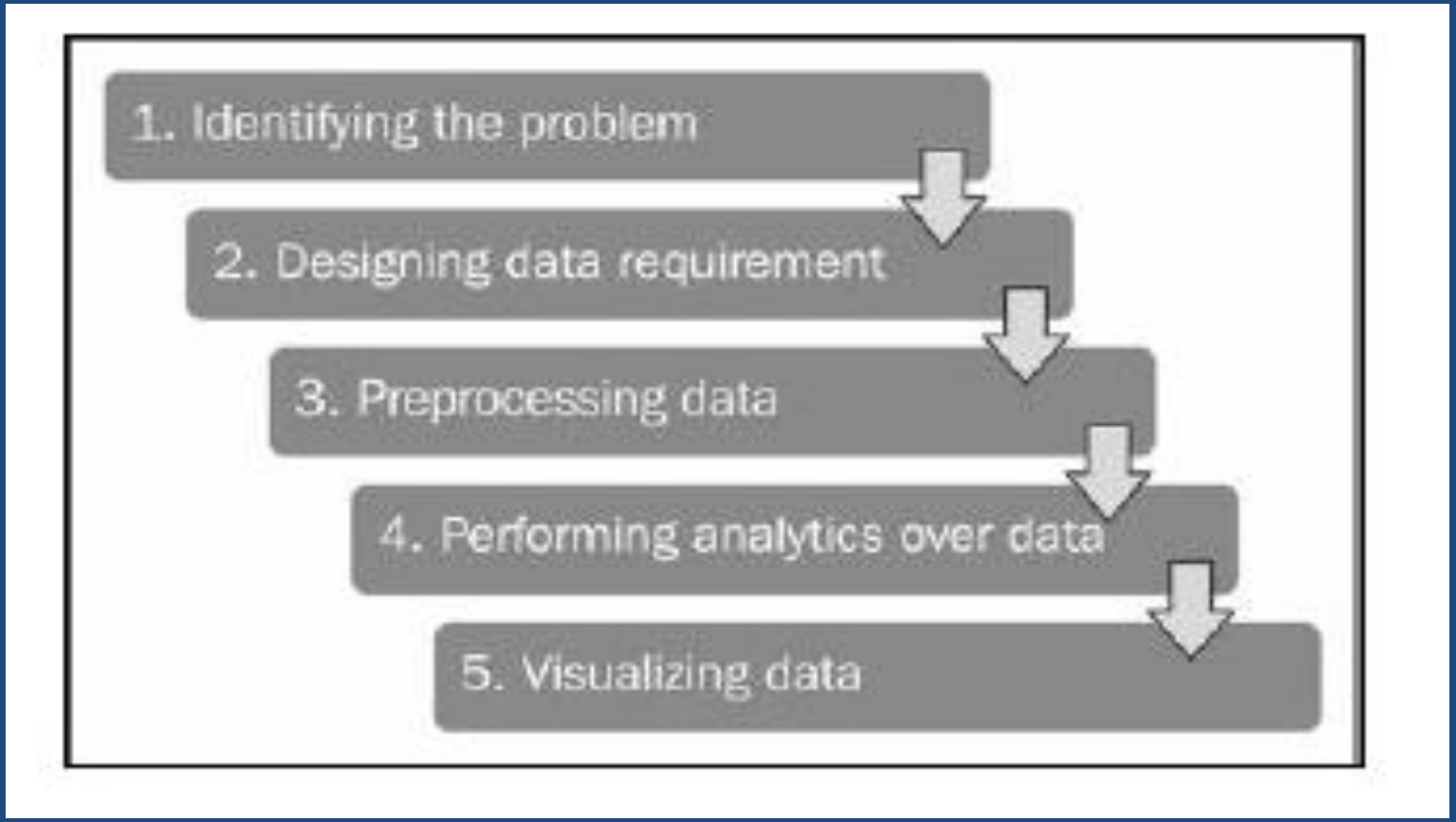
Data Analytics Project Life Cyle

Data analytics project life cycle stages



Identifying the problem

-Web Analytics

- Let's assume that we have a large e-commerce website, and we want to know how to increase the business.
- We can identify the important pages of our website by categorizing them as per popularity into high, medium, and low.
- Based on these popular pages, their types, their traffic sources, and their content:
  - we will be able to decide the roadmap to improve business by improving web traffic, as well as content.

Designing data requirement

- To perform the data analytics for a specific problem, it needs datasets from related domains.
- Based on the domain and problem specification, the data source can be decided
- And based on the problem definition; the data attributes of these datasets can be defined.
- **For example:** Social media analytics (problem specification)
  - we use the data source as Facebook or Twitter.
  - For identifying the user characteristics, we need user profile information, likes, and posts as data attributes.

Preprocessing data

- In data analytics, we do not use the same data sources, data attributes, data tools, and algorithms all the time as all of them will not use data in the same format.
- This leads to the performance of data operations:
  - data cleansing, data aggregation,
  - data transformation, data augmentation, data sorting, and data formatting
  - to provide the data in a supported format to all the data tools as well as algorithms that will be used in the data analytics.

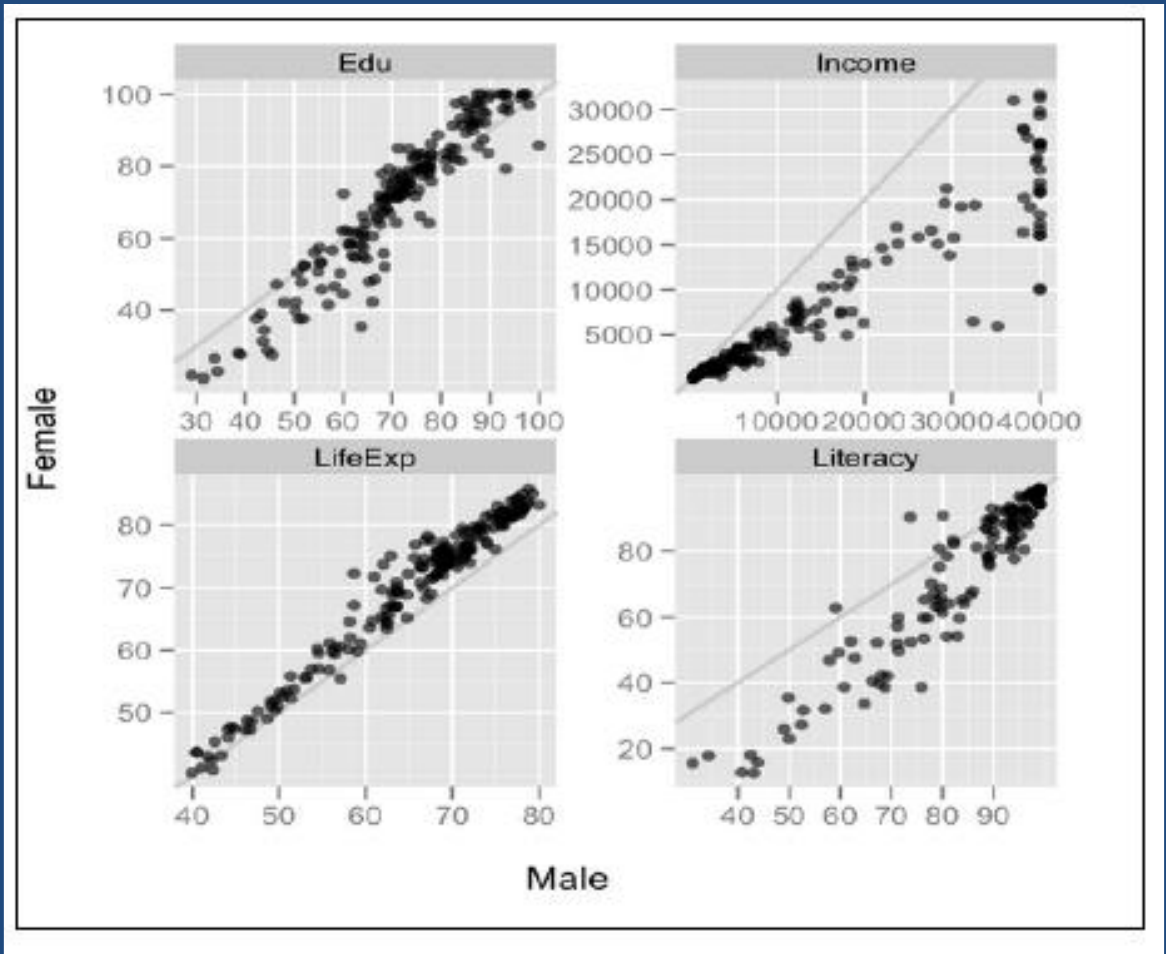
Performing analytics over data

- The data analytics operations are performed for discovering meaningful information from data to take better decisions towards business with data mining concepts.
  - descriptive or predictive analytics
  - regression, classification, clustering, and model-based recommendation.
- For Big Data, the same algorithms can be translated to MapReduce algorithms for running them on Hadoop clusters
  - by translating their data analytics logic to the MapReduce job which is to be run over Hadoop clusters.
  - These models need to be further evaluated as well as improved by various evaluation stages of machine learning concepts.
  - Improved or optimized algorithms can provide better insights.

Visualizing data

- Data visualization is used for displaying the output of data analytics.
- Visualization is an interactive way to represent the data insights.
- This can be done with various data visualization softwares / R packages.
- R packages for the visualization of datasets:
  - ggplot2: This is an implementation of the Grammar of Graphics
  - rCharts: This is an R package to create, customize, and publish interactive JavaScript visualizations from R by using a familiar lattice-style plotting interface

ggplot()


Edu
Income
LifeExp
Literacy
Female
Male
100
80
60
40
30000
25000
20000
15000
10000
5000
30 40 50 60 70 80 90 100
10000 20000 30000 40000
80
70
60
50
40 50 60 70 80
80
60
40
20
40 50 60 70 80 90

rcharts:

Interactive animated Dashboards

RHadoop Architecture

What is R? When do we need it?

- **Open-source** stat package with **visualization**

  Vibrant **community support**.

  One-line calculations galore!

  Steep learning curve but worth it!

- **Insight** into statistical properties and trends…

  or for **machine learning** purposes…

  or Big Data to be **understood** well

What is Hadoop?

Distributed file system (**HDFS**) and parallel processing framework.

Uses **MapReduce** programming model as the core.

Provides **fault tolerant** and **scalable storage** of very large datasets across machines in a cluster.

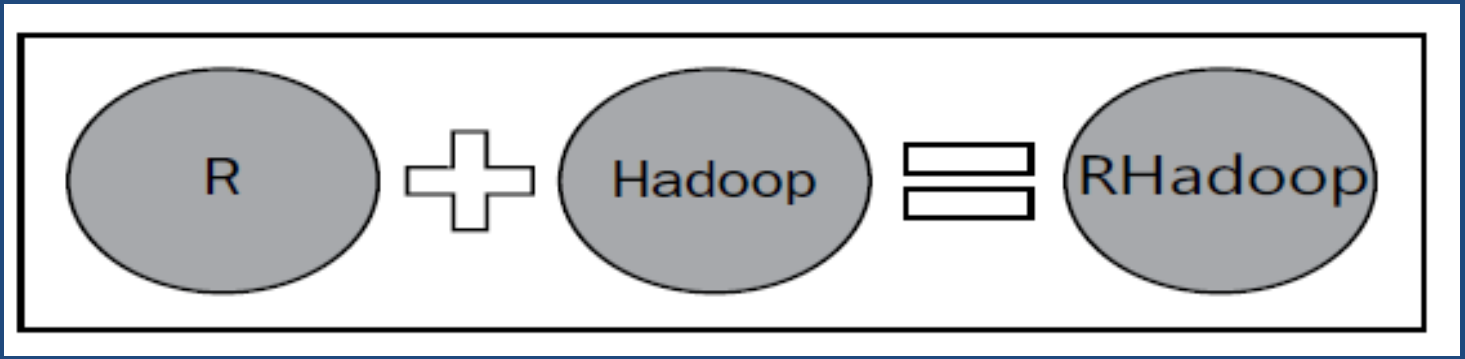
Why Interface Hadoop and R at cluster level?

- R only works on:
  - Data is in-memory, stand alone. Single-threaded, mostly.
  - “Multicore”package in R help here.
- HDFS can be “the” data and analytic store.

- Interfacing with Hadoop brings parallel processing capability to R environment.

How do we interface Hadoop and R, at cluster level?

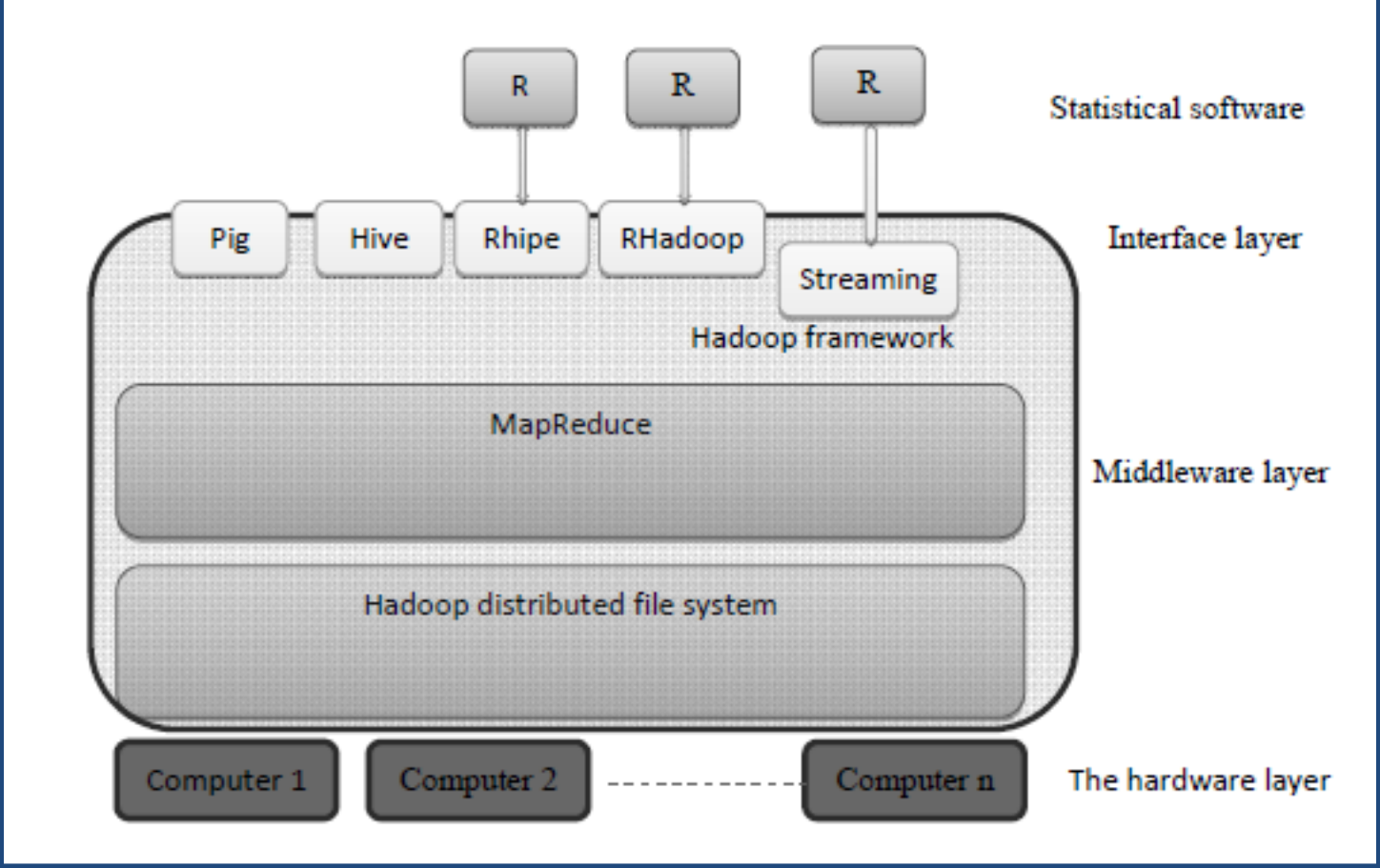
Solution: R + Hadoop



Need middleware for R and Hadoop.

- RHadoop,
- RHIPE, and
- Hadoop streaming

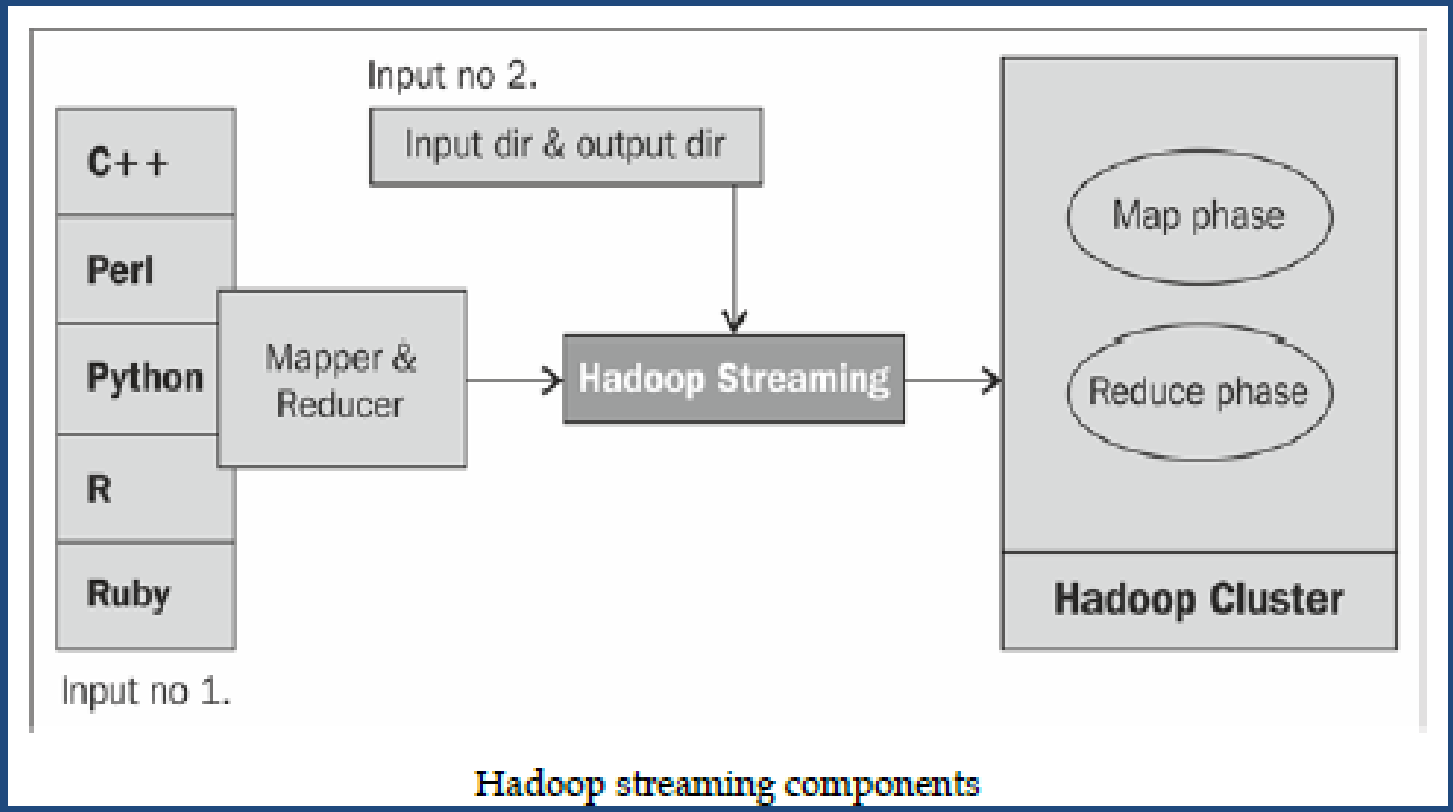
R and Hadoop


R
R
R
Statistical software
Pig
Hive
Rhipe
RHadoop
Streaming
Interface layer
Hadoop framework
MapReduce
Middleware layer
Hadoop distributed file system
Computer 1
Computer 2
Computer n
The hardware layer

R and Streaming

- Streaming is a technology **integrated in the Hadoop distribution** that allows users to run Map/Reduce jobs with any script or executable that reads data from standard input and writes the results to standard output as the mapper or reducer.
- This means that we can use Streaming together with R scripts in the map and/or reduce phase since R can read/write data from/to standard input.
- In this approach there is **no client-side integration with R** because the user will use the Hadoop command line to launch the Streaming jobs with the arguments specifying the mapper and reducer R scripts

Hadoop Streaming



Hadoop streaming components

Hadoop Steaming..

- Hadoop streaming is a Hadoop utility for running the Hadoop MapReduce job with executable scripts such as Mapper and Reducer.
- This is similar to the pipe operation in Linux. With this, the text input file is printed on stream (stdin), which is provided as an input to Mapper
- And the output (stdout) of Mapper is provided as an input to Reducer
- Finally, Reducer writes the output to the HDFS directory.

An example of a map-reduce task with R and Hadoop integrated by Streaming framework

- bin/hadoop command [generic Options] [streaming Options]

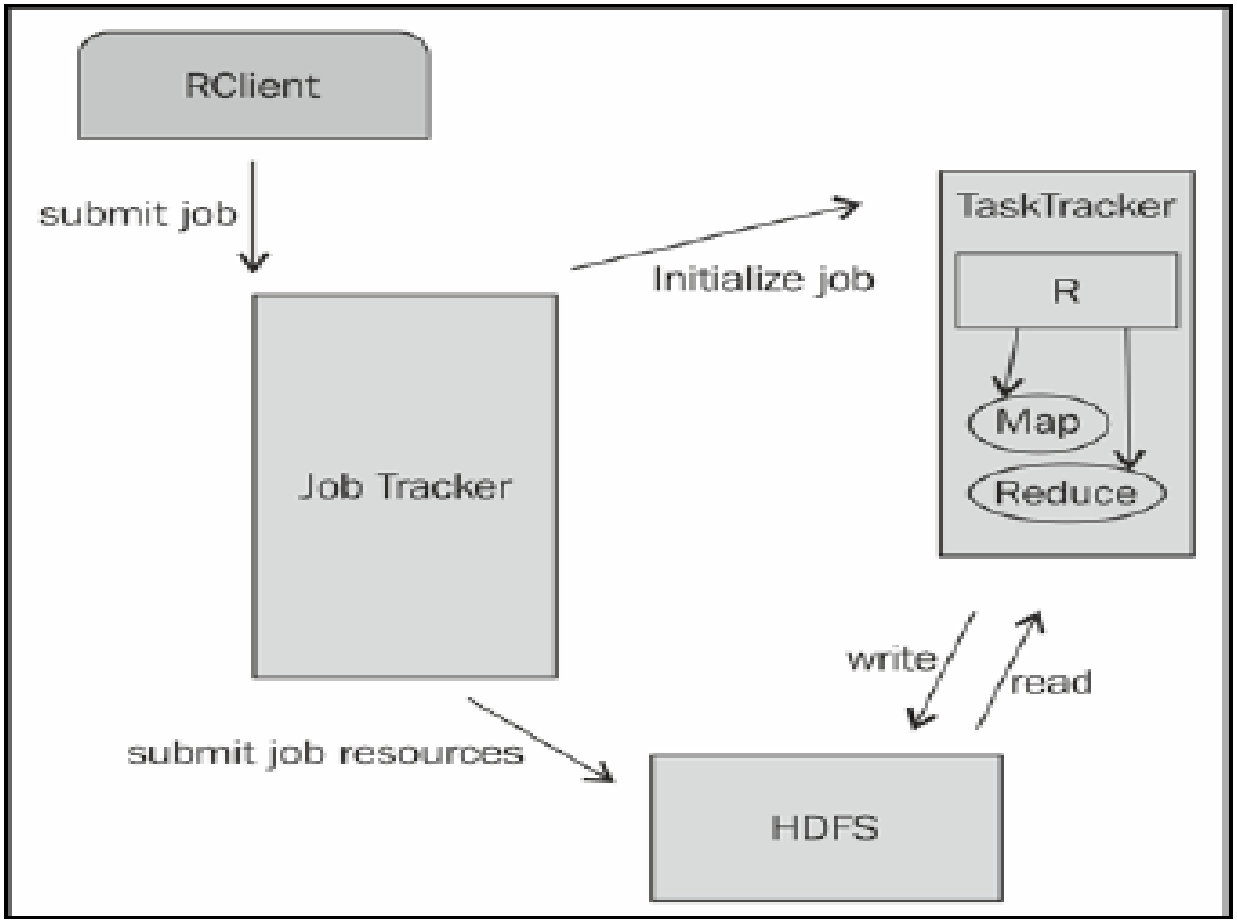
```
${HADOOP_HOME}/bin/hadoop \
        jar $HADOOP_HOME/contrib/*.jar \          Line 1
        -input /app/haadoop/input \               Line 2
        -output /app/haadoop/output \             Line 3
        -file /usr/local/hadoop/code_mapper.R \   Line 4
        -mapper code_mapper.R \                   Line 5
        -file /usr/local/hadoop/code_reducer.R \  Line 6
        -reducer code_reducer.R                   Line 7
```

- Line 1: This is used to specify the Hadoop jar files (setting up the classpathf or the Hadoop jar)
- Line 2: This is used for specifying the input directory of HDFS
- Line 3: This is used for specifying the output directory of HDFS
- Line 4: This is used for making a file available to a local machine
- Line 5: This is used to define the available R file as Mapper
- Line 6: This is used for making a file available to a local machine
- Line 7: This is used to define the available R file as Reducer

RHIPE

- It means "in a moment" in Greek and is a merger of R and Hadoop.
- It was first developed by *Saptarshi Guha* for his PhD thesis in the Department of Statistics at Purdue University in 2012.
- Currently this is carried out by the Department of Statistics team at Purdue University and other active Google discussion groups.
- The RHIPE package uses the **Divide and Recombine** technique to perform data analytics over Big Data

RHIPE Architecture



Components of RHIPE

Compenents of RHIPE

- **RClient:** RClient is an R application that calls the JobTracker to execute the job with an indication of several MapReduce job resources such as Mapper, Reducer, input format, output format, input file, output file, and other several parameters that can handle the MapReduce jobs with RClient.
- **JobTracker:** A JobTracker is the master node of the Hadoop MapReduce operations for initializing and monitoring the MapReduce jobs over the Hadoop cluster
- **TaskTracker**: TaskTracker is a slave node in the Hadoop cluster. It executes the MapReduce jobs as per the orders given by JobTracker, retrieve the input data chunks, and run R-specific Mapper and Reducer over it. Finally, the output will be written on the HDFS directory.
- **HDFS**: HDFS is a filesystem distributed over Hadoop clusters with several data nodes. It provides data services for various data operations.

Structure of Rscript using RHIPE

```
1 library(Rhipe)
2 rhinit(TRUE, TRUE);

3 map<-expression ( {lapply
        (map.values,
        function(mapper)…)})

4 reduce<-expression(
5 pre = {…},
6 reduce = {…},
7 post = {…},
8 )
```

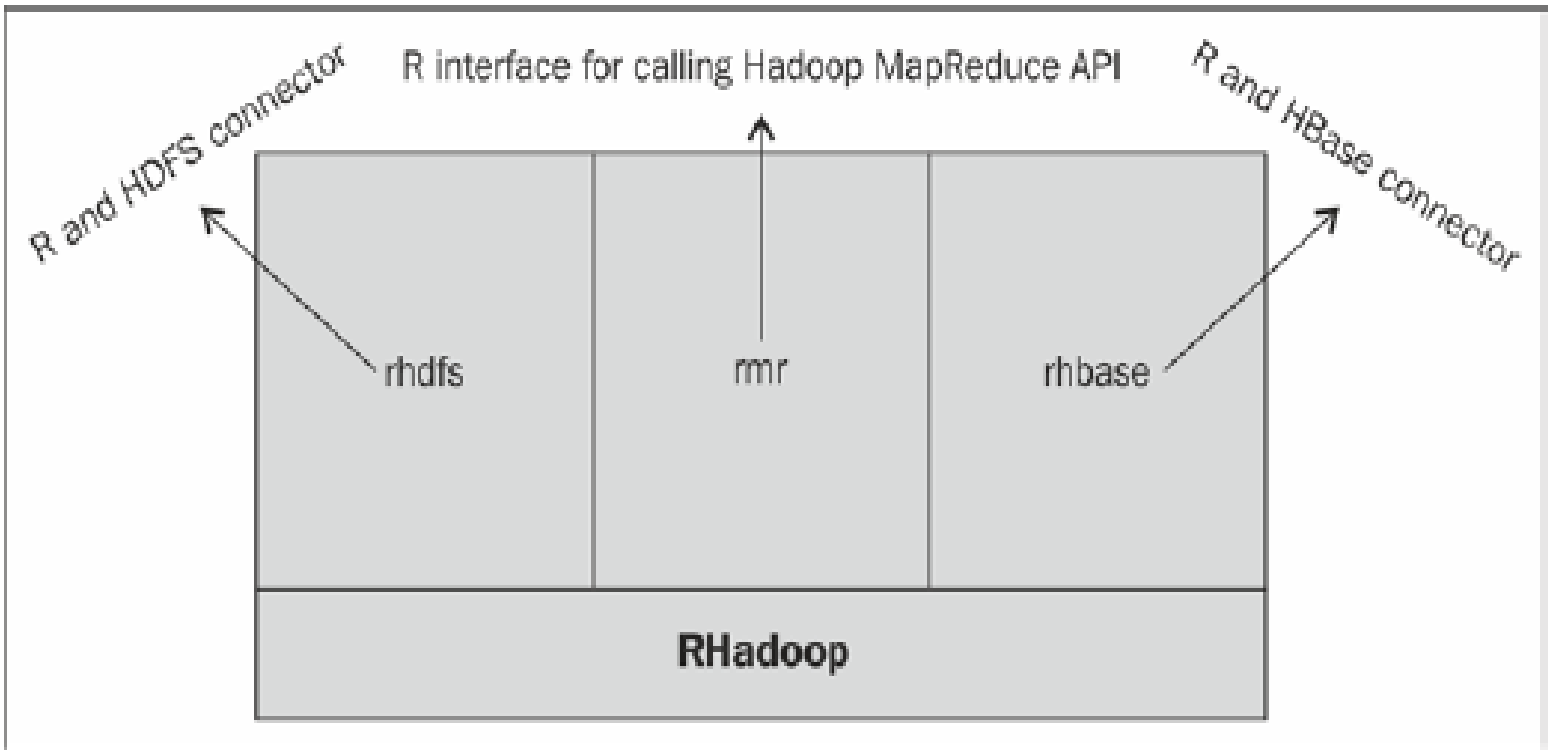
```
9 Job1 <- rhwatch(
10 map=map, reduce=reduce,
11 input=inputPath,
12 output=outputPath,
13 inout=c('text', 'text'),
14 jobname='a job name'))

15 Output_data <- rhread(Job1)
```

- Line 1: loads the Rhipe library into memory
- and initializing the Rhipe library (line 2).
- Line 3 defines the map expression to be executed by the Map task.
- Lines 4 to 8 define the reduce expression consisting in three callbacks.
  - The pre block of instructions (line 5) is called for each unique map output key before these values being sent to the reduce block.
  - The reduce block (line 6) is called then with a vector of values as argument and in the end,
  - the post block (line 7) is called to emit the output (key, value) pair.
- Line 9 shows the driver method that execute MapReduce job
- Line 15 for reading job output data from HDFS

RHadoop

- It was developed by Revolution Analytics, which is the leading commercial provider of software based on R.
- RHadoop is available with three main R packages:
  - rhdfs,
  - rmr, and
  - rhbase

Rhadoop Ecosystem



RHadoop Ecosystem

- **rhdfs** is an R interface for providing the HDFS usability from the R console.
  - Basically, rhdfs package calls the HDFS API in backend to operate data sources stored on HDFS.
- **rmr** is an R interface for providing Hadoop MapReduce facility inside the R environment.
  - The R programmer needs to just divide their application logic into the map and reduce phases and submit it with the rmr methods.
  - After that, rmr calls the Hadoop streaming MapReduce API with several job parameters as input directory, output directory, mapper, reducer, and so on, to perform the R MapReduce job over Hadoop cluster.
- **rhbase** is an R interface for operating the Hadoop HBase data source stored at the distributed network via a Thrift server.
  - The rhbase package is designed with several methods for initialization and read/write and table manipulation operations.

RHadoop Function Reference: hdfs package

Initialization

hdfs.init: This is used to initialize the rhdfs package.

Its syntax is hdfs.init().

hdfs.defaults: This is used to retrieve and set the rhdfs defaults.

Its syntax is hdfs.defaults().

RHadoop Function Reference: hdfs package

File manipulation

hdfs.put: This is used to copy files from the local filesystem to the HDFS filesystem.

hdfs.put('/usr/local/hadoop/README.txt','/RHadoop/1/')

hdfs.get(','/RHadoop/1/‘ , ‘/usr/local/hadoop/README.txt’)

hdfs.copy: This is used to copy files from one HDFS directory to another HDFS directory.

hdfs.copy('/RHadoop/1/','/RHadoop/2/')

hdfs.move: This is used to move a file from one HDFS directory to another HDFS directory.

hdfs.move('/RHadoop/1/README.txt','/RHadoop/2/')

RHadoop Function Reference: hdfs package

File manipulation

hdfs.rename: This is used to rename the file stored at HDFS from R.

```
hdfs.rename('/RHadoop/README.txt','/RHadoop/README1.txt')
```

hdfs.delete: This is used to delete the HDFS file from R.

```
hdfs.delete("/RHadoop")
```

hdfs.rm: This is used to delete the HDFS directory from R.

```
hdfs.rm("/RHadoop")
```

hdfs.chmod: This is used to change permissions of some files.

```
hdfs.chmod('/RHadoop', permissions= '777')
```

RHadoop Function Reference: hdfs package

Directory operation:

hdfs.dircreate or hdfs.mkdir: Both these functions will be used for creating a directory over the HDFS filesystem.

hdfs.mkdir("/RHadoop/2/")

hdfs.rm or hdfs.rmr or hdfs.delete - to delete the directory from HDFS.

hdfs.rm("/RHadoop/2/")

RHadoop Function Reference: hdfs package

Utility:

hdfs.ls: This is used to list the directory from HDFS.

hdfs.ls('/')

hdfs.file.info: This is used to get meta information about the file stored at HDFS.

hdfs.file.info("/RHadoop")

RHadoop Function Reference: rmr package

For storing and retrieving data:

to.dfs: This is used to write R objects from or to the filesystem.

small.ints = to.dfs(1:10)

from.dfs: This is used to read the R objects from the HDFS filesystem that are in the binary encrypted format.

from.dfs('/tmp/RtmpRMIXzb/file2bda3fa07850')

RHadoop Function Reference: rmr package

For MapReduce:

mapreduce: This is used for defining and executing the MapReduce job.

mapreduce(input, output, map, reduce, combine, input.fromat, output.format)

keyval: This is used to create and extract key-value pairs.

keyval(key, val)

Structure of Rscript using RHadoop

```
1 library(rhdfs)                          # load the library
2 library(rmr)

3 hdfs.init()                             # Initialize Rhadoop

4 map<-function(k,v) { …}                 # Define Map and Reduce functions
5 reduce<-function(k,vv) { …}

6 mapreduce(                              # mapredue job
        input =”data.txt”,
        output=”output”,
        textinputformat =rawtextinputformat,
        map = map,
        reduce=reduce
        )
```

Set Environment Variables, Load Library and Initialize

#BegIn – The below lines has to be executed each time we open R for RHadoop to work

```
Sys.setenv(HADOOP_CMD="/usr/bin/hadoop")
Sys.setenv(HADOOP_STREAMING="/usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-streaming-2.6.0-mr1-cdh5.4.2.jar")
Sys.setenv(JAVA_HOME="/usr/java/jdk1.7.0_67-cloudera")

library(rmr2)
library(rhdfs)

hdfs.init()
```

#End– The above lines has to be executed each time we open R for RHadoop to work

Sample Program 1: Square of a vector

```
ints = to.dfs(1:100)

calc = mapreduce(input = ints, map =
function(k, v) cbind(v, 2*v))

from.dfs(calc)
```

Sample Program 2: Min and Max for generated random deviates

```
small.ints = to.dfs(1:10)
## Defining the MapReduce job
mapreduce(
# defining input parameters as small.ints hdfs object, map parameter as
#function to calculate the min and max for generated random deviates.
input = small.ints,
map = function(k, v)
{
lapply(seq_along(v), function(r){
x <- runif(v[[r]])
keyval(r,c(max(x),min(x)))
})})

output<-from.dfs( '/tmp/file338d4f159cdb')
table_output<- do.call('rbind', lapply(output$val,"[[",2))     #output in table format
table_output
```

output

```
> output <- from.dfs('/tmp/Rtmpq7v2u6/file1515859585')
> table_output<- do.call('rbind', lapply(output$val,"[[",2))
> table_output
           [,1]       [,2]
 [1,] 0.8125193 0.81251934
 [2,] 0.9042196 0.45699808
 [3,] 0.8646576 0.67394221
 [4,] 0.7134127 0.43875206
 [5,] 0.6928776 0.09431795
 [6,] 0.9695492 0.08021174
 [7,] 0.8968259 0.05925483
 [8,] 0.9488959 0.14835090
 [9,] 0.8658371 0.12823777
[10,] 0.7888141 0.02754461
```

Word Count

```
# Defining wordcount MapReduce function
wordcount = function(input,
                     output = NULL,
                     pattern = " "){

# Defining wordcount Map function
wc.map = function(., lines) {
           keyval(
             unlist(strsplit(
                 x = lines,
                 split = pattern)),
           1)}


# Defining wordcount Reduce function
wc.reduce = function(word, counts ) {
                     keyval(word, sum(counts))}
```

Word Count..

- # Defining MapReduce parameters by calling mapreduce function
- mapreduce(input = input ,
- output = output,
- input.format = "text",
- map = wc.map,
- reduce = wc.reduce,
- combine = T)}

- # Running MapReduce Job by passing the Hadoop input directory location as parameter
- wordcount('/RHadoop/1/')

- # Retrieving the RHadoop MapReduce output data by passing output directory location as parameter
- from.dfs("/tmp/RtmpRMIXzb/file2bda5e10e25f")

Problem

Count the number of occurrences of randomly generated numbers

In R:

rbinom(N,n,p) generates a sequence of ntrials,each with probability p of success.The sample space will be from 1 to N

E.g.

groups = rbinom(25, n=100, prob= 0.5)

Data Analytics

Case Studies

Categorize the web page: Google Analytics

Sample Data Set:

source	pagePath	visits
facebook.com	/	1
(direct)	/gtuadmissionhelpline-team	1
(direct)	/merit-calculator	1
(direct)	/	2
facebook.com	/	5
Google	/	1

Count the number of aadhar registrations per state

- Sample Data Set:

State	District	Aadhaar generated	Enrolment Rejected
Andaman and Nicobar Islands	Nicobar	0	1
Andaman and Nicobar Islands	South Andaman	1	0
Andhra Pradesh	Ananthapur	294	160
Andhra Pradesh	Chittoor	64	69
Andhra Pradesh	Cuddapah	27	49
Andhra Pradesh	East Godavari	47	79
Andhra Pradesh	Guntur	93	78

Compute the frequency of stock market change: Yahoo.finanace

stock_infy<-read.csv ("http://ichart.finance.yahoo.com/table.csv?s=INFY.BO")

Sample Data Set:

Prices						
Date	Open	High	Low	Close	Volume	Adj Close*
20 Mar, 2015	2,233.00	2,269.50	2,233.00	2,258.95	78,700	2,258.95
19 Mar, 2015	2,227.00	2,244.15	2,224.20	2,229.90	510,700	2,229.90
18 Mar, 2015	2,250.00	2,260.75	2,220.00	2,230.20	41,500	2,230.20
17 Mar, 2015	2,260.20	2,278.50	2,218.20	2,241.95	84,800	2,241.95
16 Mar, 2015	2,210.00	2,282.00	2,210.00	2,267.00	77,100	2,267.00
13 Mar, 2015	2,226.10	2,228.80	2,194.40	2,216.45	55,600	2,216.45
12 Mar, 2015	2,184.90	2,226.35	2,177.50	2,217.80	86,200	2,217.80
11 Mar, 2015	2,190.00	2,240.00	2,172.60	2,183.60	127,000	2,183.60
10 Mar, 2015	2,200.00	2,207.00	2,163.30	2,200.10	90,300	2,200.10