Enabling Exploratory Data Science with Spark and R

Shivaram Venkataraman, Hossein Falaki (@mhfalaki)

databricks™

About Apache Spark, AMPLab and Databricks

Apache Spark is a general distributed computing engine that unifies:

- Real-time streaming (Spark Streaming)
- Machine learning (SparkML/MLLib)
- SQL (SparkSQL)
- Graph processing (GraphX)

AMPLab (Algorithms, Machines, and Peoples lab) at UC Berkeley was where Spark and SparkR were developed originally.

Databricks Inc. is the company founded by creators of Spark, focused on making big data simple by offering an end to end data processing platform in the cloud

What is R?

Language and runtime

The corner stone of R is the data frame concept

TIOBE Index for R
Source: www.tiobe.com
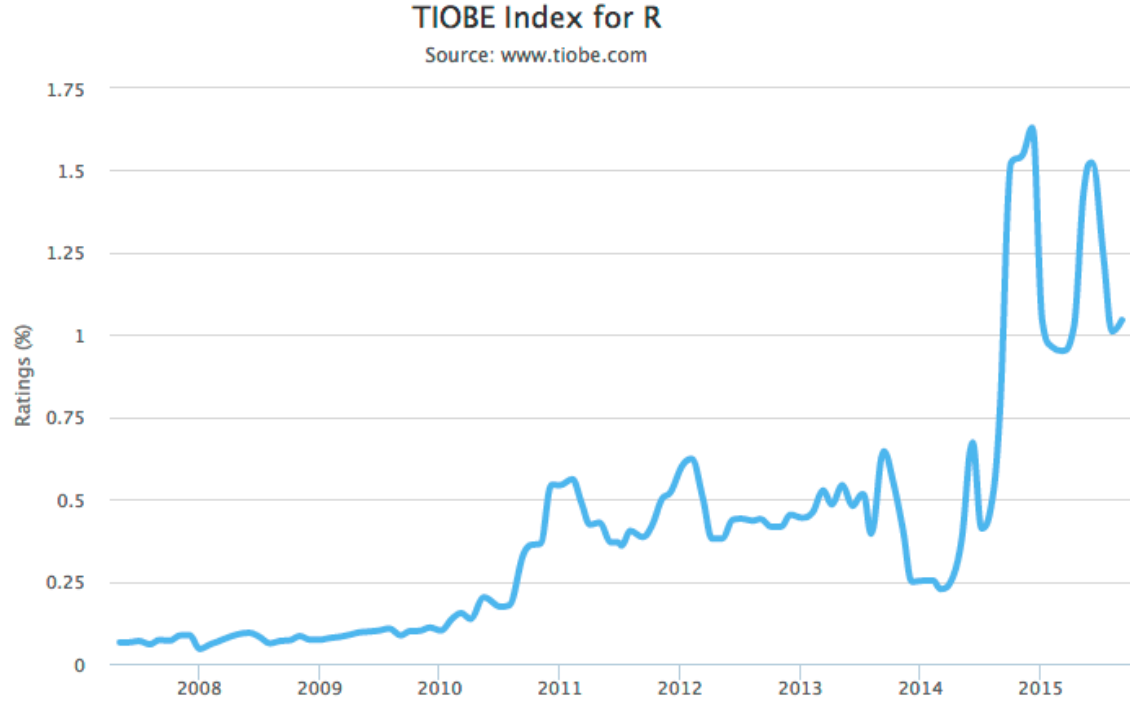
Many data scientists love R

- Open source
- Highly dynamic
- Interactive environment
- Rich ecosystem of packages
- Powerful visualization infrastructure
- Data frames make data manipulation convenient
- Taught by many schools to stats and computing students

Performance Limitations of R

R language

- R's dynamic design imposes restrictions on optimization

R runtime
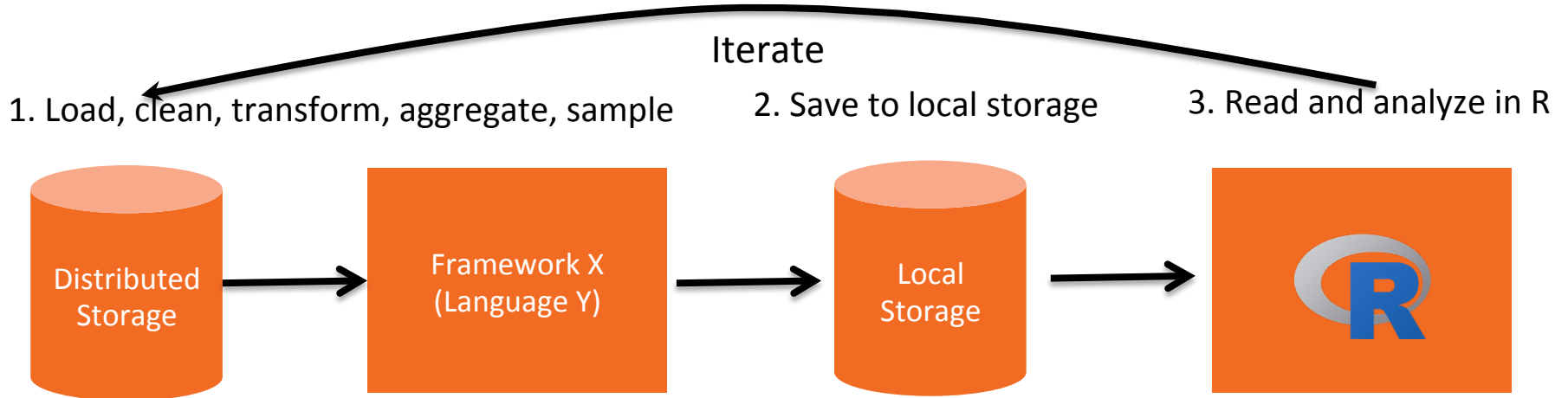
- Single threaded
- Everything has to fit in memory

databricks™

What would be ideal?

Seamless manipulation and analysis of very large data in R

- R's flexible syntax

- R's rich package ecosystem

- R's interactive environment

- Scalability (scale up and out)

- Integration with distributed data sources / storage

databricks™

Augmenting R with other frameworks

In practice data scientists use R in conjunction with other frameworks (Hadoop MR, Hive, Pig, Relational Databases, etc)

Iterate

1. Load, clean, transform, aggregate, sample   2. Save to local storage   3. Read and analyze in R

| Distributed Storage | → | Framework X (Language Y) | → | Local Storage | → | R |

databricks™

What is SparkR?

An R package distributed with Apache Spark:

- Provides R frontend to Spark
- Exposes Spark Dataframes (inspired by R and Pandas)
- Convenient interoperability between R and Spark DataFrames

Spark

distributed/robust processing, data sources, off-memory data structures

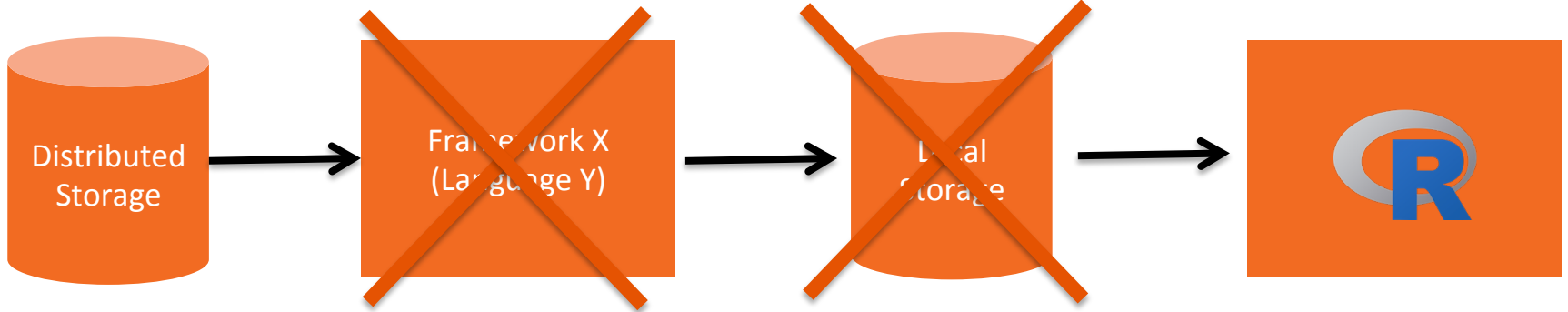**+**

R

Dynamic environment, interactivity, packages, visualization

How does SparkR solve our problems?

Iterate

1. Load, clean, transform, aggregate, sample

2. Save to local storage

3. Read and analyze in R

Distributed Storage

Framework X (Language Y)

Local Storage

R

No local storage involved

Write everything in R

Use Spark's distributed cache for interactive/iterative analysis at speed of thought

databricks™

Example SparkR program

```
# Loading distributed data
df <- read.df("hdfs://bigdata/logs", source = "json")

# Distributed filtering and aggregation
errors <- subset(df, df$type == "error")
counts <- agg(groupBy(errors, df$code), num = count(df$code))

# Collecting and plotting small data
qplot(code, num, data = collect(counts), geom = "bar", stat =
    "identity") + coord_flip()
```
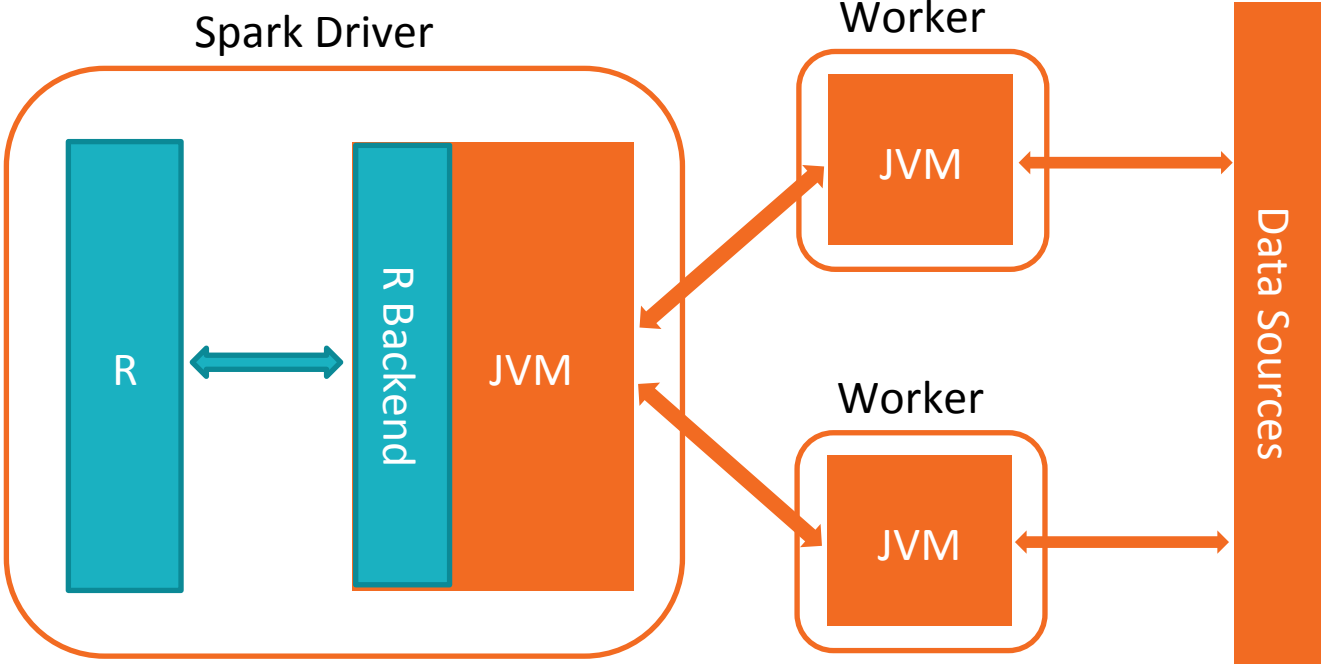
SparkR architecture

Overview of SparkR API

IO

- **read.df** / **write.df**
- **createDataFrame** / **collect**

Caching

- cache / persist / unpersist
- cacheTable / uncacheTable

Utility functions

- dim / head / take
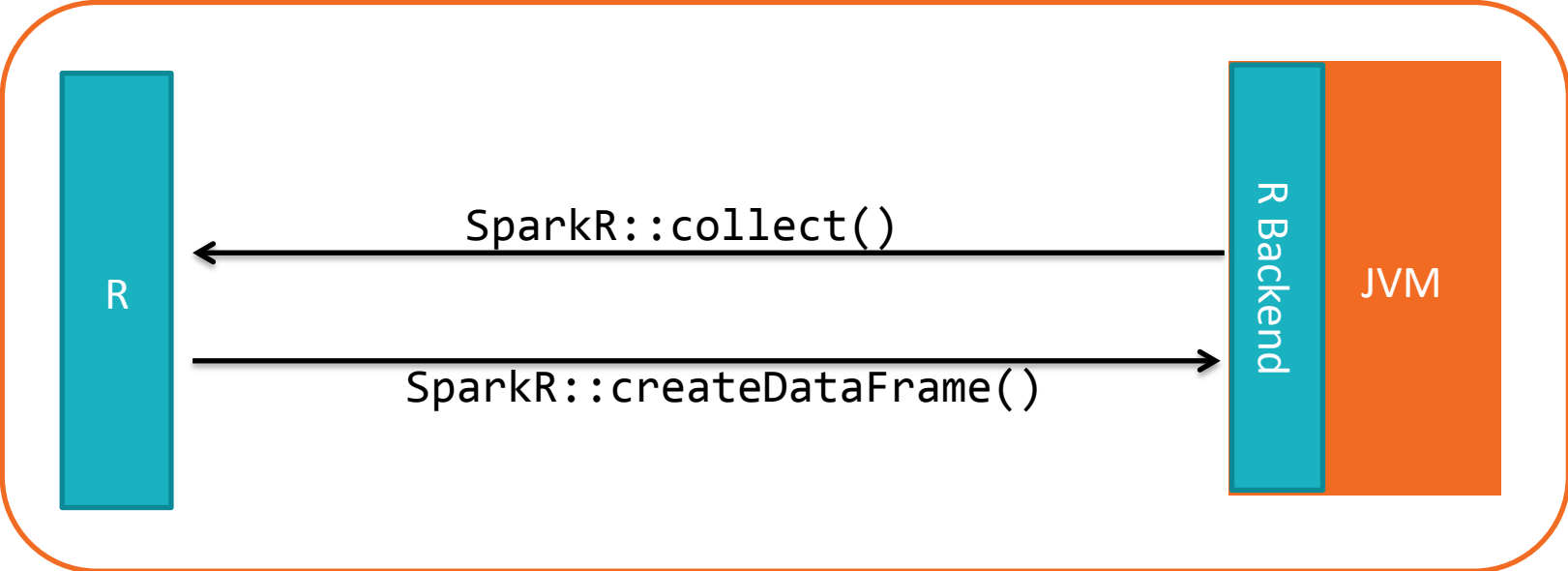- names / rand / sample / ...

ML Lib

- **glm** / predict

DataFrame API

select / subset / groupBy
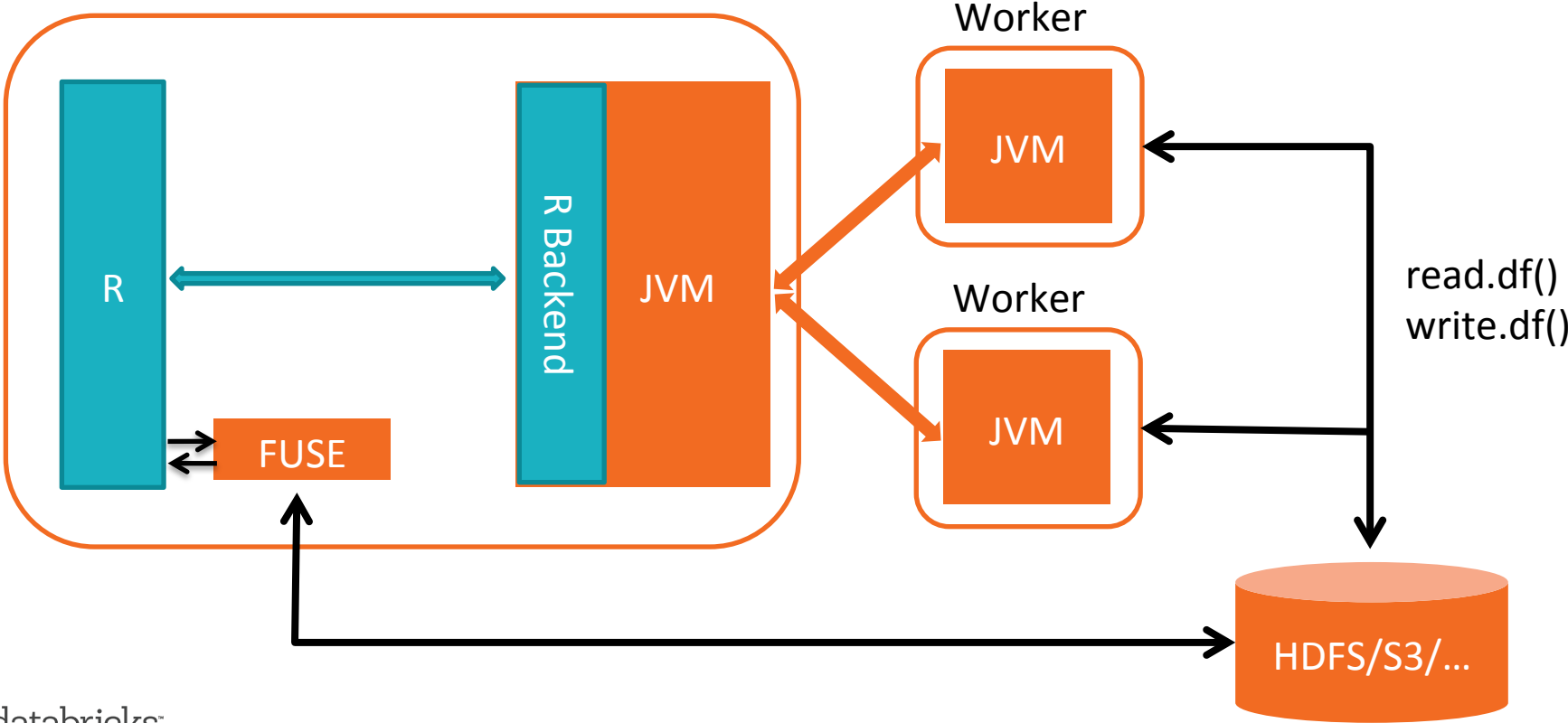head / showDF /unionAll
agg / avg / column / ...

SQL

sql / table / saveAsTable
registerTempTable / tables

Moving data between R and JVM

Moving data between R and JVM

Moving between languages

R

```
df <- read.df(...)

wiki <- filter(df, ...)

registerTempTable(wiki,
"wiki")
```

Scala

```
val wiki = table("wiki")

val parsed = wiki.map {
  Row(_, _, text: String,
_, _) =>text.split(' ')
}


val model =
Kmeans.train(parsed)
```

Spark

databricks™

Mixing R and SQL

Pass a query to SQLContext and get the result back as a DataFrame

```
# Register DataFrame as a table
registerTempTable(df, "dataTable")


# Complex SQL query, result is returned as another DataFrame
aggCount <- sql(sqlContext, "select count(*) as num, type, date
    group by type order by date desc")


qplot(date, num, data = collect(aggCount), geom = "line")
```

SparkR roadmap and upcoming features

- Exposing MLLib functionality in SparkR
  - GLM already exposed with R formula support
- UDF support in R
  - Distribute a function and data
  - Ideal way for distributing existing R functionality and packages
- Complete DataFrame API to behave/feel just like data.frame

databricks™

Example use case: exploratory analysis

- Data pipeline implemented in Scala/Python
- New files are appended to existing data partitioned by time
- Table scheme is saved in Hive metastore
- Data scientists use SparkR to analyze and visualize data
  1. `refreshTable(sqlConext, "logsTable")`
  2. `logs <- table(sqlContext, "logsTable")`
  3. Iteratively analyze/aggregate/visualize using Spark & R DataFrames
  4. Publish/share results

databricks™

Demo

databricks™

How to get started with SparkR?

- On your computer
  1. Download latest version of Spark (1.5.2)
  2. Build (maven or sbt)
  3. Run `./install-dev.sh` inside the R directory
  4. Start R shell by running `./bin/sparkR`
- Deploy Spark (1.4+) on your cluster
- Sign up for 14 days free trial at Databricks

Summary

1. SparkR is an R frontend to Apache Spark

2. Distributed data resides in the JVM

3. Workers are not running R process (yet)

4. Distinction between Spark DataFrames and R data frames

databricks™

Further pointers

http://spark.apache.org

http://www.r-project.org

http://www.ggplot2.org

https://cran.r-project.org/web/packages/magrittr

www.databricks.com

Office hour: 13-14 Databricks Booth

Thank you

Spark

databricks™