Accessing Distributed Systems from R

Giorgi Jvaridze @0xh3x

Dublin R

26 Jan 2016

Overview

Some History

Introduction to Spark

Spark R

Demo

Summary

Some History

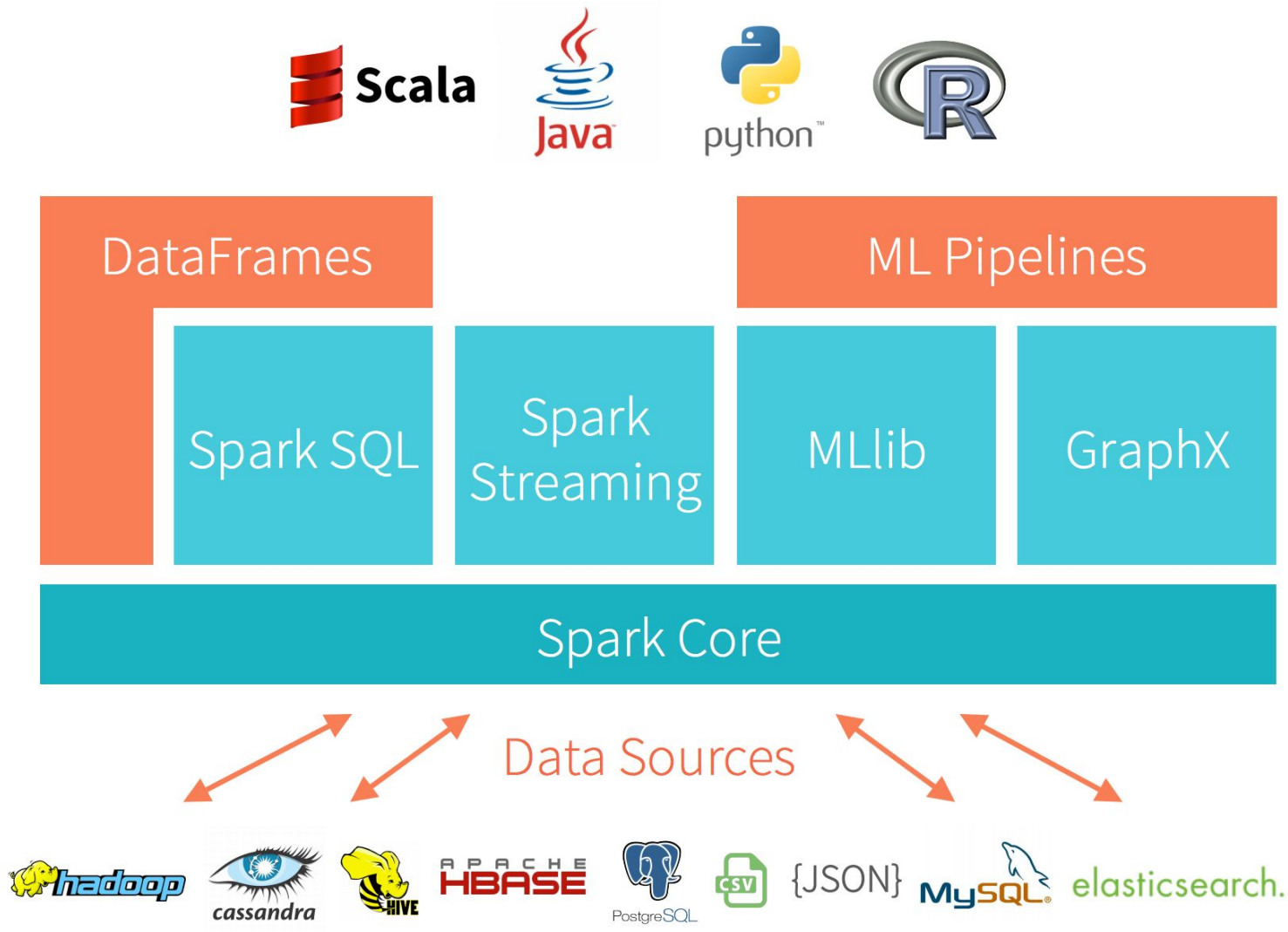
2003 - Google File System

2004 - MapReduce

2005 - Hadoop

2009 - Spark

Apache Spark™ is a fast and general engine for large-scale data processing.


Scala
Java
python
R
DataFrames
ML Pipelines
Spark SQL
Spark Streaming
MLlib
GraphX
Spark Core
Data Sources
hadoop
cassandra
HIVE
APACHE HBASE
PostgreSQL
CSV
{JSON}
MySQL
elasticsearch.

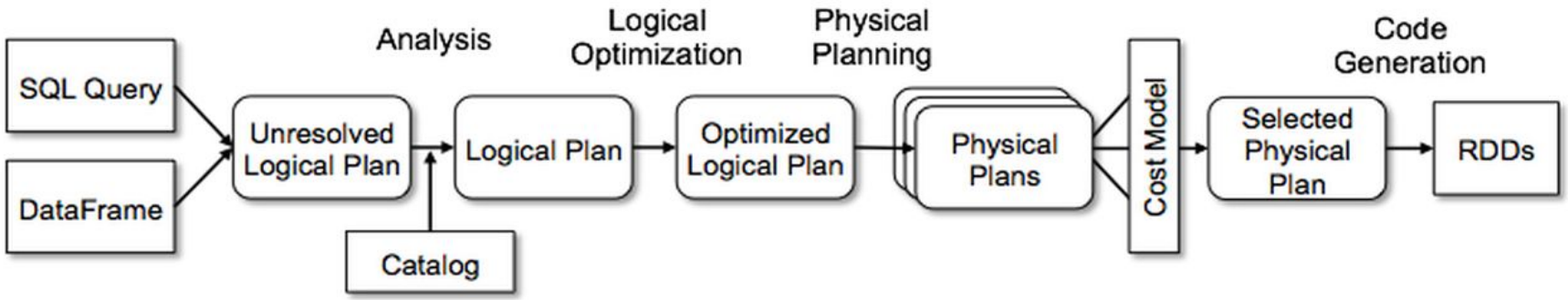
Spark Core

RDD - Resilient Distributed Dataset

- Collections of objects spread across a cluster
- Automatically rebuild on failure
- Expressive API
- Transformations (e.g. map, filter, groupBy)
- Actions (e.g. count, collect, save)

Spark SQL / DataFrames

- Inspired by **R**'s data.frame!
- Built on top of RDDs
- Declarative API
- Catalyst Optimizer



Less Code


```
private IntWritable one =
  new IntWritable(1)
private IntWritable output =
  new IntWritable()
proctected void map(
    LongWritable key,
    Text value,
    Context context) {
  String[] fields = value.split("\t")
  output.set(Integer.parseInt(fields[1]))
  context.write(one, output)
}

IntWritable one = new IntWritable(1)
DoubleWritable average = new DoubleWritable()

protected void reduce(
    IntWritable key,
    Iterable<IntWritable> values,
    Context context) {
  int sum = 0
  int count = 0
  for(IntWritable value : values) {
     sum += value.get()
     count++
    }
  average.set(sum / (double) count)
  context.Write(key, average)
}
```


```
data = sc.textFile(...).split("\t")
data.map(lambda x: (x[0], [x.[1], 1])) \
   .reduceByKey(lambda x, y: [x[0] + y[0], x[1] + y[1]]) \
   .map(lambda x: [x[0], x[1][0] / x[1][1]]) \
   .collect()
```

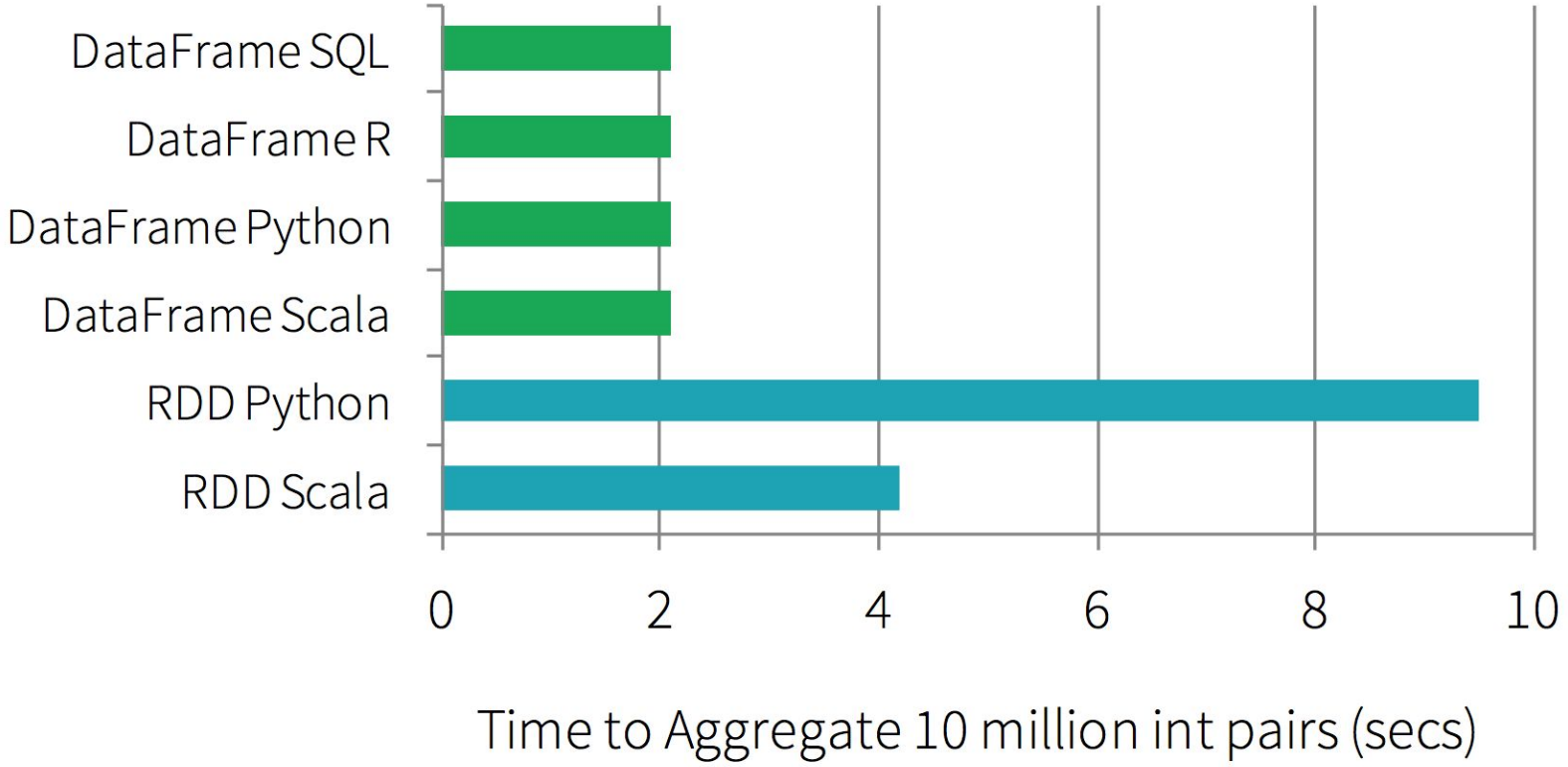
Using SQL

```
SELECT name, avg(age)
FROM people
GROUP BY name
```

Using DataFrames

```
sqlCtx.table("people") \
   .groupBy("name") \
   .agg("name", avg("age")) \
   .map(lambda …) \
   .collect()
```

Faster


DataFrame SQL
DataFrame R
DataFrame Python
DataFrame Scala
RDD Python
RDD Scala
0
2
4
6
8
10
Time to Aggregate 10 million int pairs (secs)

We <3 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

But...

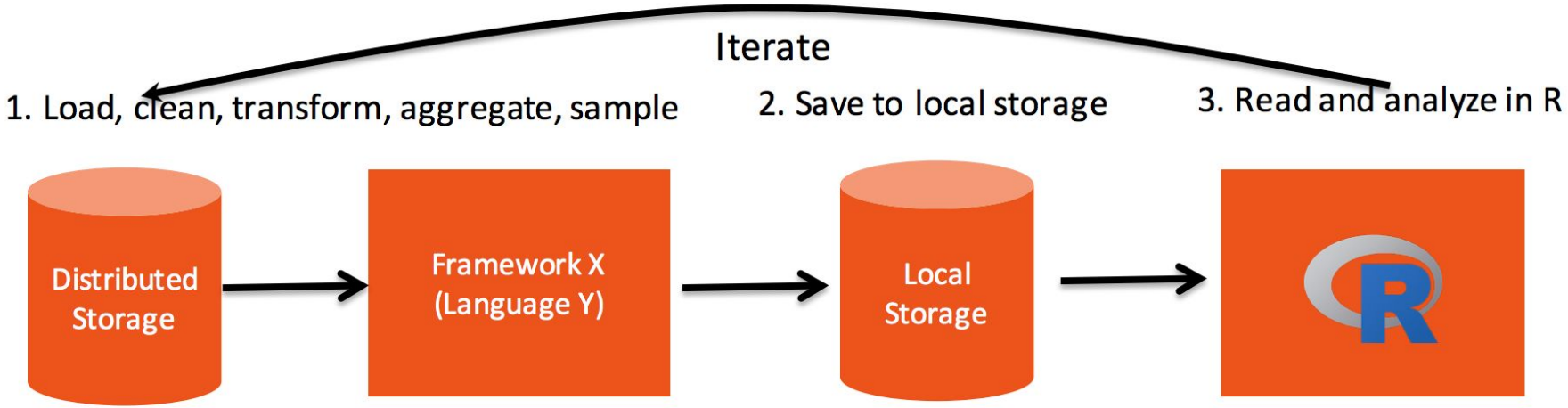
- R’s dynamic design imposes restrictions on optimization
- Single threaded
- Everything has to fit in memory

But...

- R’s dynamic design imposes restrictions on optimization
- Single threaded
- Everything has to fit in memory

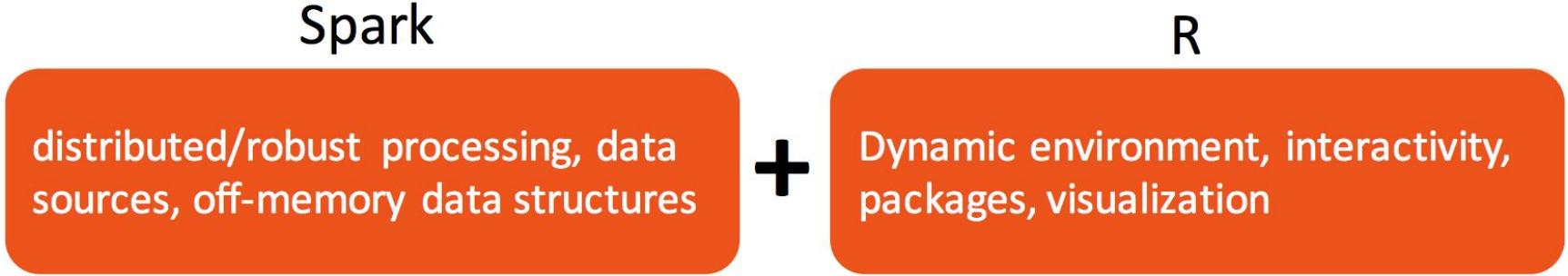
Not ideal for some kinds of tasks

Augmenting R with other frameworks

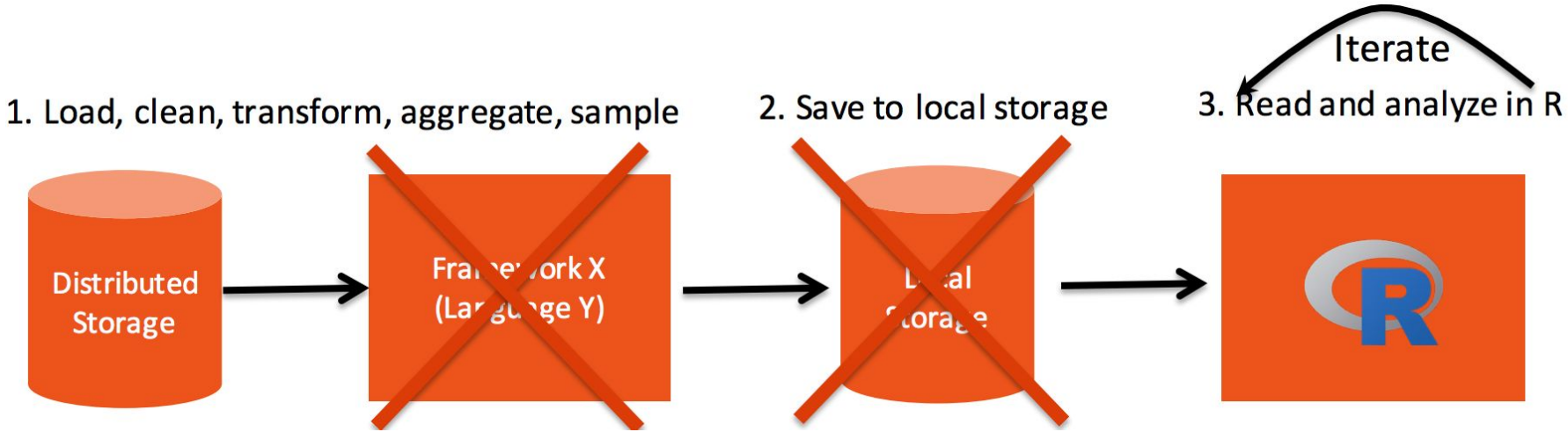


Meet SparkR

- R frontend to Spark
- Exposes Spark's DataFrame API
- Interoperability between R and Spark DataFrames



How does SparkR solve our problems?

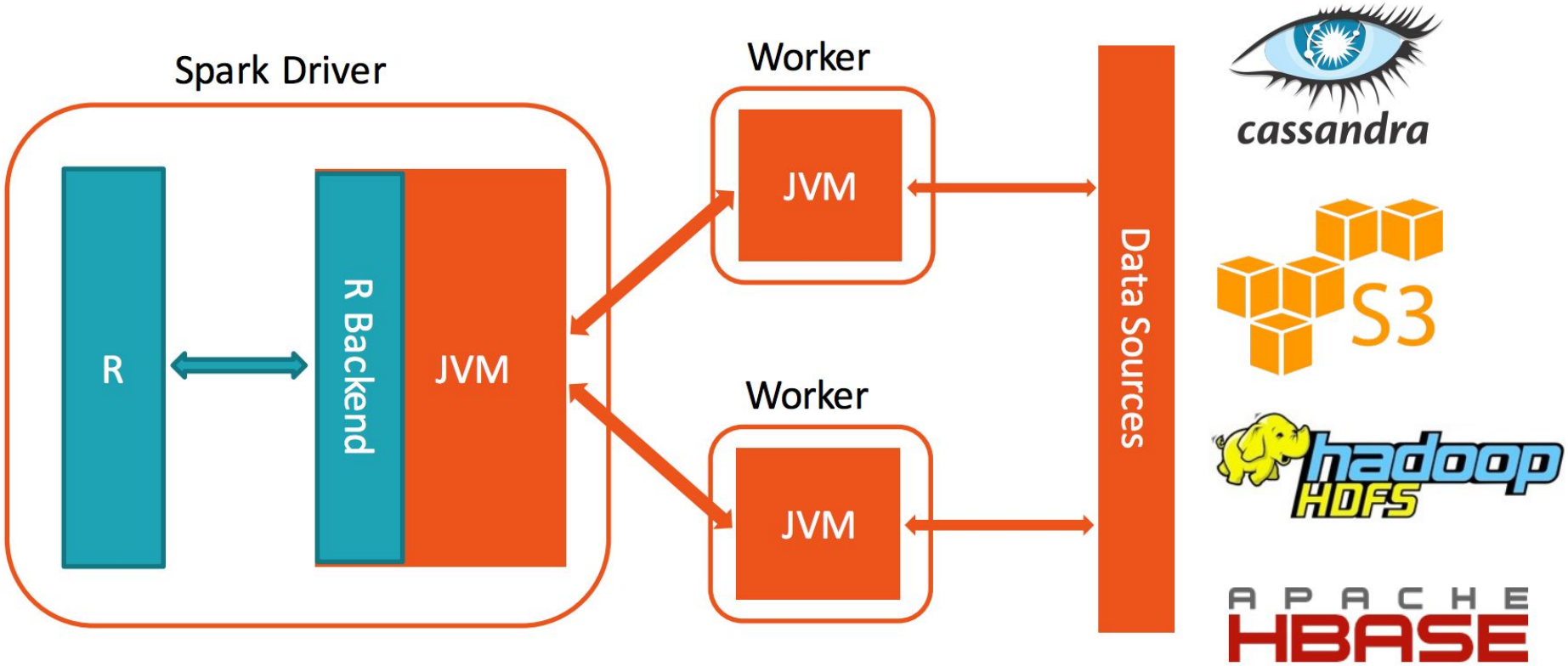


- No local storage involved
- Write everything in R
- Use Spark's distributed cache for interactive/iterative analysis

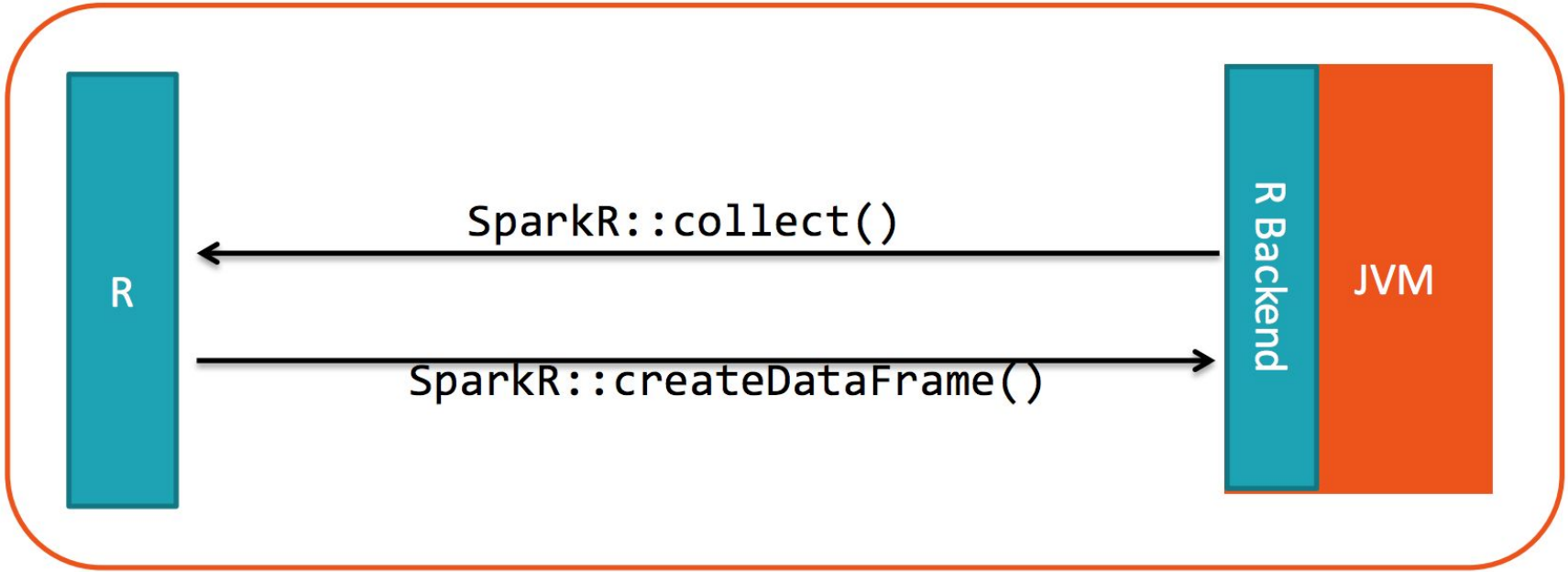
Example SparkR program

```
# 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



Moving data between R and JVM



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

Demo

Summary

Apache Spark is distributed data processing engine

SparkR is an R frontend to Apache Spark

The project is in its early stages, so not everything is supported yet

Useful for some kind of scenarios

Thanks!