An Update on Scaling Data Science with SparkR

Heiko Korndorf, Wireframe

#DSSAIS18

Agenda

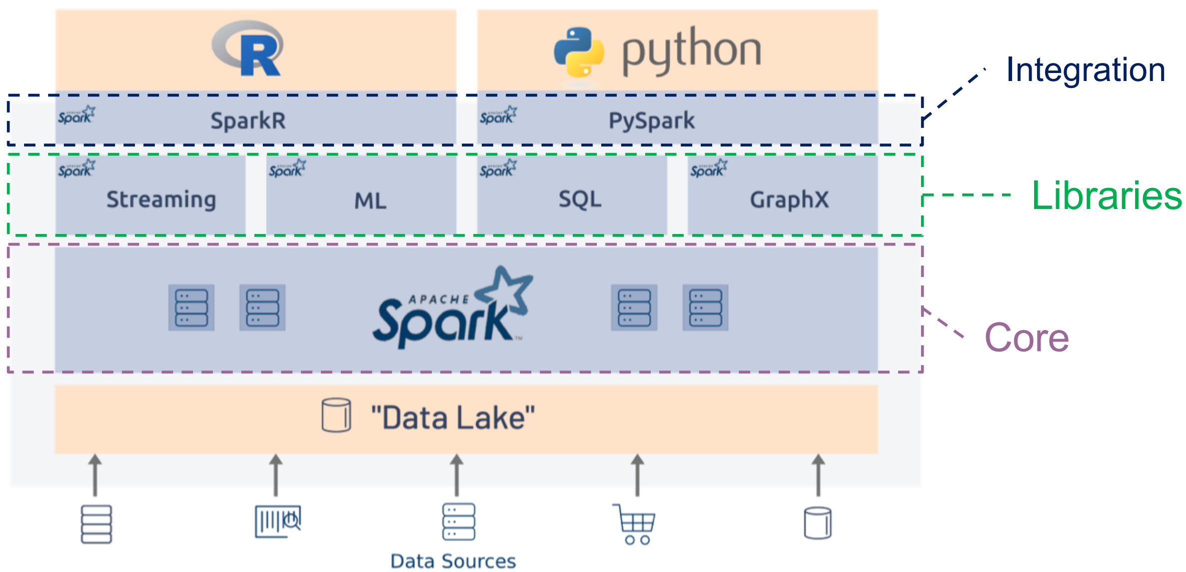
- About Me
- Spark & R
  - Spark Architecture
  - Spark DataFrames and SparkSQL
  - Natively Distributed ML with Spark ML
  - Big Compute: Parallelization with Spark UDFs
  - ML & Data-in-motion: Spark Streaming
- Tips & Pitfalls
- What About Python?
- Summary & Outlook

About Me

- MSc in Computer Science, University of Zurich
- > 20 Years in Consulting
  - Finance, Energy, Telco, Pharma, Manufacturing
  - EAI, BI/Data Warehousing, CRM, ERP, Technology
- Speaker at SparkSummit, HadoopSummit, and others
- Founder & CEO Wireframe AG
  - PatternFinder: Data Science Data Warehouse / Business Machine Intelligence
  - PatternGenerator: Development Tool for Streaming Applications
  - https://wireframe.digital

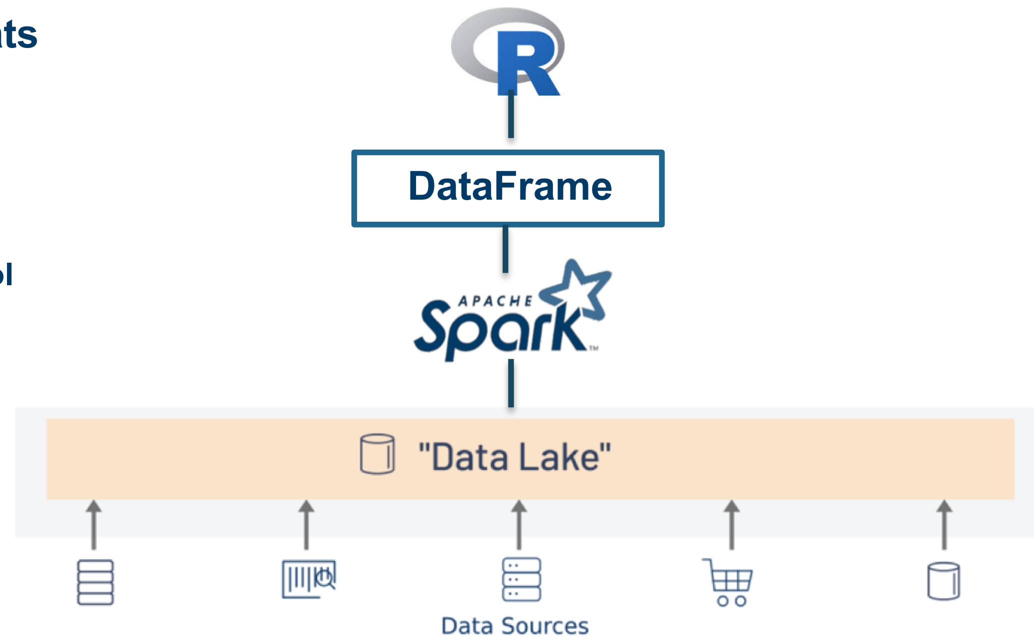
SparkR Architecture

- Execute R on cluster
  - Master/Slave
  - Out-Of-Memory Datasets
- Access Data Lake
- Powerful Libraries
  - Machine Learning
  - SQL
  - Streaming
- R Integration through SparkR



Data (Lake) Access

- Ability to read Big Data File Formats
  - HDFS, AWS S3, Azure WASB, …
  - Parquet, CSV, JSON, ORC, ...
- Security
  - Fine grained authorization
  - Role-/Attribute-based Access Control
- Governance
  - Metadata Management
  - Lineage



SparkSQL

- **Execute SQL against Spark DataFrame**

- **SELECT**
  - **Specify Projection**
- **WHERE**
  - **Filter criteria**
- **GROUPBY**
  - **Group/Aggregate**
- **JOIN**
  - **Join tables**

- **Alternatively, use**
  - **select(), where(), groupBy(), count(), etc.**

```
# show first rows of Spark DataFrame
> head(my_spark_df)
  Sepal_Length Sepal_Width Petal_Length Petal_Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa

# register as view with SparkSQL
> createOrReplaceTempView(iris_sparkdf, "iris_table")

# run SQL against Spark DataFrame
> resdf <- sql("SELECT Species, Sepal_Length, Sepal_Width
                FROM iris_table WHERE Sepal_Length > 6 AND Sepal_Length < 6.2")

# download and display result
> collect(resdf)
     Species Sepal_Length Sepal_Width
1 versicolor          6.1         2.9
2 versicolor          6.1         2.8
3 versicolor          6.1         2.8
4 versicolor          6.1         3.0
5  virginica          6.1         3.0
6  virginica          6.1         2.6
```

Spark MLlib

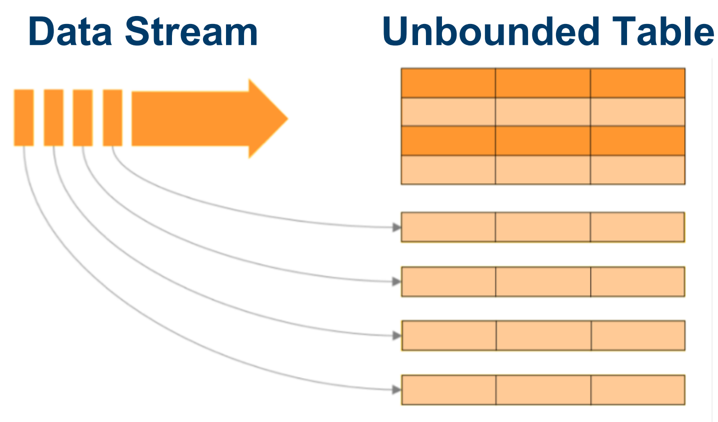
SparkR supports the following machine learning algorithms currently:

- `spark.glm` or `glm`: Generalized Linear Model
- `spark.survreg`: Accelerated Failure Time (AFT) Survival Regression Model
- `spark.naiveBayes`: Naive Bayes Model
- `spark.kmeans`: K-Means Model
- `spark.logit`: Logistic Regression Model
- `spark.isoreg`: Isotonic Regression Model
- `spark.gaussianMixture`: Gaussian Mixture Model
- `spark.lda`: Latent Dirichlet Allocation (LDA) Model
- `spark.mlp`: Multilayer Perceptron Classification Model
- `spark.gbt`: Gradient Boosted Tree Model for Regression and Classification
- `spark.randomForest`: Random Forest Model for Regression and Classification
- `spark.als`: Alternating Least Squares (ALS) matrix factorization Model
- `spark.kstest`: Kolmogorov-Smirnov Test

SparkR & Streaming

Use R to process data streams

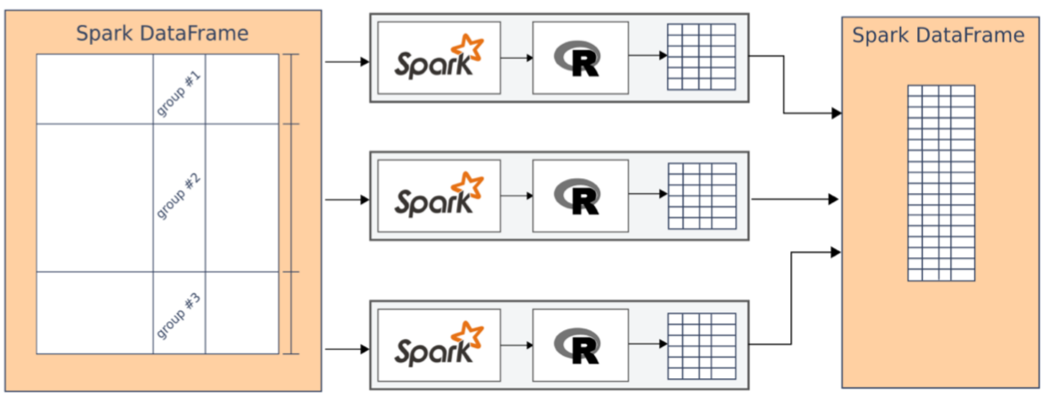
- **Structured Streaming:**
  - **DataFrames with streaming sources**
- **New data in stream is appended to an unbounded table (i.e. DataFrame)**
- **Seamless integration:**
  - **read.stream("kafka", ….)**



SparkR UDFs

SparkR Functions: gapply()/gapplyCollect(), dapply()/dapplyCollect()

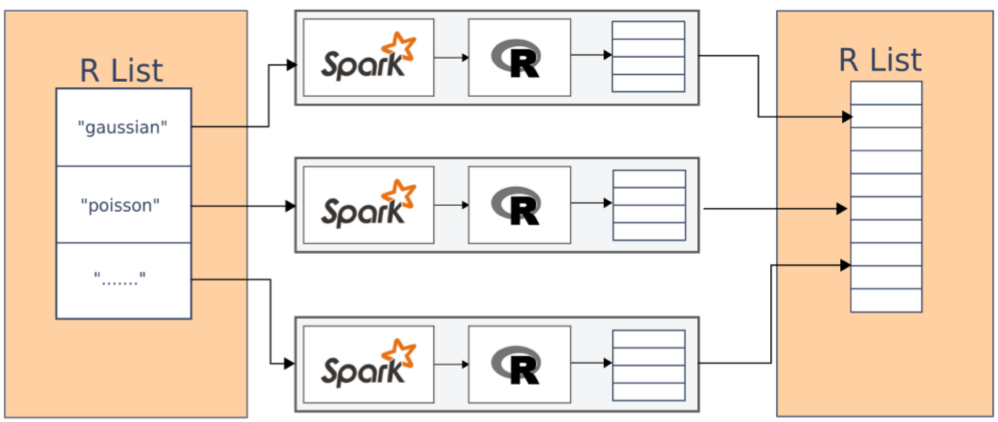
- **Apply a function to each group/partition of a Spark DataFrame**
  - **Input: Grouping Key and DataFrame partition as R data.frame**
  - **Output: R data.frame**



SparkR UDFs

SparkR spark.lapply()

- **Run a function over a list of elements and**
- **Distribute the computation with Spark**



Big Compute

Areas where massively parallel computation is relevant:

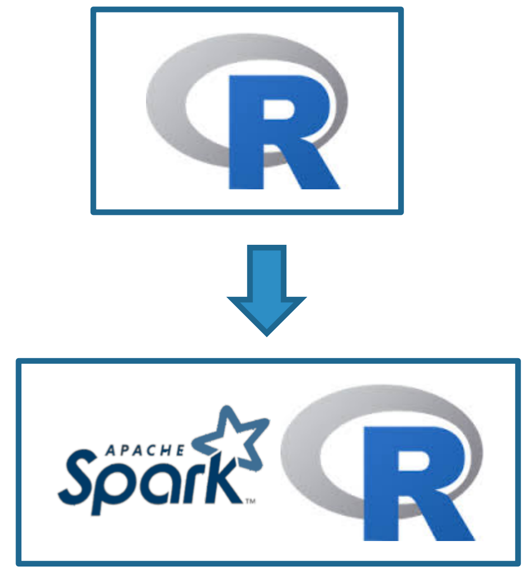
- Ensemble Learning for Time Series
- Hyperparameter Sweeps
- High-Dimensional Problem/Search-Space
  - Wireframe PatternFinder
- Shape/Motif Detection
  - IoT Pattern/Shapes
- Monte-Carlo Simulation
  - Value-at-Risk (Finance)
  - Catastrophe Modeling (Reinsurance)
  - Inventory Optimization (Oil & Gas, Manufacturing)

Big Compute

Massive Time Series Forecasting
- Sequential computation: > 22 hours
- Single-Server, parallelized: > 4.5 hours
- SparkR, Cluster w/ 25 nodes: ~ 12 minutes

Wireframe PatternFinder
- 15.500.000 Models to be computed
  - 50 DataFrames x 5 Dependants x 10 Independants x 5 Models x 100 Segments
- 0.1 Sec. per Model
- Sequential: ~ 18 Days
- 1000 Cores: ~ 26 Minutes

Implications
- Minor refactoring of R code
- Massive cost reduction by using elastically scaling of Cloud resources



Tips & Pitfalls

- Generate diagrams in SparkR
  - PDF, HTML, Rmarkdown
  - Store in shared persistence or serialize into Spark DataFrame
  - SPARK-21866 might be helpful?
- Store complex object (models) from SparkR
  - saveRDS() saves to local storage
  - Store in shared persistence or serialize into Spark DataFrame
- Run R on a YARN cluster w/o locally installed R
  - https://wireframe.digital/sparkr/running-sparkr-on-your-hadoop-cluster-without-pre-installed-r/
- Mixing Scala & R
- Not supported by Oozie's SparkAction
  - Can be replaced with ShellAction
- Not supported by Apache Livy
  - Only support for Scala, Java, and Python

And What About Python?

"Do I need to learn Python?"

Let's compare (Spark)R & Py(Spark):

- Language: Interpreted Languages, R (1993), Python (1991)
- Libraries: CRAN (> 10.000 packages), Numpy, scikit-learn, Pandas
- Package Management
- IDEs/Notebooks: Rstudio/PyCharm, Jupyter, Zeppelin, Databricks Analytics Platform, IBM Watson Studio, Cloudera Data Science Workbench, ….

And there's more:

Market Momentum	Deep Learning
Spark Support	Spark Integration

Market Momentum

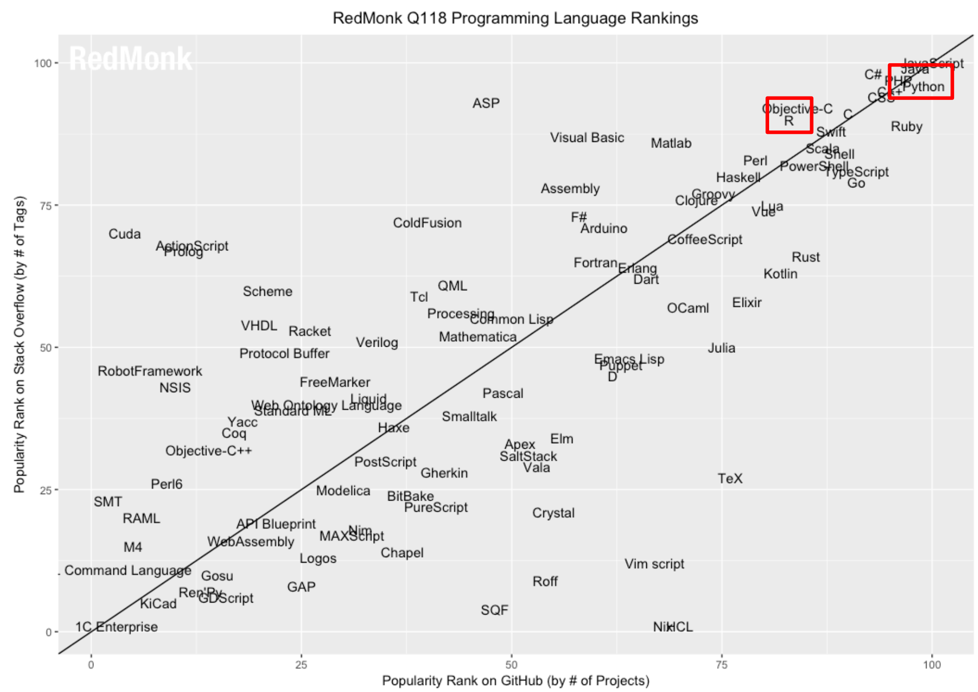
Redmonk (Jan 2018)
1. JavaScript
2. Java
3. Python
4. PHP
5. C#

.
.

10. Swift
11. Objective-C
12. R

TIOBE index (May 2018):

4. Python
11. R



https://redmonk.com/sogrady/2018/03/07/language-rankings-1-18/

Deep Learning

- Python is a first-class citizen in the Deep Learning/Neural Network world
- Using R with these DL frameworks is possible but more complex

	R	Python	Other APIs
TensorFlow	No	Yes	C++, Java, Go, Swift
Keras	Yes	Yes	No
MXNet	Yes	Yes	C++, Scala, Julia, Perl
PyTorch	No	Yes	No
CNTK	No	Yes	C++

SparkR v PySpark

- Both R and Python can access the same types of Spark APIs

	SparkR	PySpark
Data Lake Integration	Yes	Yes
Spark SQL	Yes	Yes
Spark ML	Yes	Yes
UDFs	Yes	Yes
Streaming	Yes	Yes

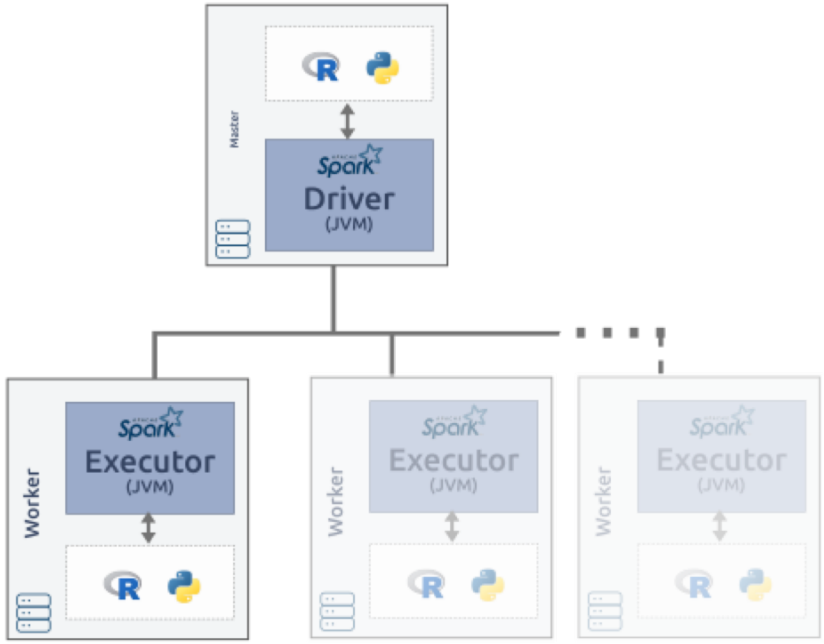
Spark Integration

JVM vs Non-JVM

- **Spark Executors run in JVM**
- **R & Python run in different processes**

- **Data must be moved between both environments (SerDe)**

- **Low performance**



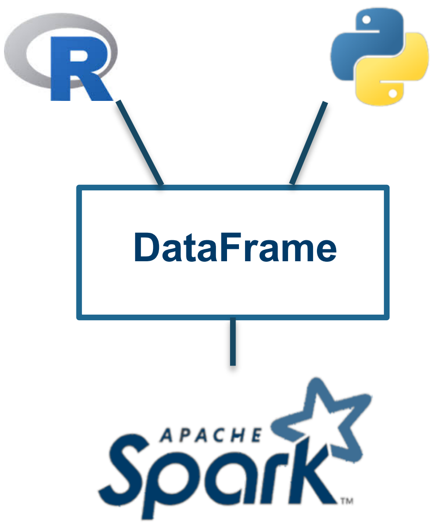
Apache Arrow

In-Memory Columnar Data Format

- Cross-Language
- Optimized for numeric data
- Zero-copy reads (no serialization)

- Support
  - Spark 2.3
  - Python
  - R Bindings not available yet

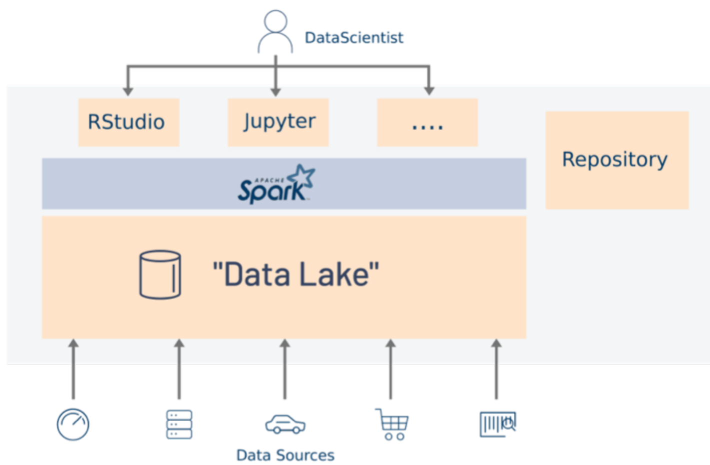
- And more: Parquet, HBase, Cassandra, …
- Will also improve R-Python-Integration (rpy2, reticulate)



See RStudio Blog (04/19/2018): https://blog.rstudio.com/2018/04/19/arrow-and-beyond/

Summary & Outlook

- **Spark is the best option to scale R**
  - **See also sparklyr, R Server for Spark**
- **Common Environment for Dev and Production**
  - **"Looks like R to Data Science, looks like Spark to Data Engineers"**
- **Security & Data Governance**
  - **Row-/Column-Level Access Control**
  - **Full Data Lineage (Up- and Downstream)**
- **Shared Memory Format**
  - **Apache Arrow!**
  - **Mix and Match R, Python, and Scala**
- **Towards an Open Data Science Platform**



Thank You!

Heiko Korndorf
heiko.korndorf@wireframe.digital