RHive : Integrating R and Hive

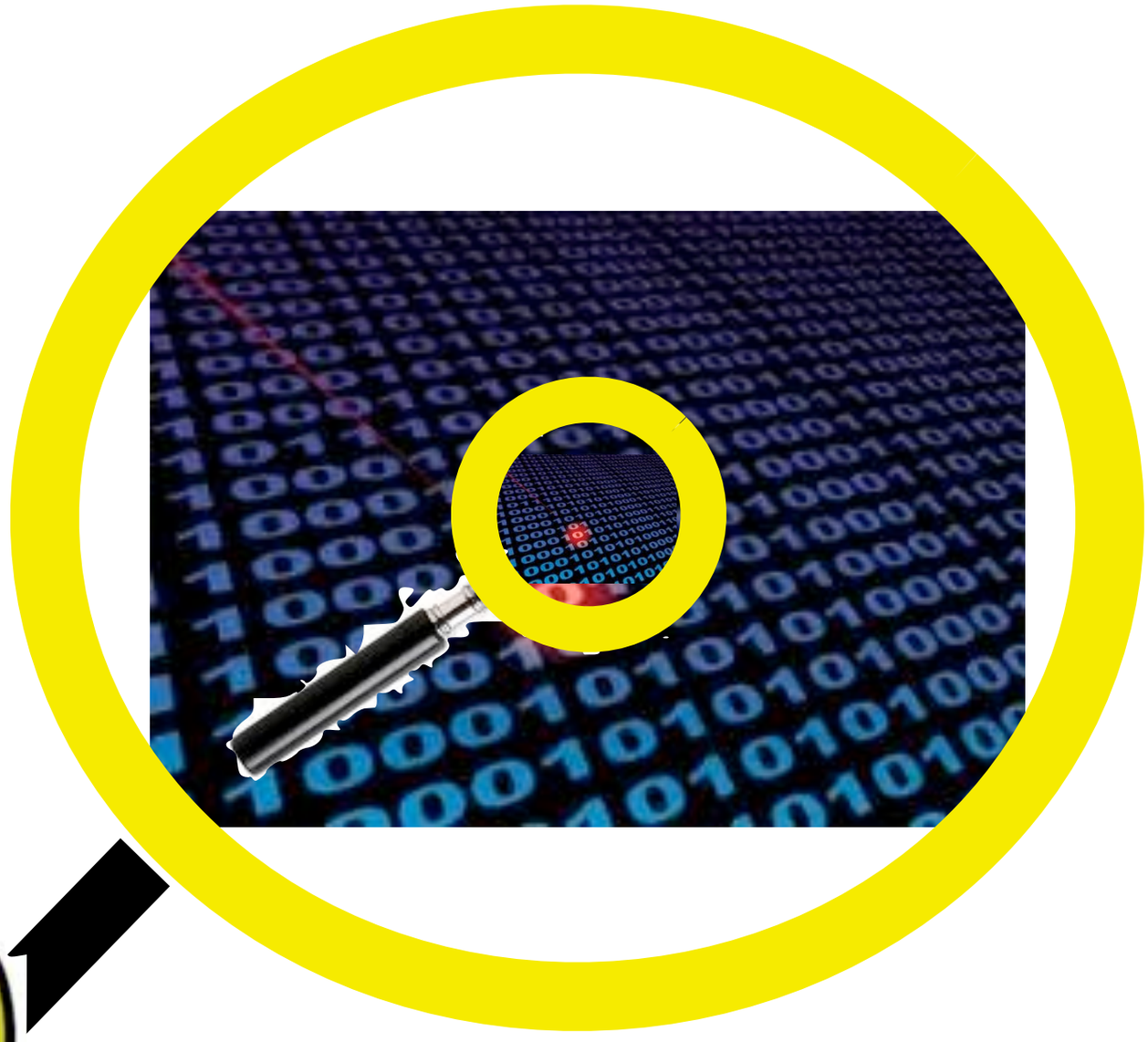
Introduction

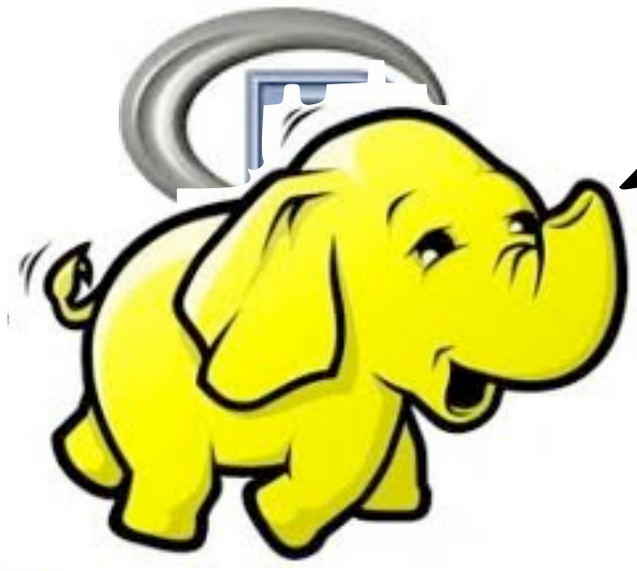
JunHo Cho

Data Analysis Platform Team

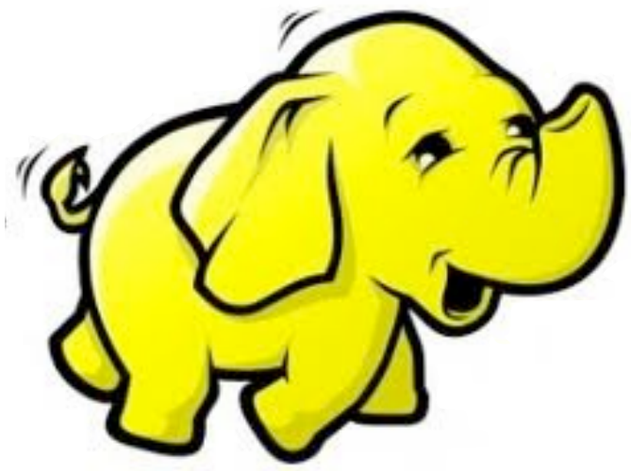
NEXR
TOWARD OPEN PLATFORM

Analysis of Data





Analysis of Data



MapReduce

SPSS

sas

CF Classifier Decision Tree

Recommendation

Clustering Graph

NEXR

RHive

Related Works

- RHIPE
- RHadoop
- hive (Hadoop InteractiVE)
- seuge

Must understand MapReduce

RHive is inspired by ...

- Many analysts have been used R for a long time
- Many analysts can use SQL language
- There are already a lot of statistical functions in R
- R needs a capability to analyze big data
- Hive supports SQL-like query language (HQL)
- Hive supports MapReduce to execute HQL

R is the best solution for familiarity
Hive is the best solution for capability

NexR
TOWARD OPEN PLATFORM

RHive

RHive Components

- Hadoop
  - store and analyze big data
- Hive
  - use HQL instead of MapReduce programming
- R
  - support friendly environment to analysts

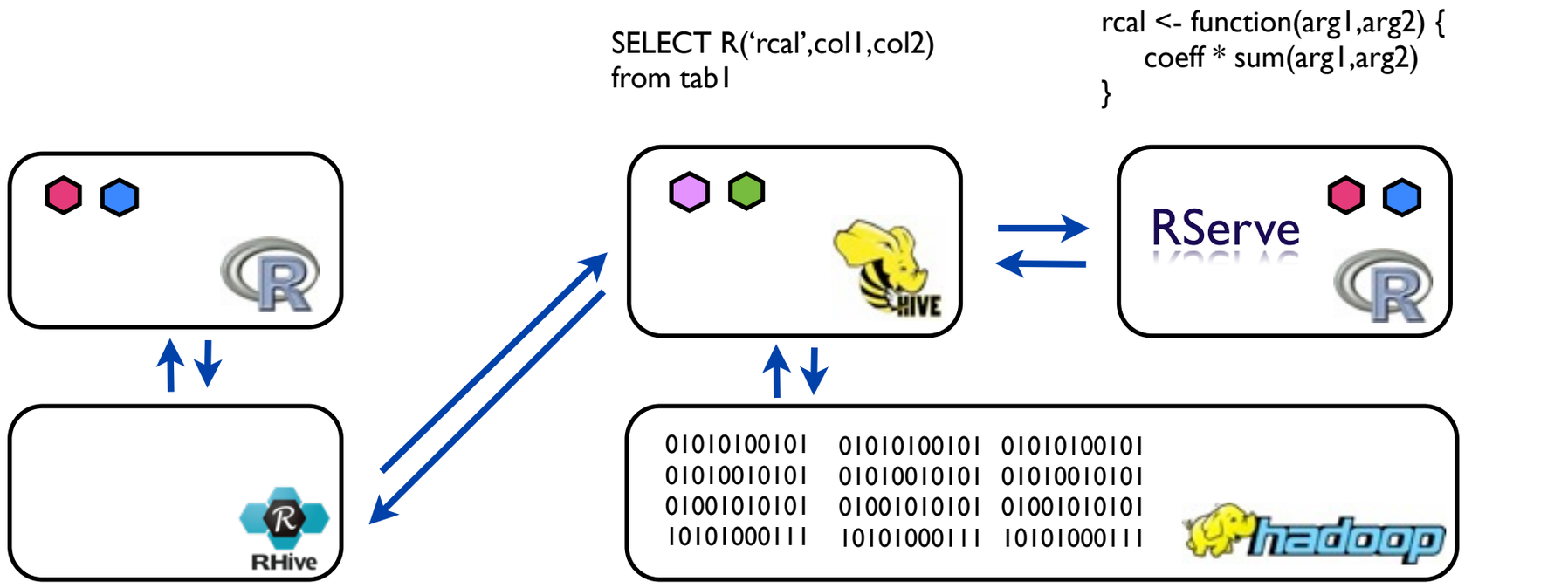
NEXR

RHive

RHive - Architecture

Execute R Function Objects and R Objects through Hive Query

Execute Hive Query through R



RHive API

- Extension R Functions
  - rhive.connect
  - rhive.query
  - rhive.assign
  - rhive.export
  - rhive.napply
  - rhive.sapply
  - rhive.aggregate
  - rhive.list.tables
  - rhive.load.table
  - rhive.desc.table

- Extension Hive Functions
  - RUDF
  - RUDAF
  - GenericUDTFExpand
  - GenericUDTFUnFold

NEXR TOWARD OPEN PLATFORM

RHive

RUDF - R User-defined Functions

SELECT R('*R function-name*',col1,col2,...,TYPE)

- UDF doesn't know return type until calling R function
  - TYPE : return type

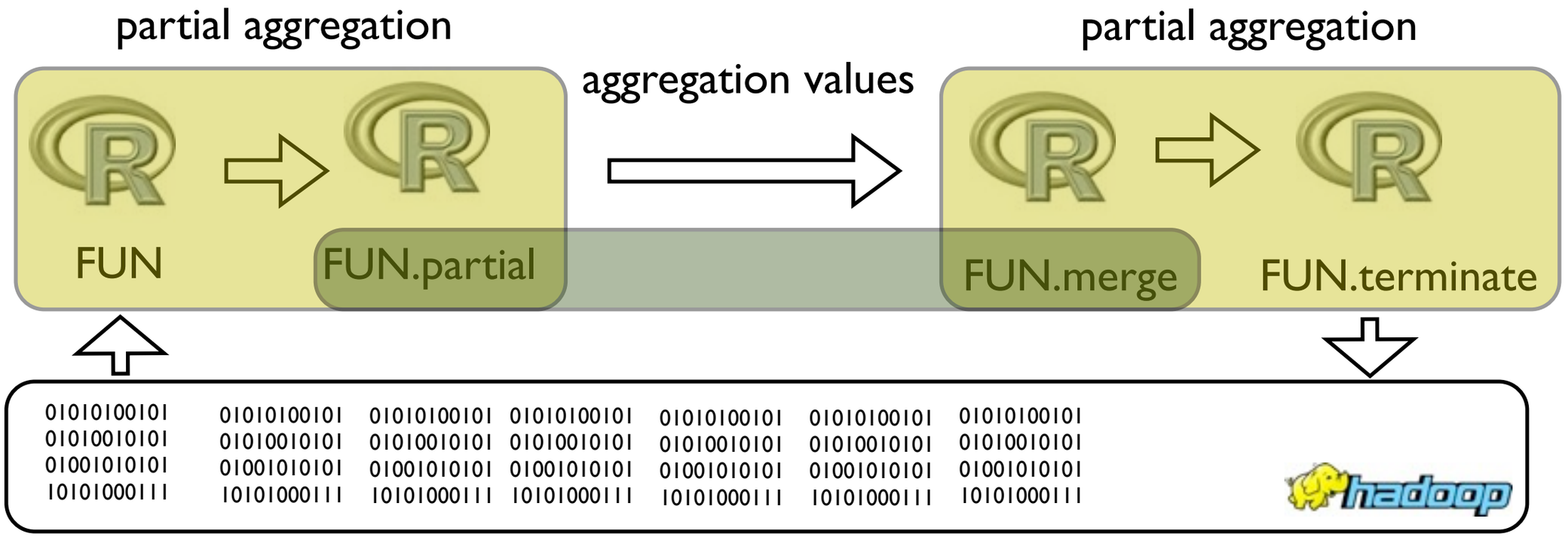
Example : R function which sums all passed columns

```
sumCols <- function(arg1,...) {
    sum(arg1,...)
}
rhive.assign('sumCols',sumCols)
rhive.exportAll('sumCols',hadoop-clusters)
result <- rhive.query("SELECT R('sumCols', col1, col2, col3, col4, 0.0) FROM tab")
plot(result)
```

RUDAF - R User-defined Aggregation Function

SELECT RA('*R function-name*',col1,col2,...)

- R can not manipulate large dataset
- Support UDAF's life cycle
  - iterate, partial-merge, merge, terminate
- Return type is only string delimited by ',' - "data1,data2,data3,..."



UDTF : unfold and expand

- RUDAF only returns string delimited by ‘,’
- Convert RUDAF’s result to R data.frame

unfold(string_value,type1,type2,...,delimiter)

expand(string_value,type,delimiter)

RA(‘newcenter’,...) return “num1,num2,num3” per cluster-key

select unfold(tb1.x,0.0,0.0,0.0,’,’) as (col1,col2,col3) from (select RA(‘newcenter’, attr1,attr2,attr3,attr4) as x from table group by cluster-key

NEXR

RHive

napply and sapply

rhive.napply(table-name,FUN,col1,...)

rhive.sapply(table-name,FUN,col1,...)

- napply : R apply function for Numeric type
- sapply : R apply function for String type

Example : R function which sums all passed columns

```
sumCols <- function(arg1,...) {
    sum(arg1,...)
}
result <- rhive.napply("tab", sumCols, col1, col2, col3, col4)
rhive.load.table(result)
```

NEXR TOWARD OPEN PLATFORM

RHive

napply

- ‘napply’ is similar to R apply function
- Store big result to HDFS as Hive table

```
rhive.napply <- function(tablename, FUN, col = NULL, ...) {
    if(is.null(col))
        cols <- ""
    else
        cols <- paste(",",col)

    for(element in c(...)) {
         cols <- paste(cols,",",element)
    }

    exportname <- paste(tablename,"_sapply",as.integer(Sys.time()),sep="")

    rhive.assign(exportname,FUN)
    rhive.exportAll(exportname)

    tmptable <- paste(exportname,"_table")
    rhive.query(
          paste("CREATE TABLE ", tmptable," AS SELECT ","R('",exportname,"'",cols,",0.0) FROM ",tablename,sep=""))

    tmptable
}
```

aggregate

rhive.aggregate(table-name,hive-FUN,...,goups)

- RHive aggregation function to aggregate data stored in HDFS using HIVE Function

Example : Aggregate using SUM (Hive aggregation function)

*result <- **rhive.aggregate**("emp","SUM", sal,groups="deptno")*
rhive.load.table(result)

Examples - predict flight delay

```
library(RHive)

rhive.connect()
```

- Retrieve training set from large dataset stored in HDFS

```
train <- rhive.query("SELECT dayofweek,arrdelay,distance FROM airlines TABLESAMPLE(BUCKET 1 OUT OF 10000 ON rand())

train$arrdelay <- as.numeric(train$arrdelay)

train$distance <- as.numeric(train$distance)

train <- train[!(is.na(train$arrdelay) | is.na(train$distance)),]

model <- lm(arrdelay ~ distance + dayofweek,data=train)
```

Native R code

- Export R object data

```
rhive.assign("model", model)
```

- Analyze big data using model calculated by R

```
predict_table <- rhive.napply("airlines",function(arg1,arg2,arg3) {

        if(is.null(arg1) | is.null(arg2) | is.null(arg3))  return(0.0)

        res <- predict.lm(model, data.frame(dayofweek=arg1,arrdelay=arg2,distance=arg3))

        return(as.numeric(res)) }, 'dayofweek','arrdelay','distance')
```

HiveQuery + R code

NexR

RHive

DEMO

Conclusion

- RHive supports HQL, not MapReduce model style
- RHive allows analytics to do everything in R console
- RHive interacts R data and HDFS data

- Future & Current Works
  - Integrate Hadoop HDFS
  - Support Transform/Map-Reduce Scripts
  - Distributed Rserve
  - Support more R style API
  - Support machine learning algorithms (k-means, classifier, ...)

NexR

RHive

Cooperators

- JunHo Cho
- Seonghak Hong
- Choonghyun Ryu

YOU!

How to join RHive project

- Logo

- github (https://github.com/nexr/RHive)
- CRAN (http://cran.r-project.org/web/packages/RHive)
- Welcome to join RHive project

References

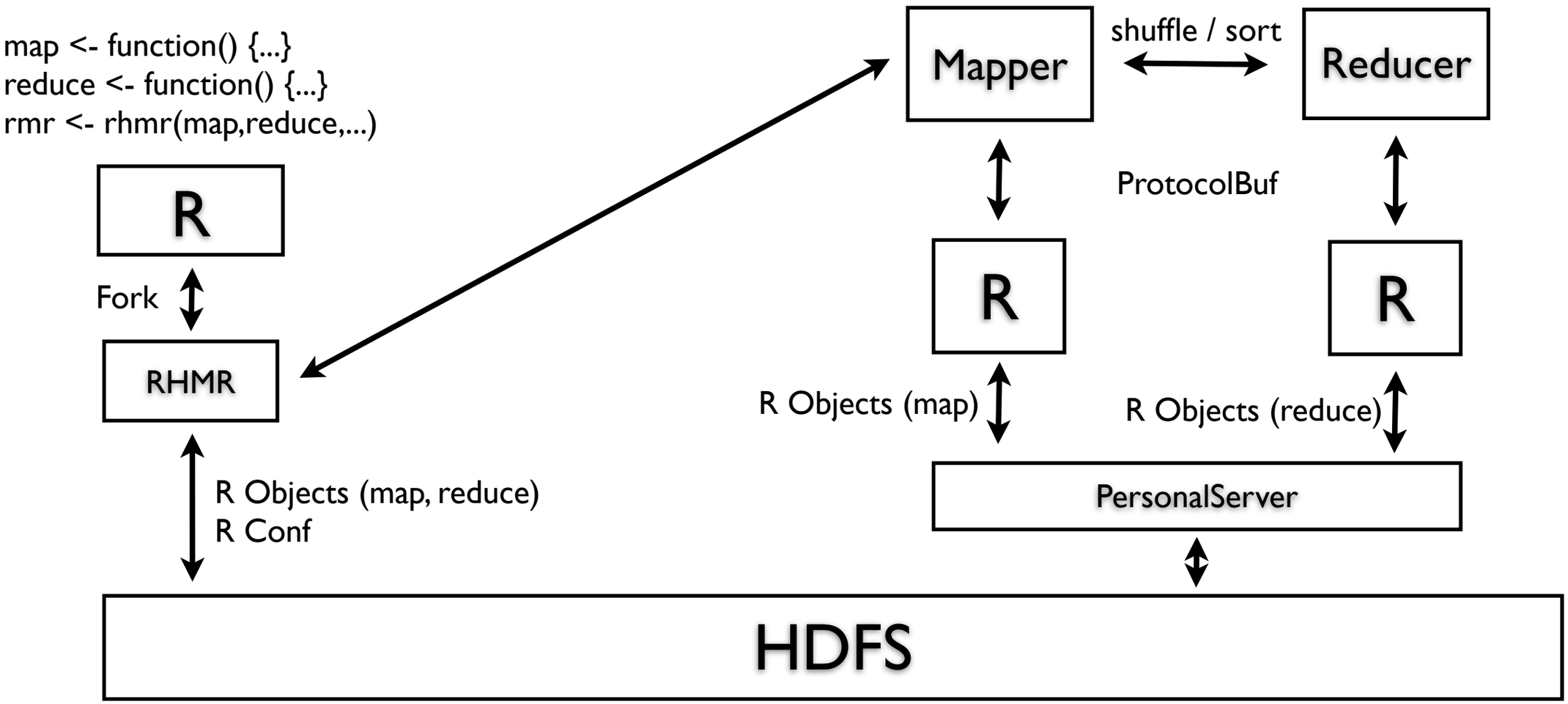
- Recardo (https://mpi-inf.mpg.de/~rgemulla/publications/das10ricardo.pdf)
- RHIPE (http://ml.stat.purdue.edu/rhipe)
- Hive (http://hive.apache.org)
- Parallels R by Q. Ethan McCallum and Stephen Weston

jun.cho@nexr.com

Appendix

RHIPE

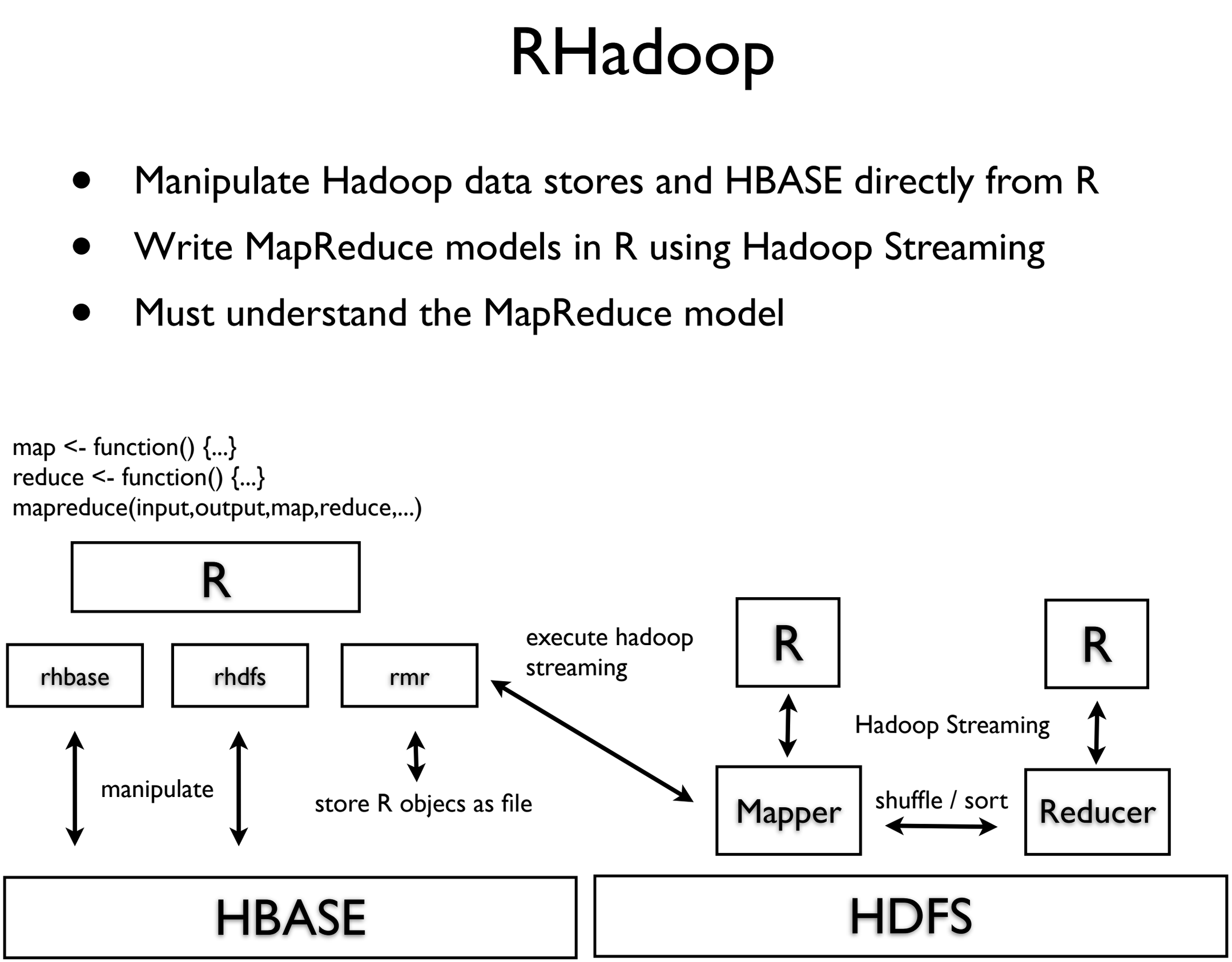
- the R and Hadoop Integrated Processing Environment
- Must understand the MapReduce model



RHadoop

- Manipulate Hadoop data stores and HBASE directly from R
- Write MapReduce models in R using Hadoop Streaming
- Must understand the MapReduce model

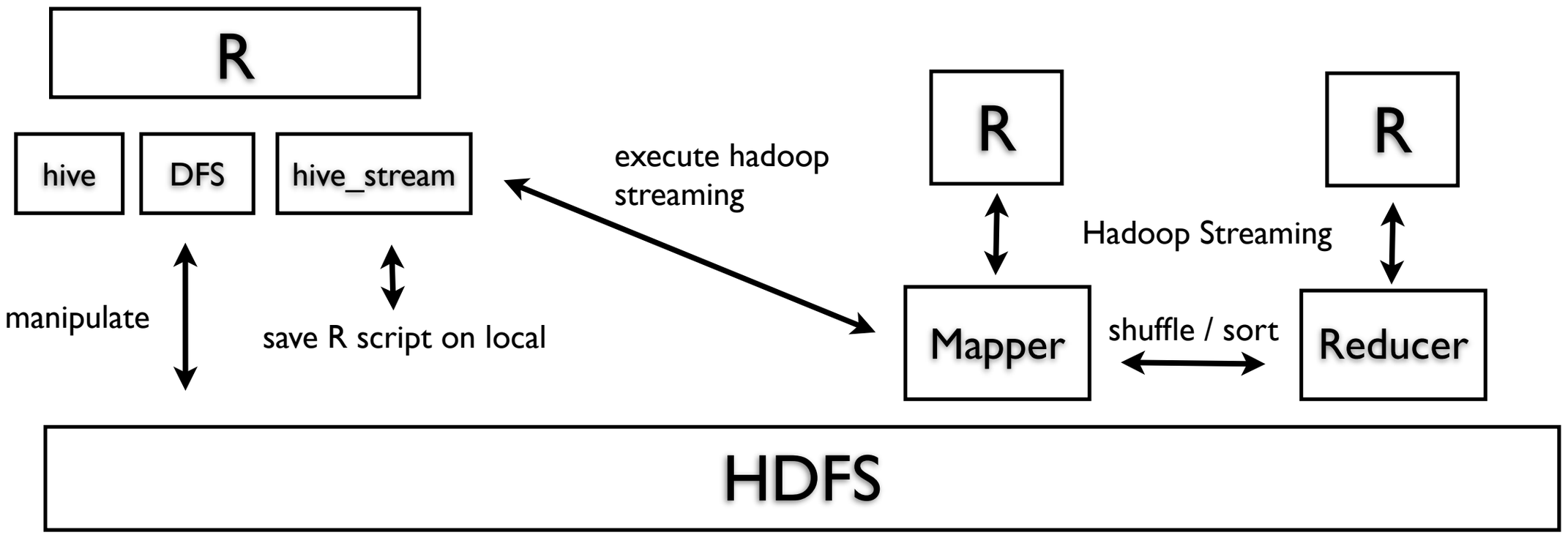
```
map <- function() {...}
reduce <- function() {...}
mapreduce(input,output,map,reduce,...)
```



hive(Hadoop InteractiVE)

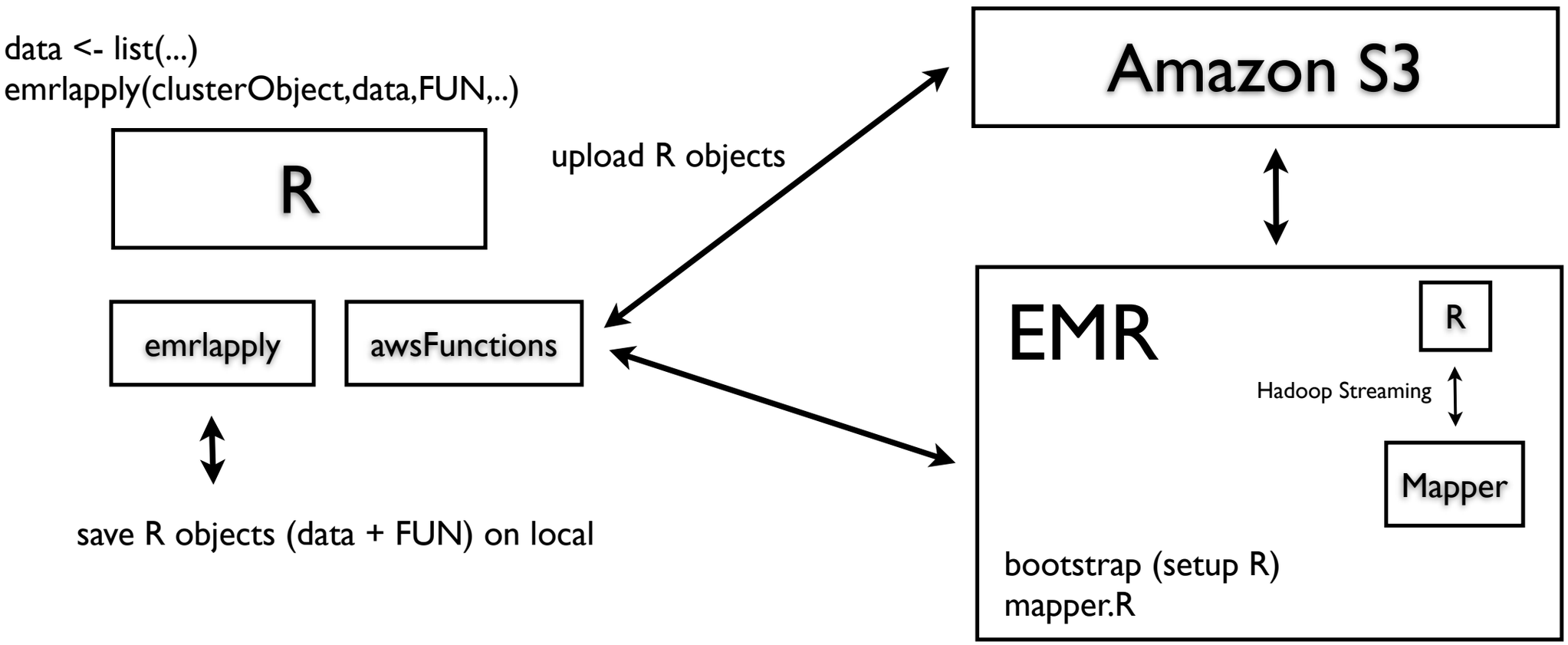
- R extension facilitating distributed computing via the MapReduce paradigm
- Provide an interface to Hadoop, HDFS and Hadoop Streaming
- Must understand the MapReduce model

```
map <- function() {...}
reduce <- function() {...}
hive_stream(map,reduce,...)
```



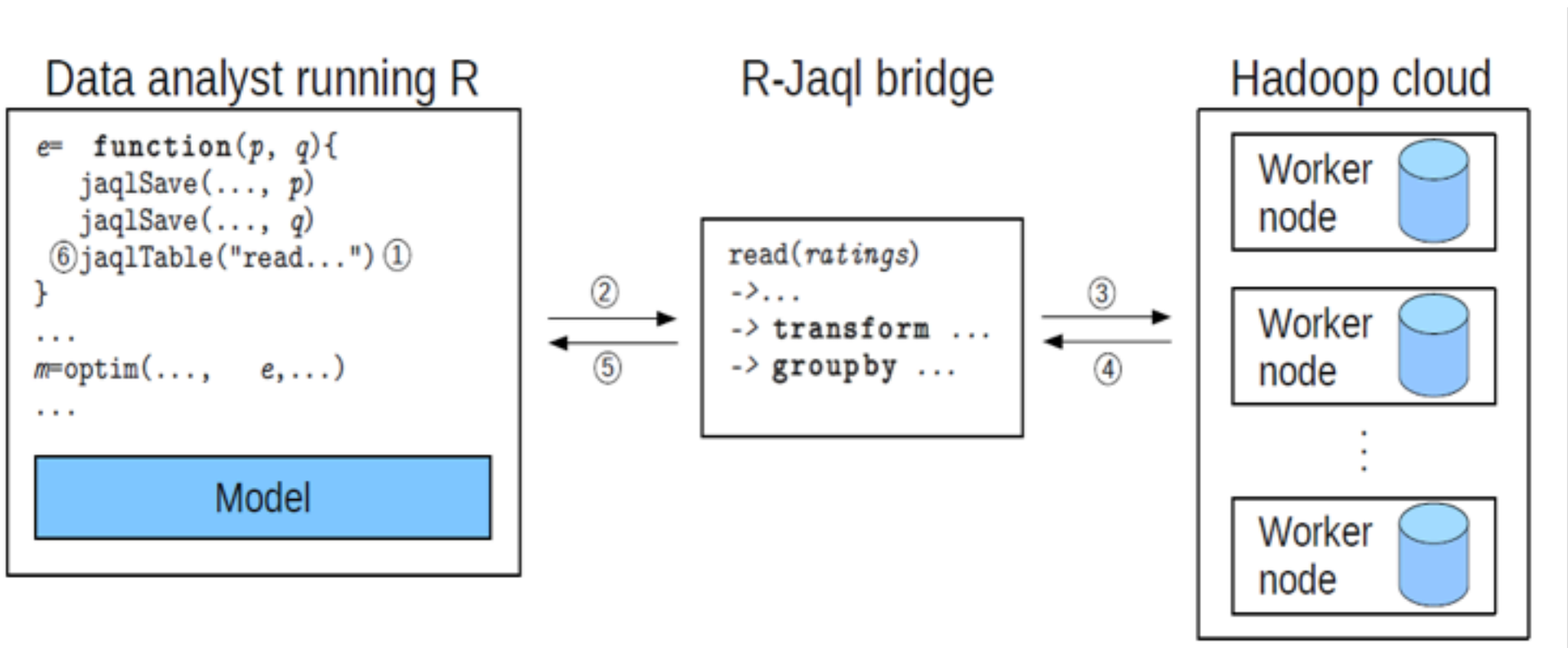
seuge

- Simple parallel processing with a fast and easy setup on Amazon's WS.
- Parallel lapply function for the EMR engine using Hadoop streaming.
- Does not support MapReduce model but only Map model.



Ricardo

- Integrate R and Jaql (JSON Query Language)
- Must know how to use uncommon query, Jaql
- Not open-source



Ref : Ricardo-SIGMOD10