SparkR: Interactive R at scale

Shivaram Venkataraman
Zongheng Yang

Fast !

Scalable

Interactive

Statistics !

Packages

Plots

Fast !

Statistics !

Scalable

+

Packages

Interactive Shell

Plots

RDD

Transformations

map
filter
groupby
…

Actions

count
collect
save
…

R

Q: How can I use a loop to [...insert task here...] ?
A: Don't. Use one of the apply functions.

From: http://nsaunders.wordpress.com/2010/08/20/a-brief-introduction-to-apply-in-r/

R + RDD = R2D2

R + RDD =
RRDD

lapply
lapplyPartition
groupByKey
reduceByKey
sampleRDD
collect
cache
…

broadcast
includePackage
textFile
parallelize

Example: Word Count

```
lines <- textFile(sc, args[[2]])
```

Example: Word Count

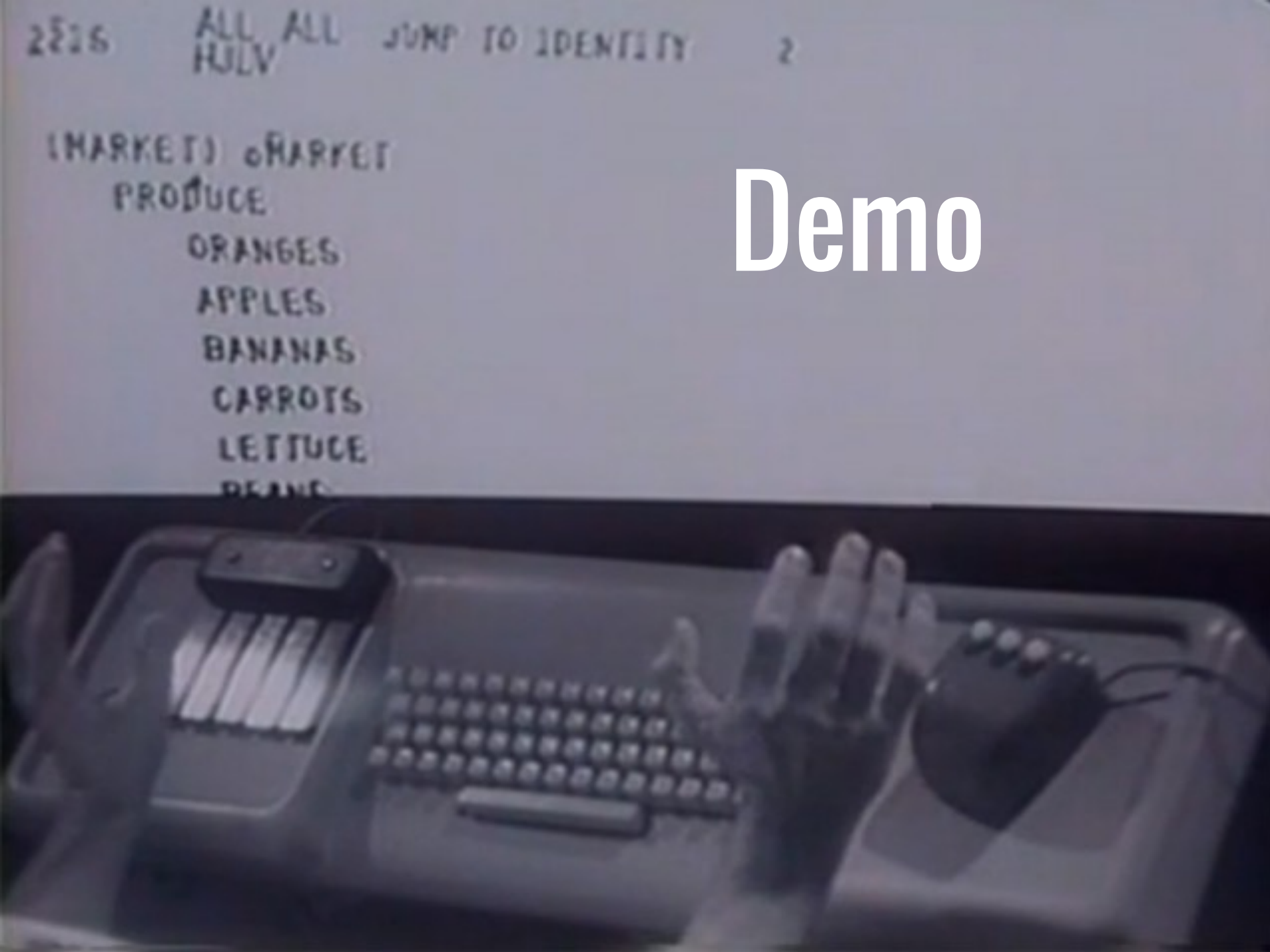
```
lines <- textFile(sc, args[[2]])

words <- flatMap(lines,
                 function(line) {
                   strsplit(line, " ")[[1]]
                 })
wordCount <- lapply(words,
                    function(word) {
                        list(word, 1L)
                    })
```

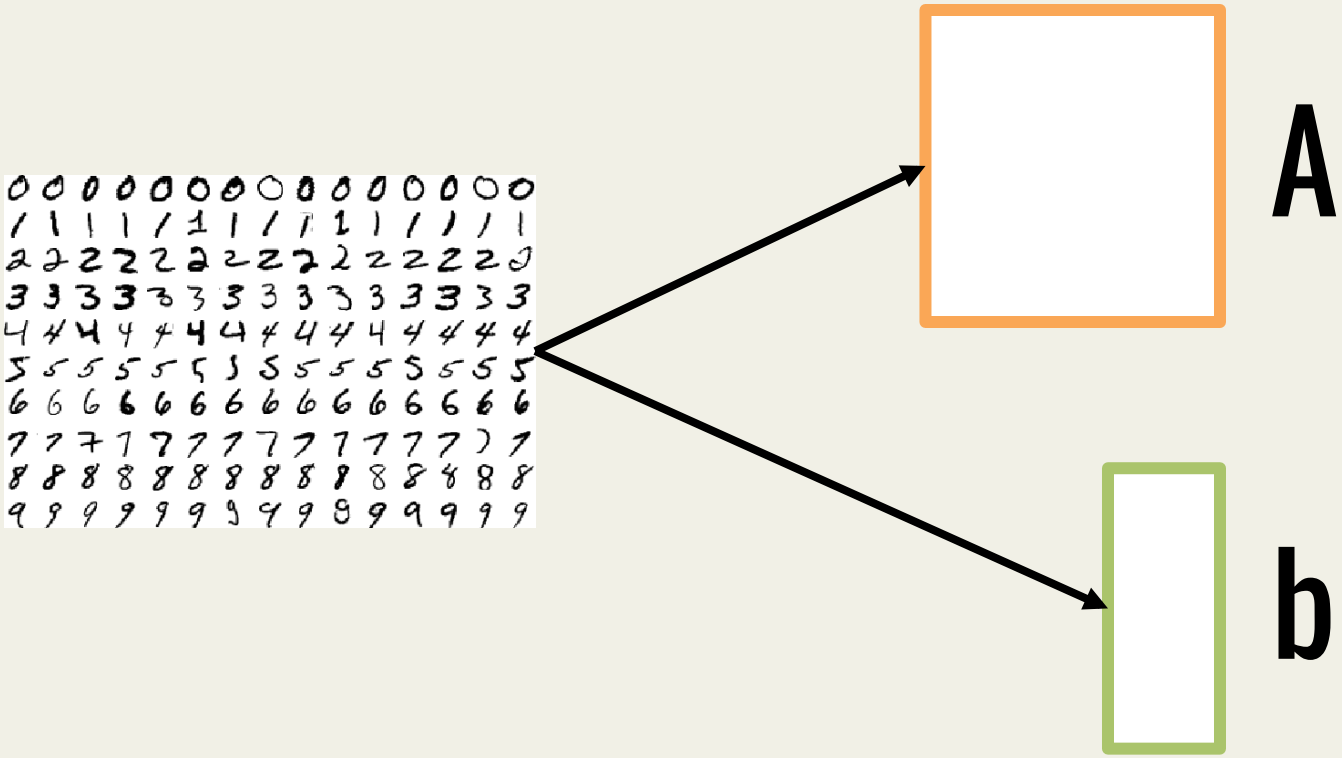
Example: Word Count

```
lines <- textFile(sc, args[[2]])

words <- flatMap(lines,
                 function(line) {
                   strsplit(line, " ")[[1]]
                 })
wordCount <- lapply(words,
                    function(word) {
                        list(word, 1L)
                    })
counts <- reduceByKey(wordCount, "+", 2L)
output <- collect(counts)
```


2216 ALL ALL JUMP TO IDENTITY 2
HULV
(MARKET) cMARKET
PRODUCE
ORANGES
APPLES
BANANAS
CARROTS
LETTUCE
Demo

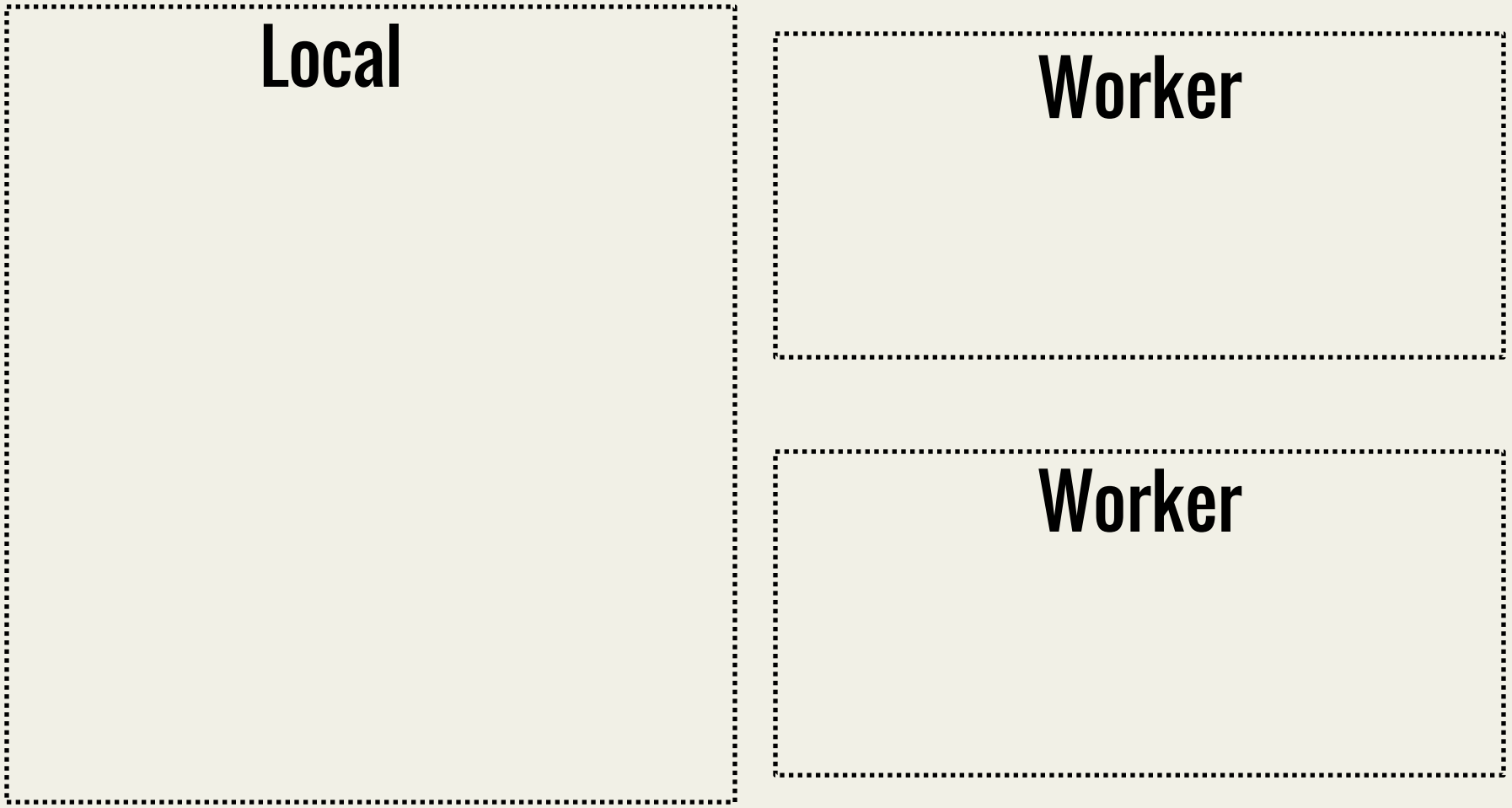
MNIST



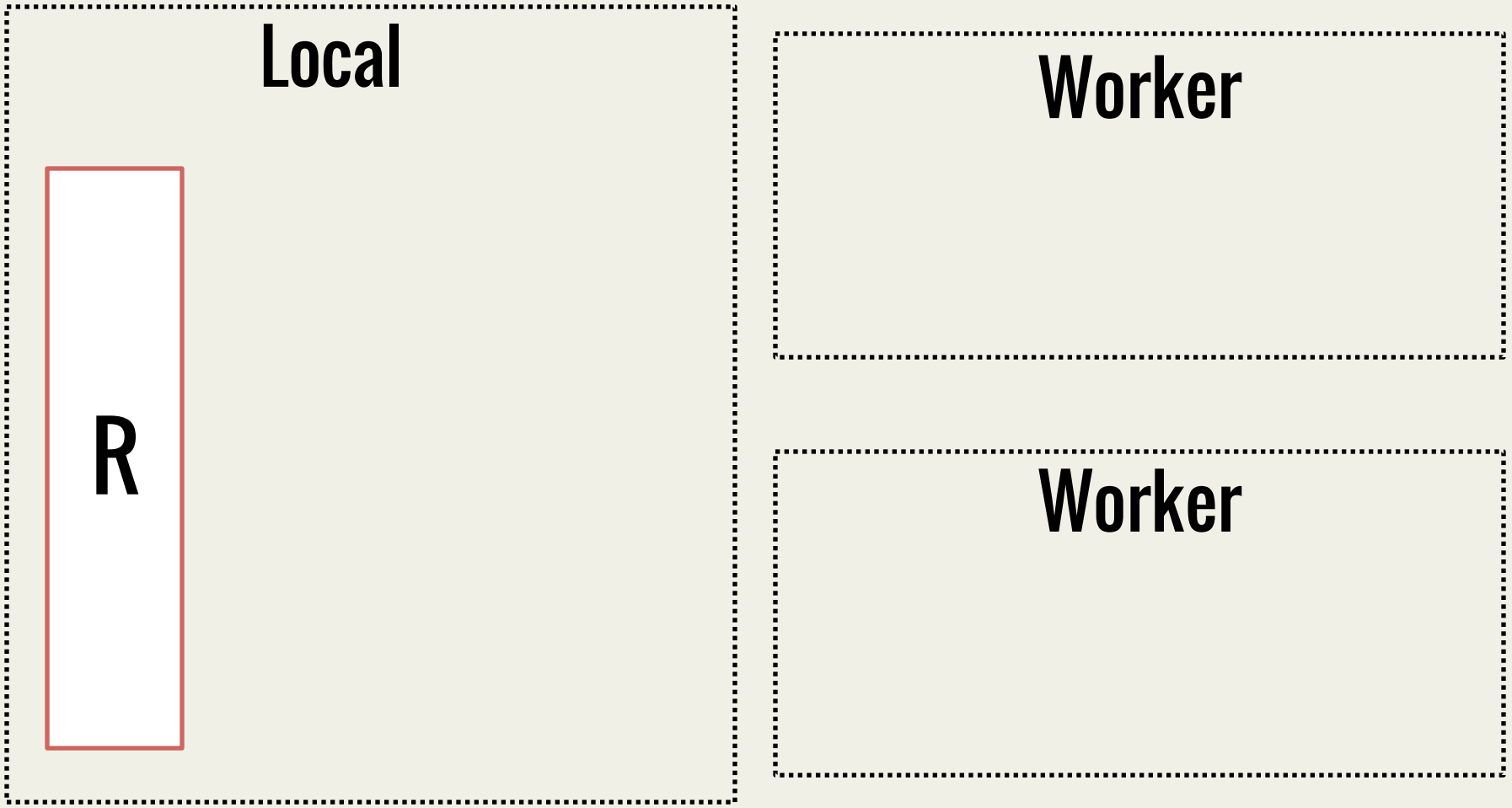
Minimize $\|Ax - b\|_2$


How does this work ?

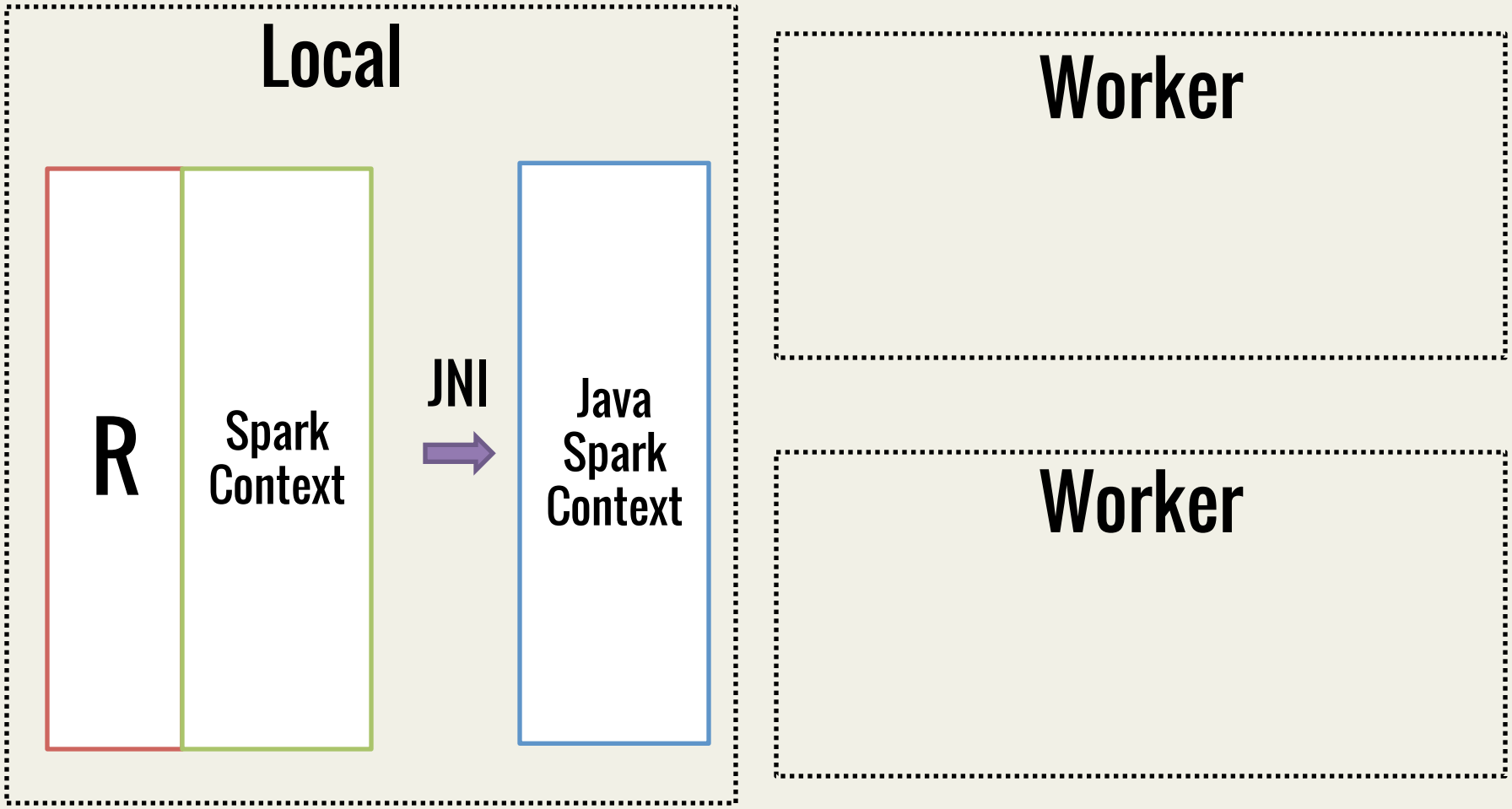
Dataflow



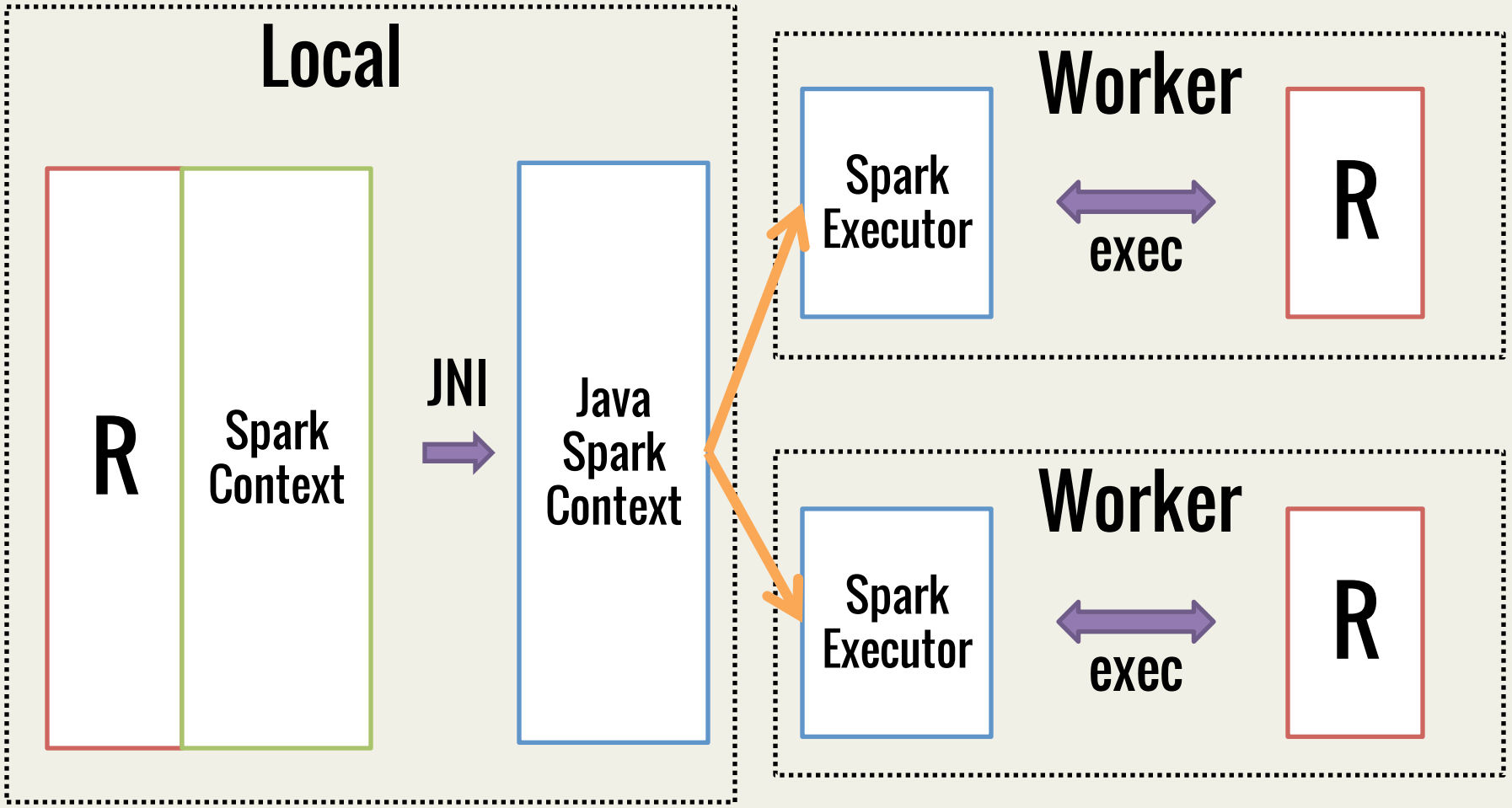
Dataflow

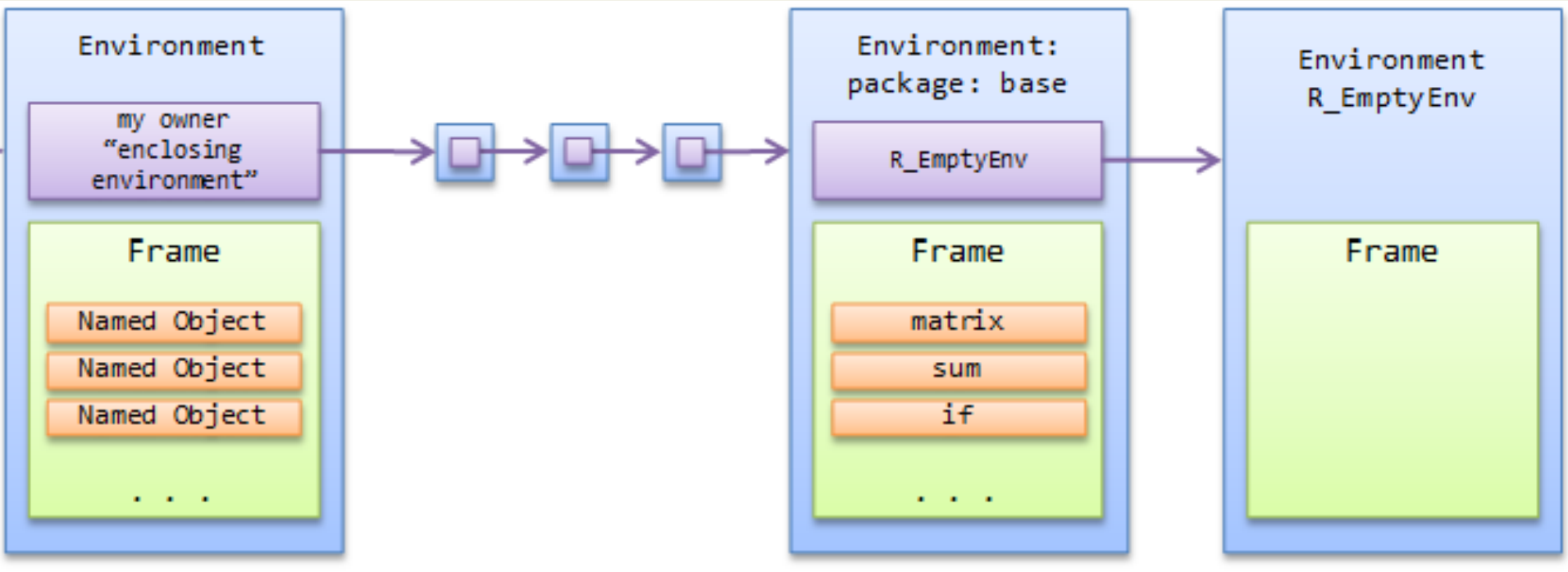


Dataflow



Dataflow


Local
R
Spark Context
JNI
Java Spark Context
Worker
Spark Executor
exec
R
Worker
Spark Executor
exec
R



From http://obeautifulcode.com/R/How-R-Searches-And-Finds-Stuff/

save {base}

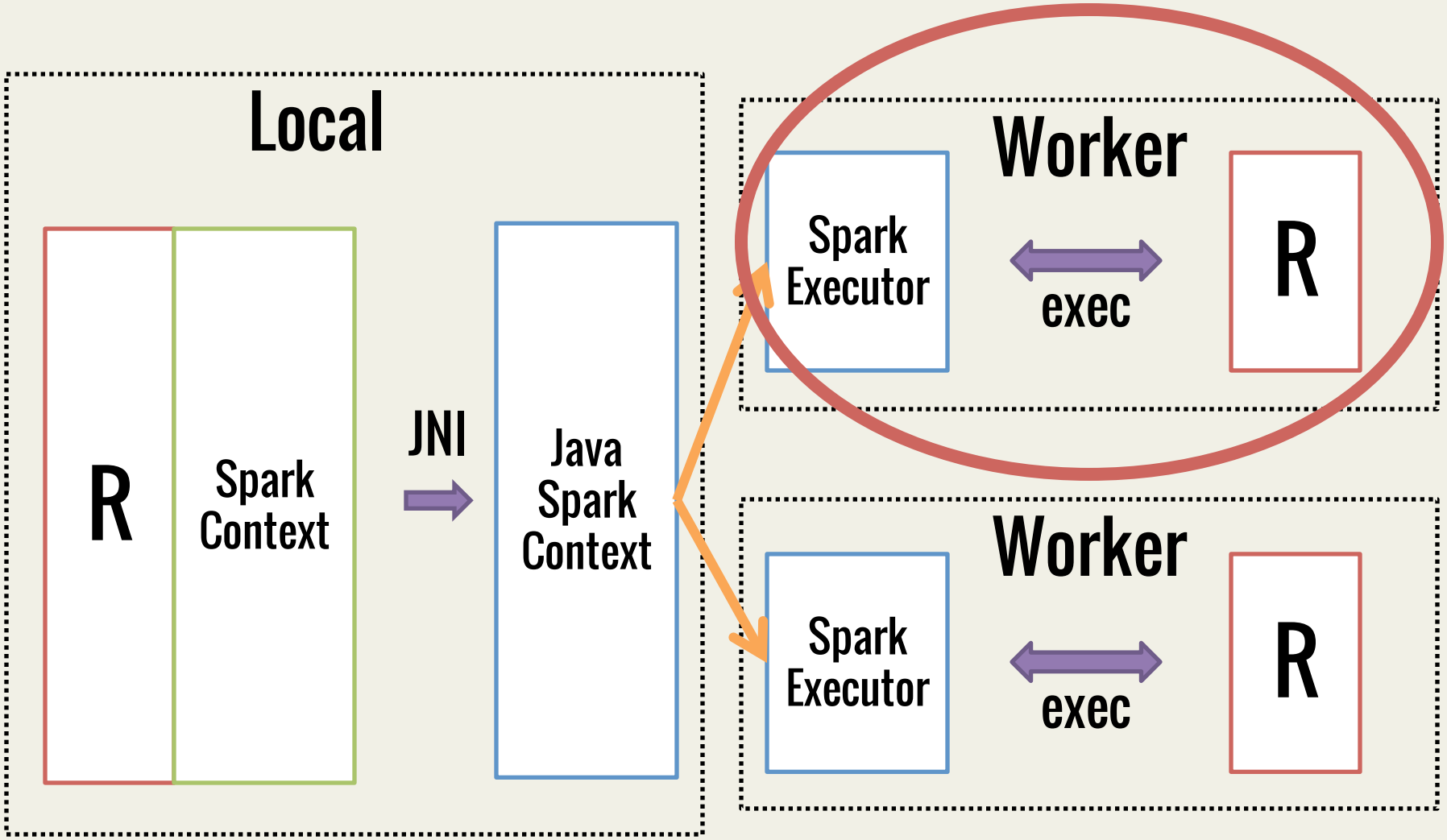
Save R Objects

Description

`save` writes an external representation of `R` objects to the specif
from the file at a later date by using the function `load` (or `data`

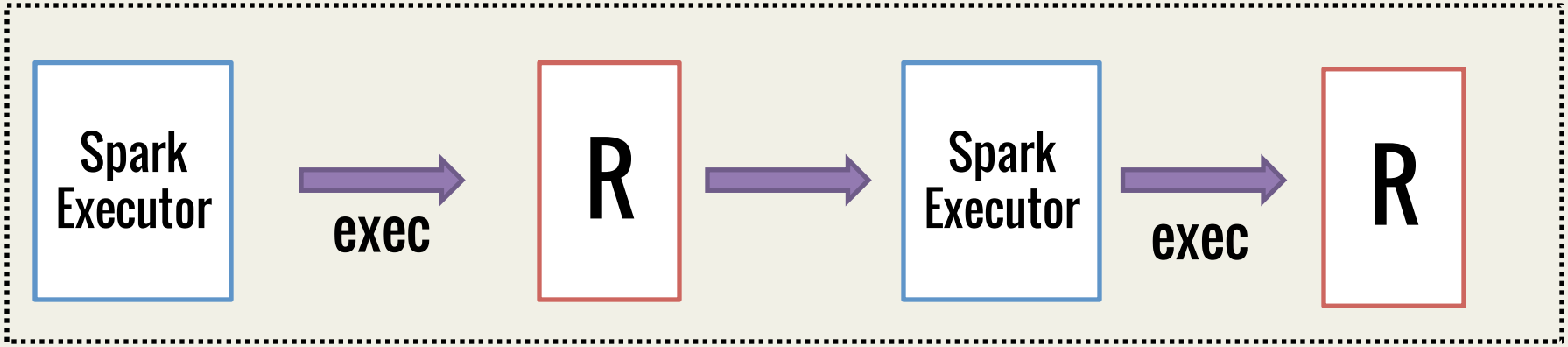
`save.image()` is just a short-cut for 'save my current workspac
`file = ".RData")`. It is also what happens with `q("yes")`.

Dataflow

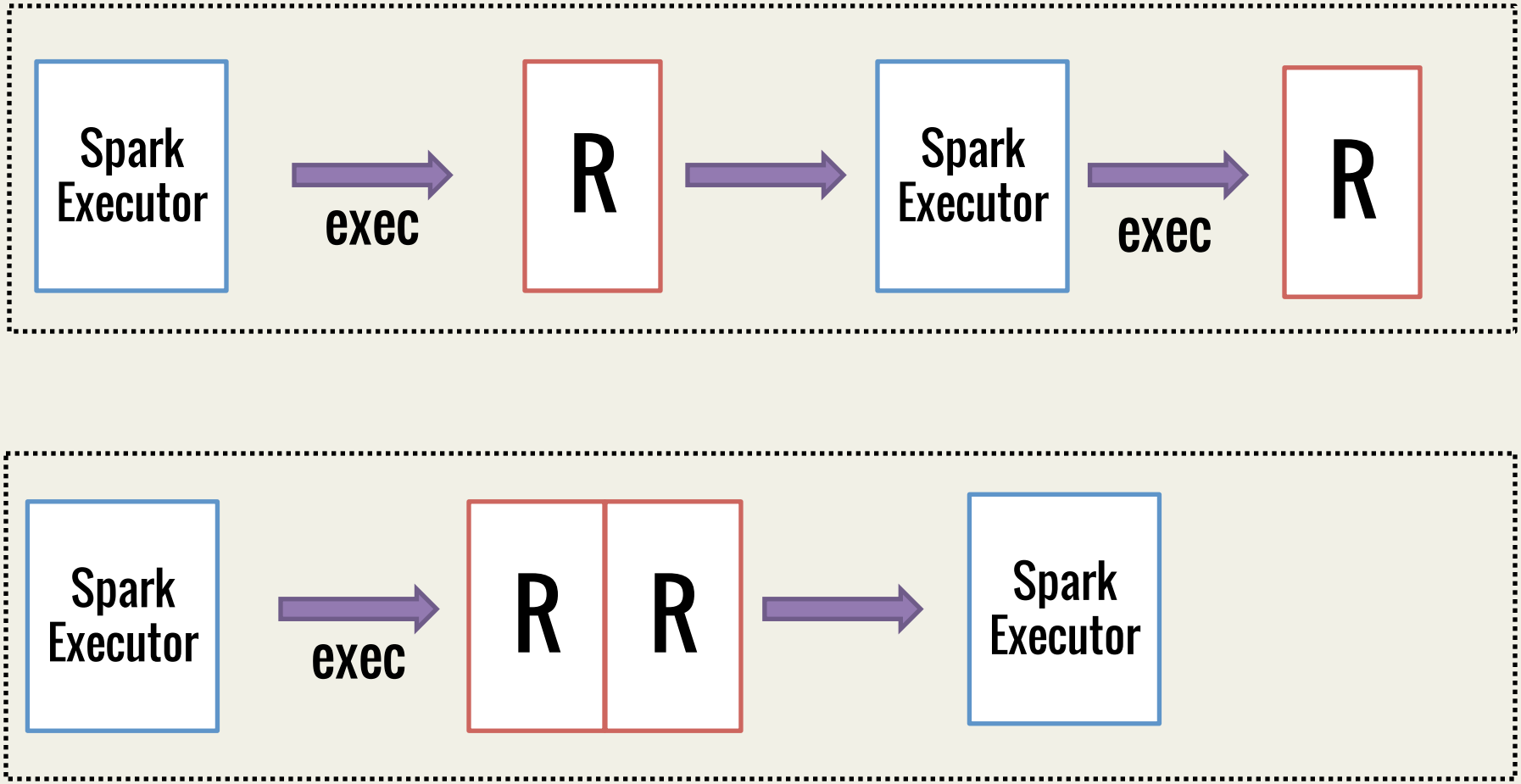

Local
R
Spark Context
JNI
Java Spark Context
Worker
Spark Executor
exec
R
Worker
Spark Executor
exec
R

Pipelined RDD

```
words <- flatMap(lines,…)
wordCount <- lapply(words,…)
```



Pipelined RDD



Alpha developer release

One line install !

```
install_github("amplab-extras/SparkR-pkg",
               subdir="pkg")
```

SparkR Implementation

Very similar to PySpark

Spark is easy to extend

- 292 lines of Scala code
- 1694 lines of R code
- 549 lines of **test** code in R

amplab-extras/SparkR-pkg

build passing

R frontend for Spark

Current | Build History | Pull Requests | Branch Summary

Build	34	Commit	c5bce07 (master)
State	Passed	Compare	aacd72657106...c5bce07ef517
Finished	23 days ago	Author	Zongheng Yang
Duration	9 min 37 sec	Committer	Zongheng Yang
Message	Merge pull request #30 from shivaram/string-tests		

Add tests for partitioning with string keys

Also on github

EC2 setup

All Spark examples
MNIST demo

Hadoop2, Maven build

In the Roadmap

DataFrame support using Catalyst

Calling MLLib from R

Daemon R processes

RDD → distributed lists

SparkR

Serialize closure

Re-use R packages

Combine scalability & utility

SparkR

https://github.com/amplab-extras/SparkR-pkg

Shivaram Venkataraman shivaram@cs.berkeley.edu

Spark User mailing list user@spark.apache.org

Example: Logistic Regression

```
pointsRDD <- textFile(sc, "hdfs://myfile")
weights <- runif(n=D, min = -1, max = 1)

# Logistic gradient
gradient <- function(partition) {
  X <- partition[,1]; Y <- partition[,-1]
  t(X) %*% (1/(1 + exp(-Y * (X %*% weights))) - 1) * Y
}
```

Example: Logistic Regression

```
pointsRDD <- textFile(sc, "hdfs://myfile")
weights <- runif(n=D, min = -1, max = 1)

# Logistic gradient
gradient <- function(partition) {
  X <- partition[,1]; Y <- partition[,-1]
  t(X) %*% (1/(1 + exp(-Y * (X %*% weights))) - 1) * Y
}

#Iterate
weights <- weights - reduce(
      lapplyPartition(pointsRDD, gradient), "+")
```

How does it work ?