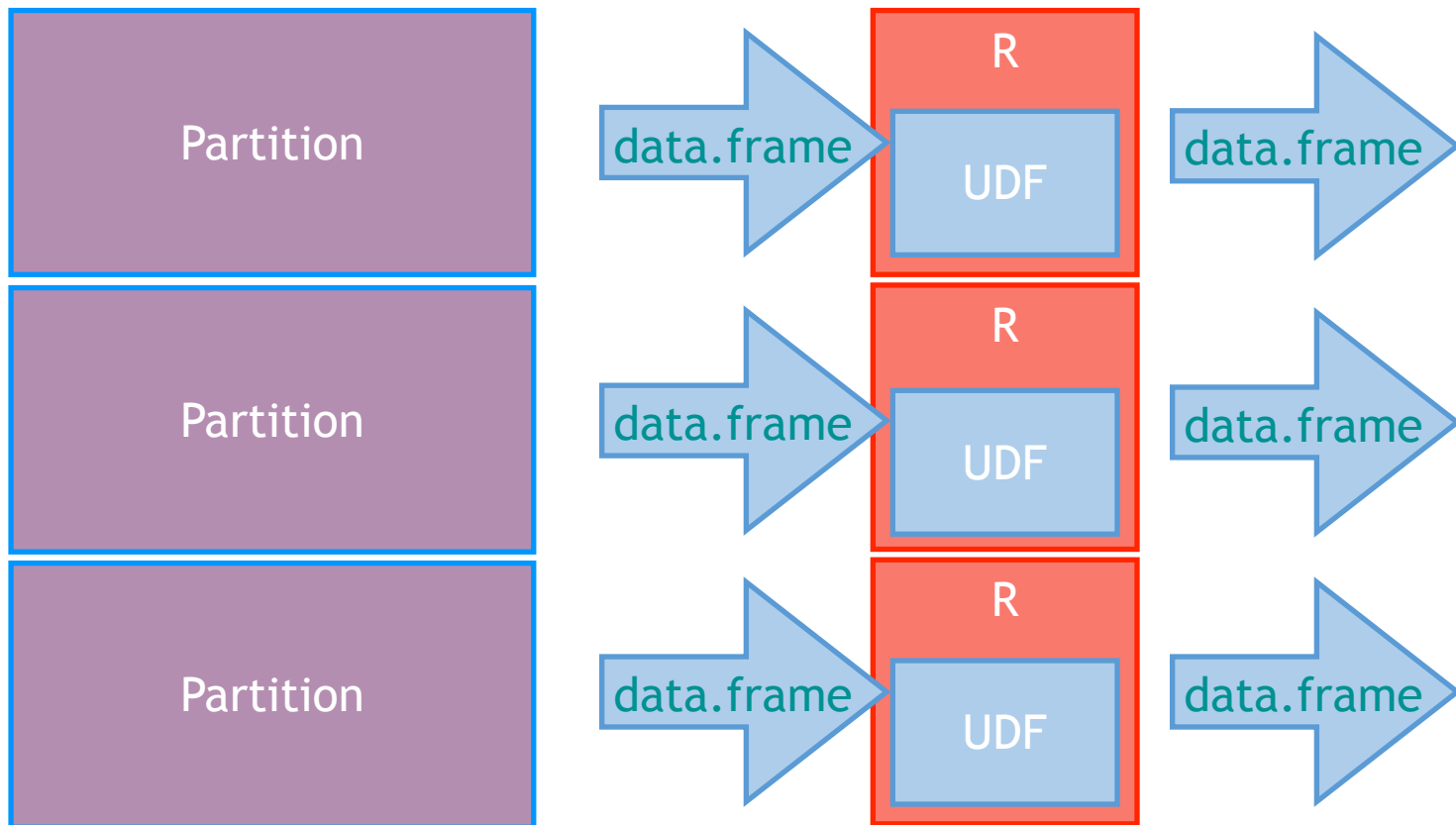
- Limited API sets
 - Non-trivial to map spark.ml API to R API
 - Keeping up to changes
 - Almost all (except One vs Rest)
 - Simple API, but fixed ML pipeline
- Debugging is hard
 - Not a ML specific problem
 - Getting better?

Native-R UDF

- User-Defined Functions - custom transformation
- Apply by Partition
- Apply by Group



Parallel Processing By Partition



UDF: Apply by Partition

- Similar to R `apply`
- Function to process each partition of a DataFrame
- Mapping of Spark/R data types

```
dapply(carsSubDF,  
      function(x) {  
        x <- cbind(x, x$mpg * 1.61)  
      },  
      schema)
```

UDF: Apply by Partition + Collect

- No schema

```
out <- dapplyCollect(  
  carsSubDF,  
  function(x) {  
    x <- cbind(x, "kmpg" = x$mpg*1.61)  
  })
```

Example - UDF

```
results <- dapplyCollect(train,
  function(x) {
    model <-
randomForest::randomForest(as.factor(dep_delayed_
15min) ~ Distance + night + early, data = x,
importance = TRUE, ntree = 20)
    predictions <- predict(model, t)
    data.frame(UniqueCarrier = t$UniqueCarrier,
delayed = predictions)
  })
```

access package
"randomForest:"
at each invocation

closure capture -
serialize &
broadcast "t"

UDF: Apply by Group

- By grouping columns

```
gapply(carsDF, "cyl",  
      function(key, x) {  
        y <- data.frame(key, max(x$mpg))  
      },  
      schema)
```

UDF: Apply by Group + Collect

- No Schema

```
out <- gapplyCollect(carsDF, "cyl",  
  function(key, x) {  
    y <- data.frame(key, max(x$mpg))  
    names(y) <- c("cyl", "max_mpg")  
    y  
  })
```


UDF: data type mapping

* not a complete list

R	Spark
byte	byte
integer	integer
float	float
double, numeric	double
character, string	string
binary, raw	binary
logical	boolean
POSIXct, POSIXlt	timestamp
Date	date
array, list	array
env	map

UDF Challenges

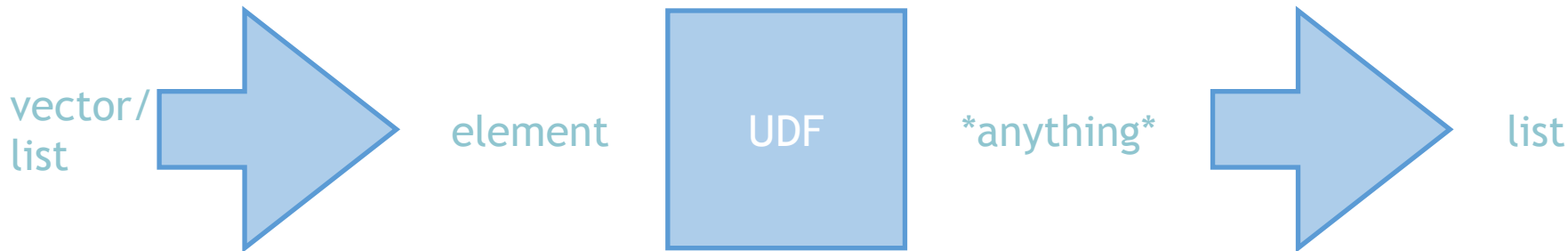
- “struct”
 - No support for nested structures as columns
- Scaling up / data skew
 - What if partition or *group* too big for single R process?
- Not enough data variety to run model?
- Performance costs
 - Serialization/deserialization, data transfer
 - esp. beware of closure capture

UDF: lapply

- Like R `lapply` or `doParallel`
- Good for “embarrassingly parallel” tasks
 - Such as hyperparameter tuning

UDF: lapply

- Take a native R list, distribute it
- Run the UDF in parallel



UDF: parallel distributed processing

- Output is a list - needs to fit in memory at the driver

```
costs <- exp(seq(from = log(1), to = log(1000),  
length.out = 5))  
train <- function(cost) {  
  model <- e1071::svm(Species ~ ., iris, cost =  
cost)  
  summary(model)  
}  
summaries <- spark.lapply(costs, train)
```

Walkthrough

Demo at felixcheung.github.io

One last thing...

SparkR as a Package (target 2.1.1)

- Goal: simple one-line installation of SparkR from CRAN
`install.packages("SparkR")`
- Spark Jar downloaded from official release and cached automatically, or manually `install.spark()` since Spark 2.0.0
- R vignettes
- Community can write packages that depends on SparkR package
- Advanced Spark JVM interop APIs

```
sparkR.newJObject  
sparkR.callJMethod  
sparkR.callJStatic
```

Ecosystem

- RStudio [sparklyr](#)
- [RevoScaleR](#)/RxSpark, R Server
- [H2O R](#)
- Apache SystemML (R-like API)
- Renjin (not Spark)
- IBM BigInsights Big R (not Spark!)

Recap: SparkR 2.0.0, 2.1.0

- `SparkSession`
- ML
- UDF

What's coming in SparkR 2.1.1

- Fix Gamma family with GLM, optimizer in LDA (SPARK-19133, SPARK-19066)
- Partitioning DataFrame (SPARK-18335, SPARK-18788)

```
df <- as.DataFrame(cars, numPartitions = 10)  
getNumPartitions(df)
```
- Setting column R-friendly shortcuts (SPARK-19130, SPARK-18823)

```
df$foo <- 1  
df[[myname]] <- 1; df[[2]] <- df$eruptions / 60
```
- Spark UI URL `sparkR.uiWebUrl` (SPARK-18903)
- `install.spark` better download error handling (SPARK-19231)

What's coming in SparkR 2.2.0

- More, richer ML - Bisecting K-means

More in-planning and *not* committed - feedback appreciated!

- Tweedie GLM
- `collect` performance (SPARK-18924)
- ML Pipeline in SparkR (SPARK-18822)
- Richer RFormula support (SPARK-18570, SPARK-18569)
- Better tree ensemble summary (SPARK-18348)
- ML persistence format (SPARK-15572)

Thank You.

<https://github.com/felixcheung>

linkedin: <http://linkd.in/1OeZDb7>

blog: <http://bit.ly/1E2z6OI>

