Distributed Data Analysis with Hadoop and R

Jonathan Seidman and Ramesh Venkataramaiah, Ph. D.
Chicago Big Data – August 2011
Flow of this Talk

• Introductions

• Hadoop, R and Interfacing the two

• Our Prototypes

• A use case for interfacing Hadoop and R

• Alternatives for Running R on Hadoop

• Alternatives to Hadoop and R

• Conclusions

• References
Who We Are

• Ramesh Venkataramaiah, Ph. D.
  – Principal Engineer, TechOps
  – rvenkataramaiah@orbitz.com
  – @rvenkatar

• Jonathan Seidman
  – Lead Engineer, Business Intelligence/Big Data Team
  – jseidman@orbitz.com
  – @jseidman

• Orbitz Careers
  – @OrbitzTalent
Launched in 2001, Chicago, IL

Over 160 million bookings
Hadoop and R as an analytic platform?
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.
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.
Our Options

• Data volume reduction by sampling
  – Very bad for long-tail data distribution
  – Approximation lead to bad conclusion

• Scaling R
  – Still in-memory
  – But make it parallel using segue, Rhide, R-Hive…

• Use sql-like interfaces
  – Apache Hive with Hadoop
  – File sprawl and process issues

• Regular DBMS
  – How to fit square peg in a round hole
  – No in-line R calls from SQL but commercial efforts are underway.

• This Talk: Interface Hadoop with R over dataspaces
Why Interface Hadoop and R at cluster level?

- R only works on:

- HDFS can be “the” data and analytic store.

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

Options to interface Hadoop and R, at cluster level?
Our prototypes
User segmentations
Hotel bookings
Airline Performance*

* Public dataset
Before Hadoop

Non-transactional Data (e.g. searches)

Transactional Data (e.g. Bookings)

Data Warehouse
With Hadoop

- Non-Transactional Data (e.g. Searches) in Hadoop
- Transactional Data (e.g. Bookings) in Data Warehouse
Getting a Buy-in

presented a long-term, unstructured data growth story and explained how this will help harness long-tail opportunities at lowest cost.

- Traditional DW
- Classical Stats
- Sampling

- Big Data
- Specific spikes
- Median is not the message

* From a blog
## Workload and Resource Partition

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Data Volume</th>
<th>Platform preference</th>
<th>Resource Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection</td>
<td>Scalable, elastic GB to TB</td>
<td>Hadoop (cluster level)</td>
<td>Developers</td>
</tr>
<tr>
<td>Aggregation/Summary</td>
<td>Large scale, Big data GB to TB</td>
<td>Rhipe Hadoop streaming Hadoop Interactive</td>
<td>Developers Analysts Machine Learning Teams</td>
</tr>
<tr>
<td>Modeling/Visualization</td>
<td>Small datasets, In-memory, MB to GB</td>
<td>R (stand-alone)</td>
<td>Analysts Machine Learning Teams</td>
</tr>
</tbody>
</table>
User Segmentation by Browsers

![Prices by Browser chart]

- MSIE
- Firefox
- Safari
- Chrome
Seasonal variations

- Customer hotel stay gets longer during summer months
- Could help in designing search based on seasons.
Airline Performance
Description of Use Case

• Analyze openly available dataset: Airline on-time performance.
• Dataset was used in “Visualization Poster Competition 2009”
  – Approximately 120 MM records totaling 120GB.
• Available at: http://stat-computing.org/dataexpo/2009/

A sample of the text file

```
1987, 10, 23, 5, 1841, 1750, 2105, 2005, PS, 1905, NA, 144, 135, NA, 60, 51, LAX, SEA, 954, NA, NA, 0, NA, 0, ...
1987, 10, 24, 6, 1752, 1750, 2010, 2005, PS, 1905, NA, 138, 135, NA, 5, 2, LAX, SEA, 954, NA, NA, 0, NA, 0, ...
... 
... 
```
Our dataset

<table>
<thead>
<tr>
<th>Year</th>
<th>Month</th>
<th>Count</th>
<th>Airline</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007</td>
<td>1</td>
<td>21062</td>
<td>9E</td>
<td>7.54505700</td>
</tr>
<tr>
<td>2007</td>
<td>10</td>
<td>22290</td>
<td>9E</td>
<td>5.45684200</td>
</tr>
<tr>
<td>2007</td>
<td>11</td>
<td>20813</td>
<td>9E</td>
<td>4.74520700</td>
</tr>
<tr>
<td>2007</td>
<td>12</td>
<td>21888</td>
<td>9E</td>
<td>22.47679000</td>
</tr>
</tbody>
</table>
Airline Delay Plot: R code

```r
> deptdelays.monthly.full <- read.delim("~/OSCON2011/Delays_by_Month.dat", header=F)
> View(deptdelays.monthly.full)
> names(deptdelays.monthly.full) <- c("Year","Month","Count","Airline","Delay")

> Delay_by_month <- deptdelays.monthly.full[order(deptdelays.monthly.full $Delay,decreasing=TRUE),]

> Top_10_Delay_by_Month <- Delay_by_month[1:10,]
> Top_10_Normal <- ((Delay - mean(Delay)) / sd(Delay))

> symbols( Month, Delay, circles= Top_10_Normal, inches=.3, fg="white",bg="red",...)
> text(Month, Delay, Airline, cex = 0.5)
```
Airline delay
> library(lattice)
> deptdelays.monthly.full$Year <- as.character(deptdelays.monthly.full$Year)
> h <- histogram(~Delay|Year, data=deptdelays.monthly.full, layout=c(5,5))
> update(h)
Running R on Hadoop: Hadoop Streaming
Hadoop Streaming – Overview

• An alternative to the Java MapReduce API which allows you to write jobs in any language supporting stdin/stdout.
• Limited to text data in current versions of Hadoop. Support for binary streams added in 0.21.0.
• Requires installation of R on all DataNodes.
Hadoop Streaming – Dataflow

* Map function receives input records line-by-line via standard input.

<table>
<thead>
<tr>
<th>Year</th>
<th>Month</th>
<th>Day</th>
<th>Value1</th>
<th>Value2</th>
<th>Value3</th>
<th>Value4</th>
<th>Value5</th>
<th>Value6</th>
<th>Value7</th>
<th>Value8</th>
<th>Value9</th>
<th>Value10</th>
<th>Value11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1988</td>
<td>1</td>
<td>17</td>
<td>PI</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SYR,BWI</td>
</tr>
<tr>
<td>1988</td>
<td>1</td>
<td>17</td>
<td>PI</td>
<td></td>
<td>150</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1987</td>
<td>10</td>
<td>11</td>
<td>PS</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SAN,SFO</td>
</tr>
<tr>
<td>1987</td>
<td>10</td>
<td>11</td>
<td>PS</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1987</td>
<td>10</td>
<td>11</td>
<td>PS</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1987</td>
<td>10</td>
<td>11</td>
<td>DL</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hadoop Streaming – Dataflow Continued

* Reduce receives map output key/value pairs sorted by key, line-by-line.
Hadoop Streaming Example – map.R

```r
#!/usr/bin/env Rscript

# For each record in airline dataset, output a new record consisting of
# "CARRIER\YEAR\MONTH \t DEPARTURE_DELAY"

con <- file("stdin", open = "r")
while (length(line <- readLines(con, n = 1, warn = FALSE)) > 0) {
  fields <- unlist(strsplit(line, "\\,"))
  # Skip header lines and bad records:
  if (!(identical(fields[[1]], "Year") & length(fields) == 29) {
    deptDelay <- fields[[16]]
    # Skip records where departure delay is "NA":
    if (!(identical(deptDelay, "NA"))) {
      # field[9] is carrier, field[1] is year, field[2] is month:
      cat(paste(fields[[9]], "|", fields[[1]], "|", fields[[2]], sep=""), "\t",
           deptDelay, "\n")
    }
  }
}
close(con)
```
Hadoop Streaming Example – reduce.R

```r
#!/usr/bin/env Rscript

# For each input key, output a record composed of
# YEAR \t MONTH \t RECORD_COUNT \t AIRLINE \t AVG_DEPT_DELAY

con <- file("stdin", open = "r")
delays <- numeric(0) # vector of departure delays

lastKey <- ""
while (length(line <- readLines(con, n = 1, warn = FALSE)) > 0) {

  split <- unlist(strsplit(line, "\t"))

  key <- split[[1]]
depDelay <- as.numeric(split[[2]])

  if (!identical(lastKey, "") & (!identical(lastKey, key))) {
    keySplit <- unlist(strsplit(lastKey, "\""))

    cat(keySplit[[2]], "\t", keySplit[[3]], "\t", length(delays), "\t", keySplit[[1]], "\t", (mean(delays)), "\n")

    lastKey <- key
    delays <- c(delays, deptDelay)
  } else {
    # Still working on same key so append dept delay value to vector:
    lastKey <- key
    delays <- c(delays, deptDelay)
  }
}

# We're done, output last record:
keySplit <- unlist(strsplit(lastKey, "\"\""))
cat(keySplit[[2]], "\t", keySplit[[3]], "\t", length(delays), "\t", keySplit[[1]], "\t", (mean(delays)), "\n")
```
Running R on Hadoop: Hadoop Interactive
Hadoop Interactive (hive) – Overview

• Very unfortunate acronym.
• Provides an interface to Hadoop from the R environment.
  – Functions to access HDFS
  – Control Hadoop
  – And run streaming jobs directly from R
• Allows HDFS data, including the output from MapReduce processing, to be manipulated and analyzed directly from R.
• Seems to still have some rough edges.
Hadoop Interactive – Example

```r
#!/usr/bin/env Rscript

mapper <- function() {
  ...
}

reducer <- function() {
  ...
}

library(hive)
DFS_dir_remove("/dept-delay-month", recursive = TRUE, henv = hive())
hive_stream(mapper = mapper, reducer = reducer,
  input="/data/airline/", output="/dept-delay-month")
results <- DFS_read_lines("/dept-delay-month/part-r-00000", henv = hive())
```
Running R on Hadoop: RHIPE
RHIPE – Overview

- Active project with frequent updates and active community.
- RHIPE is based on Hadoop streaming source, but provides some significant enhancements, such as support for binary files.
- Developed to provide R users with access to same Hadoop functionality available to Java developers.
  - For example, provides `rhcounter()` and `rhstatus()`, analogous to counters and the reporter interface in the Java API.
RHIPE – Overview

• Can be somewhat confusing and intimidating.
  – Then again, the same can be said for the Java API.
  – Worth taking the time to get comfortable with.
RHIPE – Overview

• Allows developers to work directly on data stored in HDFS in the R environment.
• Also allows developers to write MapReduce jobs in R and execute them on the Hadoop cluster.
• RHIPE uses Google protocol buffers to serialize data. Most R data types are supported.
  – Using protocol buffers increases efficiency and provides interoperability with other languages.
• Must be installed on all DataNodes.
RHIPE – MapReduce

```r
map <- expression({})
reduce <- expression(
    pre = {…},
    reduce = {…},
    post = {…}
)

z <- rhmr(map=map,reduce=reduce,
          inout=c("text","sequence"),
          ifolder=INPUT_PATH ,
          ofolder=OUTPUT_PATH,
          ...
)

rhex(z)
```
### RHIPE – Dataflow

**Keys** = […]

**Values** =

```
[1988,1,9,6,1348,1331,1458,1435,PI,942,NA,70,64,NA,23,17,SYR,BWI...
1988,1,17,7,1331,1331,1440,1435,PI,942,NA,69,64,NA,5,0,SYR,BWI...
1987,10,14,3,741,730,912,849,PS,1451,NA,91,79,NA,23,11,SAN,SFO...
1987,10,21,3,728,730,848,849,PS,1451,NA,80,79,NA,-1,-2,SAN,SFO...
1987,10,23,5,731,730,902,849,PS,1451,NA,91,79,NA,13,1,SAN,SFO...
1987,10,30,5,1712,1658,1811,1800,DL,475,NA,59,62,NA,11,14,LEX,ATL...]
```

**Input to map**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>1988</td>
</tr>
<tr>
<td>PI</td>
<td>1988</td>
</tr>
<tr>
<td>PS</td>
<td>1987</td>
</tr>
<tr>
<td>PS</td>
<td>1987</td>
</tr>
<tr>
<td>PS</td>
<td>1987</td>
</tr>
<tr>
<td>DL</td>
<td>1987</td>
</tr>
</tbody>
</table>

**Output from map**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PII1988</td>
<td>17</td>
</tr>
<tr>
<td>PII1988</td>
<td>0</td>
</tr>
<tr>
<td>PSI1987</td>
<td>11</td>
</tr>
<tr>
<td>PSI1987</td>
<td>-2</td>
</tr>
<tr>
<td>PSI1987</td>
<td>1</td>
</tr>
<tr>
<td>DL1987</td>
<td>14</td>
</tr>
</tbody>
</table>

* Note that Input to map is a vector of keys and a vector of values.
RHIPE – Dataflow Continued

* Note that input to reduce is each unique key and a vector of values associated with that key.

### Input to reduce

<table>
<thead>
<tr>
<th>Key</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>DL1987/10</td>
<td>[14]</td>
</tr>
<tr>
<td>PI1988/1</td>
<td>[0, 17]</td>
</tr>
<tr>
<td>PSI1987/10</td>
<td>[1,11,-2]</td>
</tr>
</tbody>
</table>

### Output from reduce

<table>
<thead>
<tr>
<th>Year</th>
<th>Value1</th>
<th>Value2</th>
<th>Key</th>
<th>Value3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1987</td>
<td>10</td>
<td>1</td>
<td>DL</td>
<td>14</td>
</tr>
<tr>
<td>1988</td>
<td>1</td>
<td>2</td>
<td>PI</td>
<td>8.5</td>
</tr>
<tr>
<td>1987</td>
<td>10</td>
<td>3</td>
<td>PS</td>
<td>3.333333</td>
</tr>
</tbody>
</table>
```r
#!/usr/bin/env Rscript

library(Rhipe)
rhinit(TRUE, TRUE)

map <- expression({
  # For each input record, parse out required fields and output new record:
  extractDeptDelays = function(line) {
    fields <- unlist(strsplit(line, "\,"))
    # Skip header lines and bad records:
    if (!identical(fields[[1]], "Year") & length(fields) == 29) {
      deptDelay <- fields[[16]]
      # Skip records where departure delay is "NA":
      if (!identical(deptDelay, "NA")) {
        # field[9] is carrier, field[1] is year, field[2] is month:
        rhcollect(paste(fields[[9]], "\", fields[[1]], "\", fields[[2]], sep=""),
                  deptDelay)
      }
    }
  }
  # Process each record in map input:
  lapply(map.values, extractDeptDelays)
})
```
reduce <- expression(
  reduce = {
    count <- length(reduce.values)
    avg <- mean(as.numeric(reduce.values))
    keySplit <- unlist(strsplit(reduce.key, "\\"))
  },
  post = {
    rhcollect(keySplit[[2]],
      paste(keySplit[[3]], count, keySplit[[1]], avg, sep="\t"))
  }
)
RHIPE – Example

```r
# Create job object:
z <- rhmr(map=map, reduce=reduce, 
        ifolder="/data/airline/", ofolder="/dept-delay-month",
        inout=c('text', 'text'), jobname='Avg Departure Delay By Month')
# Run it:
rhex(z)

# Visualize results:
library(lattice)
rhget("/dept-delay-month/part-r-00000", "deptdelay.dat")
depth_delays.monthly.full <- read.delim("deptdelay.dat", header=F)
names(deptdelays.monthly.full) <- c("Year", "Month", "Count", "Airline", "Delay")
depth_delays.monthly.full$Year <- as.character(deptdelays.monthly.full$Year)
h <- histogram(~Delay|Year, data=deptdelays.monthly.full, layout=c(5,5))
update(h)
```
Running R on Hadoop: Segue
Segue – Overview

• Intended to work around single-threading in R by taking advantage of Hadoop streaming to provide simple parallel processing.
  – For example, running multiple simulations in parallel.
• Suitable for embarrassingly pleasantly parallel problems – big CPU, not big data.
• Runs on Amazon’s Elastic Map Reduce (EMR).
  – Not intended for internal clusters.
• Provides `emrLapply()`, a parallel version of `lapply()`
Segue – Example

```r
estimatePi <- function(seed) {
  set.seed(seed)
  numDraws <- 1e6

  r <- .5 # radius... in case the unit circle is too boring
  x <- runif(numDraws, min=-r, max=r)
  y <- runif(numDraws, min=-r, max=r)
  inCircle <- ifelse((x^2 + y^2)^.5 < r, 1, 0)

  return(sum(inCircle) / length(inCircle) * 4)
}

seedList <- as.list(1:1000)
require(segue)
myCluster <- createCluster(20)
myEstimates <- emrapply( myCluster, seedList, estimatePi )
stopCluster(myCluster)
```
Predictive Analytics on Hadoop: Sawmill
Sawmill – Overview

• A framework for integrating a PMML-compliant Scoring Engine with Hadoop.
• Hadoop streaming allows easier integration of a scoring engine into reducer code (Python and R).
  – The output of a MapReduce run becomes a segmented PMML model – one segment for each partition
• Training the models and Scoring are separate MapReduce jobs.
• Interoperates with open source scoring engines such as Augustus, as well as a forthcoming R scoring engine.
Alternatives

Alternate languages/libraries:

• Apache Mahout
  – Scalable machine learning library.
  – Offers clustering, classification, collaborative filtering on Hadoop.

• Python
  – Many modules available to support scientific and statistical computing.
Alternatives

Alternative parallel processing frameworks:

• Revolution Analytics
  – Provides commercial packages to support processing big data with R.

• Other HPC/parallel processing packages for R, e.g. Rmpi or snow.
Alternatives

Apache Hive + RJDBC?

- We haven’t been able to get it to work yet.
- You can however wrap calls to the Hive client in R to return R objects. See https://github.com/satpreetsingh/rDBwrappers/wiki
Alternatives

Interesting solutions that you can’t have:

• Ricardo
  – Developed as part of a research project at IBM.
  – Interesting paper published, but apparently no plans to make available.
Conclusions

- If practical, consider using Hadoop to aggregate data for input to R analyses.
- Avoid using R for general purpose MapReduce use.
Conclusions

• For simple MapReduce jobs, or “embarrassingly” parallel jobs on a local cluster, consider Hadoop streaming.
  – Limited to processing text only.
  – But easy to test at the command line outside of Hadoop:
    • $ cat DATAFILE | ./map.R | sort | ./reduce.R
• To run compute-bound analyses with relatively small amount of data on the cloud look at Segue.
Conclusions

• Otherwise, your best bet is RHIPE.
• Also consider alternatives – Mahout, Python, etc.
Conclusions

On an operational note:

• Make sure your cluster nodes are consistent – same version of R installed, required libraries are installed on each node, etc.
Example Code

• https://github.com/jseidman/hadoop-R
References

• Hadoop
  – Apache Hadoop project: http://hadoop.apache.org/

• R
References

• Hadoop Streaming
  – Word count example in R: https://forums.aws.amazon.com/thread.jspa?messageID=129163
References

• Hadoop InteractiVE
  – Project page on CRAN:
    http://cran.r-project.org/web/packages/hive/index.html
  – Simple Parallel Computing in R Using Hadoop:
    http://www.rmetrics.org/Meielisalp2009/Presentations/Theussl1.pdf
References

- **RHIPE**
  - RHIPE - R and Hadoop Integrated Processing Environment:  
    http://www.stat.purdue.edu/~sguha/rhipe/
  - Wiki: https://github.com/saptarshiguha/RHIPE/wiki
  - Installing RHIPE on CentOS:  
    https://groups.google.com/forum/#!topic/brumail/qH1wjTBgwYI
  - Introduction to using RHIPE: http://ml.stat.purdue.edu/rhafen/rhipe/
  - RHIPE combines Hadoop and the R analytics language, SD Times:  
    http://www.sdtimes.com/link/34792
  - Using R and Hadoop to Analyze VoIP Network Data for QoS, Hadoop World 2010:
    - video:  
      http://www.cloudera.com/videos/
      hw10_video_using_r_and_hadoop_to_analyze_voip_network_data_for_qos
    - slides:  
      http://www.cloudera.com/resource/
      hw10_voice_over_ip_studying_traffic_characteristics_for_quality_of_service
References

• Segue
  – Project page: http://code.google.com/p/segue/
  – Google Group: http://groups.google.com/group/segue-r
  – Abusing Amazon’s Elastic MapReduce Hadoop service easily, from R, Jefferey Breen: http://jeffreybreen.wordpress.com/2011/01/10/segue-r-to-amazon-elastic-mapreduce-hadoop/
References

• Sawmill
  – More information:
    • Open Data Group www.opendatagroup.com
    • oscon-info@opendatagroup.com
  – Augustus, an open source system for building & scoring statistical models
    • augustus.googlecode.com
  – PMML
    • Data Mining Group: dmg.org
  – Analytics over Clouds using Hadoop, presentation at Chicago Hadoop User Group:
References

• Ricardo
References

• General references on Hadoop and R
  – Pete Skomoroch’s R and Hadoop bookmarks: http://www.delicious.com/pskomoroch/R+hadoop
  – Quora – How can R and Hadoop be used together?: http://www.quora.com/How-can-R-and-Hadoop-be-used-together
References

• Mahout
  – Mahout project: http://mahout.apache.org/
• Python
• CRAN Task View: High-Performance and Parallel Computing with R, a set of resources compiled by Dirk Eddelbuettel: http://cran.r-project.org/web/views/HighPerformanceComputing.html
References

• Other examples of airline data analysis with R:
  – A simple Big Data analysis using the RevoScaleR package in Revolution R:
And finally…

Parallel R (working title), Q Ethan McCallum, Stephen Weston, O’Reilly Press, due autumn 2011

“R meets Big Data - a basket of strategies to help you use R for large-scale analysis and computation.”