A DataFrame Abstraction Layer for SparkR

Chris Freeman

alteryx

Agenda

- What is SparkR?
- History of DataFrames
- Why DataFrames?
- How do DataFrames work?
- Demo
- On the Roadmap
- Questions

What is SparkR?

- New R language API for Spark and SparkSQL
- Exposes existing Spark functionality in an R-friendly syntax via the DataFrame API
- Has its own shell, but can also be imported like a standard R package and used with Rstudio.

What is SparkR?

- An opportunity to make Spark accessible to the large community of R developers who already have clear ideas about how to do analytics in R
- No need to learn a new programming paradigm when working with Spark

History of DataFrames

- SparkR began as an R package that ported Spark's core functionality (RDDs) to the R language.
- The next logical step was to add SparkSQL and SchemaRDDs.
- Initial implementation of SQLContext and SchemaRDDs working in SparkR

History of DataFrames

[SPARK-5097][SQL] DataFrame

```
This pull request redesigns the existing Spark SQL dsl, which already provides data
frame like functionalities.
```

History of DataFrames

[SPARK-5097][SQL] DataFrame

This pull request redesigns the existing Spark SQL dsl, which already provides data frame like functionalities.

```
@deprecated("use toDF", "1.3.0")
def toSchemaRDD: DataFrame = this
```

History of DataFrames

[SPARK-5097][SQL] DataFrame

This pull request redesigns the existing Spark SQL dsl, which already provides data frame like functionalities.

```
@deprecated("use toDF", "1.3.0")
def toSchemaRDD: DataFrame = this
```

Me:

History of DataFrames

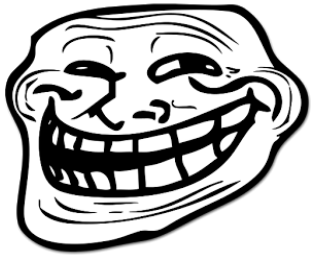
[SPARK-5097][SQL] DataFrame

This pull request redesigns the existing Spark SQL dsl, which already provides data frame like functionalities.

```
@deprecated("use toDF", "1.3.0")
def toSchemaRDD: DataFrame = this
```

Me:

Reynold:



Maybe this isn't such a bad thing…

How can I use Spark to do something simple?

Let's say we wanted to do this with regular RDDs. What would that look like?

```
"Michael, 29"
 "Andy, 30"
"Justin, 19"
 "Bob, 22"
"Chris, 28"
"Garth, 36"
"Tasha, 24"
 "Mac, 30"
"Neil, 32"
```

How can I use Spark to do something simple?

```
peopleRDD <- textFile(sc, “people.txt”)
lines <- flatMap(peopleRDD,
                 function(line) {
                   strsplit(line, ", ")
                 })
```

How can I use Spark to do something simple?

```
peopleRDD <- textFile(sc, “people.txt”)
lines <- flatMap(peopleRDD,
                 function(line) {
                   strsplit(line, ", ")
                 })
ageInt <- lapply(lines,
                 function(line) {
                   as.numeric(line[2])
                 })
```

How can I use Spark to do something simple?

```
peopleRDD <- textFile(sc, “people.txt”)
lines <- flatMap(peopleRDD,
                 function(line) {
                   strsplit(line, ", ")
                 })
ageInt <- lapply(lines,
                 function(line) {
                   as.numeric(line[2])
                 })
sum <- reduce(ageInt,function(x,y) {x+y})
```

How can I use Spark to do something simple?

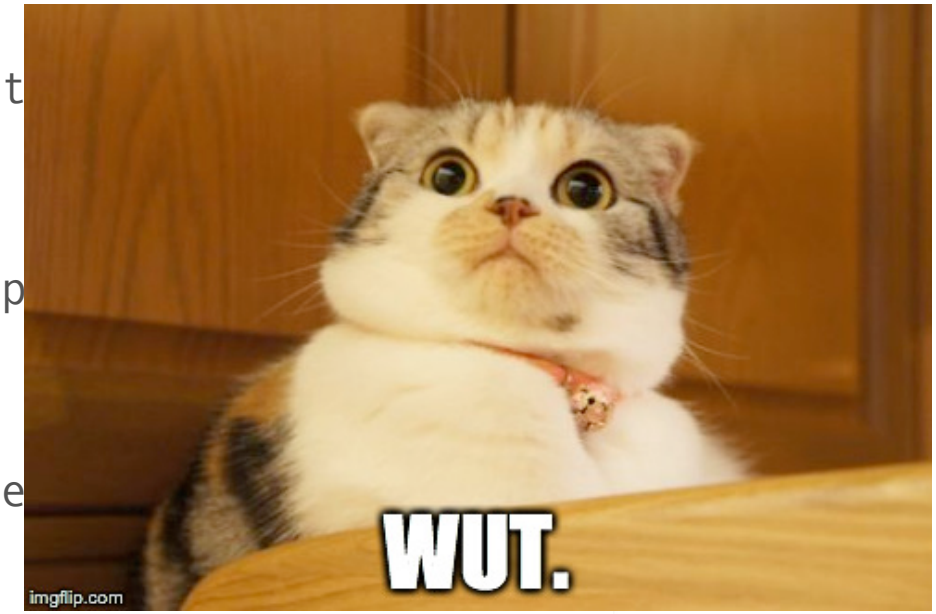
```
peopleRDD <- textFile(sc, “people.txt”)
lines <- flatMap(peopleRDD,
                 function(line) {
                   strsplit(line, ", ")
                 })
ageInt <- lapply(lines,
                 function(line) {
                   as.numeric(line[2])
                 })
sum <- reduce(ageInt,function(x,y) {x+y})
avg <- sum / count(peopleRDD)
```

How can I use Spark to do something simple?

```
peopleRDD <-
lines <- flat


ageInt <- lap


sum <- reduce
avg <- sum /
```



There's got to be a better way.

What I'd hoped to see

```
{"name":"Michael", "age":29}
{"name":"Andy", "age":30}
{"name":"Justin", "age":19}
{"name":"Bob", "age":22}
{"name":"Chris", "age":28}
{"name":"Garth", "age":36}
{"name":"Tasha", "age":24}
{"name":"Mac", "age":30}
{"name":"Neil", "age":32}
```

What I'd hoped to see

```
df <- read.df(sqlCtx, "people.json", "json")
```

What I'd hoped to see

```
df <- read.df(sqlCtx, "people.json", "json")

avg <- select(df, avg(df$age))
```

Why DataFrames?

- Uses the distributed, parallel capabilities offered by RDDs, but imposes a schema on the data
- More structure == Easier access and manipulation
- Natural extension of existing R conventions since DataFrames are already the standard

Why DataFrames?

- Super awesome distributed, in-memory collections

Why DataFrames?

- Super awesome distributed, in-memory collections
- Schemas == metadata, structure, declarative instead of imperative

Why DataFrames?

- Super awesome distributed, in-memory collections
- Schemas == metadata, structure, declarative instead of imperative
- ????

Why DataFrames?

- Super awesome distributed, in-memory collections
- Schemas == metadata, structure, declarative instead of imperative
- ????
- Profit

DataFrames in SparkR

- Multiple Components:
  - A set of native S4 classes and methods that live inside a standard R package
  - A SparkR backend that passes data structures and method calls to the JVM
  - A set of “helper” methods written in Scala

Why does the structure matter?

- Native R classes allow us to extend the existing DataFrame API by adding R-like syntax and interactions
- Handoff to the JVM gives us full access to Spark's DAG capabilities and Catalyst optimizations, e.g. constant-folding, predicate pushdown, and code generation.

SparkR DataFrame Features

- Column access using '$' or '[ ]' just like in R
- dplyr-like DataFrame manipulation:
  - filter
  - groupBy
  - summarize
  - mutate
- Access to external R packages that extend R syntax

Demo Time!

On the Roadmap

- Spark 1.4: SparkR becomes an official API
  - Primarily focused on SparkSQL/DataFrame implementation
- Spark 1.5: Extend SparkR to include machine learning capabilities (e.g. sparkML)
- For more information, be sure to check out "**SparkR: The Past, Present, and Future**" at 4:30 on the Data Science track.

Integration with alteryx

- Drag-and-drop GUI for data analysis
- Spark functionality built directly into existing tools using SparkR
- Interact with a remote Spark cluster from your desktop via Alteryx Designer
- Combine local and in-database data sources in one workflow.

Developer Community

- SparkR originated at UC Berkeley AMPLAB, with additional contributions from Alteryx, Intel, Databricks, and others.
- Working on integration with Spark Packages
  - Easily extend Spark with new functionality and distribute via the Spark Package repository

Questions?

Slides, Demo, and Data available on GitHub at:

https://github.com/cafreeman/SparkR_DataFrame_Demo


@15lettermax


cafreeman