rhipe Documentation

Release 0.5

Saptarshi Guha

September 10, 2009

CONTENTS

Mainpage

CHAPTER

ONE

SETTING UP RHIPE

1.1 Requirements

1. *Protobuffers*

   RHIPE uses Google's Protobuf library for serialization. This(the C/C++ libraries) must be installed on *all* machines (master/workers). Get Protobuffers from http://code.google.com/p/protobuf/. RHIPE already has the protobuf jar file inside it.

   Non Standard Locations If installing protobuf to a non standard location, update the PKG_CONFIG_PATH variable, e.g

   ```
   export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:$CUSTROOT/lib/pkgconfig/
   ```

2. *R* , tested on 2.8
3. *rJava* The R package needs rJava.

Tested on RHEL Linux, though *may* work on Windows

1.2 Installation

On every machine

```
tar zxvf rhipe.VERSION.tar.gz
R CMD INSTALL rhipe.VERSION
```

To load it

```
library(Rhipe)
```

CHAPTER

TWO

THE RHLAPPLY COMMAND

2.1 Introduction

rhapply applies a user defined function to the elements of a given R list or the function can be run over the set of numbers from 1 to n. In the former case the list is written to a sequence file,whose length is the default setting of rhwrite.

Running a hundreds of thousadands of seperate trials can be terribly inefficient, instead consider grouping them, i.e set mapred.max.tasks to a value much smaller than the length of the list.

2.2 Return Value

rhlapply returns a list, the names of which is equal to the names of the input list (if given).

2.3 Function Usage

```
rhlapply <- function( ll=NULL,
                      fun,
                      ifolder="",
                      ofolder="",
                      readIn=T,
                      inout=c('lapply','sequence')
                      mapred=list()
                      setup=NULL,jobname="rhlapply",...
                      )
```

Description follows

ll The list object, optional. Applies fun to ll[[i]] . If instead ll is a numeric, applies fun to each element of seq(1,ll). If not given, must provide a value for ifolder

fun A function that takes only one argument.

ifolder If ll is null, provide a source here. Also change the value of inout[1] to either text or sequence.

readIn The results are stored in a temporary sequence file on the DFS which is deleted. Should the results be returned in a list? Default is TRUE. For large number of output key-values (e.g 1MM) set this to FALSE, using the default options to rhread is extremely slow.

ofolder If given the results are written to this folder and not deleted. If not, they are written to temporary folder, read back in (assuming `readIn` is TRUE) and deleted.

mapred Options passed onto `rhmr`

setup And expression that is called before running `func`. Called once per JVM.

... passed onto RHMR.

2.3.1 RETURN

An object that is passed onto `rhex`.

CHAPTER

THREE

THE RHMR COMMAND

3.1 Introduction

The `rhmr` command runs a general mapreduce program using user supplied map and reduce commands.

3.2 Return Value

In general a set of files on the Hadoop Distributed File System. It can be of Text Format or a Sequence file format. In case of the latter, the key and values can be any R data structure.

3.3 Function

```
rhmr <- function(map,reduce=NULL,
          combiner=F, #CANNOT BE CHANGED
          setup=NULL,
          cleanup=NULL,
          ofolder='',
          ifolder='',
          inout=c("text","text"),
          mapred=NULL,
          shared=c(),
          jarfiles=c(),
          copyFiles=F,
          opts=rhoptions(),jobname="")
```

map A map expression, not a function. The map expression can expect a list of keys in `map.keys` and list of values in `map.values`.

reduce Can be null if only a map job. If not,reduce should be an expression with three attributes

pre Called for a new key, but no values have been read. The key is present in `reduce.key`.

reduce Called for reducing the incoming values. The values are in a list called `reduce.values`

post Called when all the values have been sent.

combiner Uses a combiner if TRUE. If so, then `reduce.values` present in the `reduce$reduce` expression will be a *subset* of values.

setup An expression that can be called to setup the environment. Called once for every task. It can be a list of two attributes `map` and `reduce` which are expressions to be run in the map and reduce stage. If a single expression then that is run for both map and reduce

cleanup Same as for `setup`, run when all work for a task is complete.

ifolder A folder or file to be processed. Can be a vector of strings.

ofolder The folder to store output in. Side effects will be copied here.

**inout` A vector of input type and output type. `text`** indicates Text Format. Use `mapred.field.separator` to seperate the elements of a vector.

sequence is a sequence format. Outputs in this form /can/ be used as an input.

binary is a simple binary format consisting of key-length, key data, value-length, value data where the lengths are integers in network order. Though *much* faster than sequence in terms of reading in data, it *cannot* be used an input to a map reduce operation.

shared A vector of files on the HDFS that will be copied to the working directory of the R program. These files can then be loaded as easily as `load(filename)` (removed leading path)

jarfiles Copy jar files if required. Experimental, probably doesn't work.

copyFiles For side effects to be copied back to the DFS, set this to TRUE, otherwise they wont be copied.

mapred Set Hadoop options here and RHIPE options.

jobname the jobname, if not given, then current date and time is the job title.

3.4 RHIPE Options

rhipe_stream_buffer The size of the STDIN buffer used to write data to the R process(in bytes) *default:* 10*1024 bytes

mapred.textoutputformat.separator The text that seperates the key from value when `inout[2]` equals text. *default:* Tab

mapred.field.separator The text that seperates fields when `inout[2]` equals text. *default:* Space

rhipe_reduce_buff_size The maximum length of `reduce.values` *default:* 10,000

rhipe_map_buff_size The maximum length of `map.values` (and `map.keys`) *default:* 10,000

3.5 Status, Counters and Writing Output

3.5.1 Status

To update the status use `rhstatus` which takes a single string e.g `rhstatus("Nice")` This will also indicate progress.

3.5.2 Counter

To update the counter C in the group G with a number N, user `rhcounter(G,C,N)` where C and G are strings and N is a number.

3.5.3 Output

To output data use `rhcollect(KEY,VALUE)` where KEY and VALUE are R objects that can be serialized by `rhsz` (see the misc page). If one needs to send across complex R objects e.g the KEY is a function, do something like `rhcollect(serialize(KEY,NULL),VALUE)`

3.6 Side Effect files

Files written to `tmp/` (no leading slash !) e.g `pdf("tmp/x.pdf")` will be copied to the output folder.

CHAPTER

FOUR

MISCELLANEOUS COMMANDS

4.1 Introduction

This is a list of supporting functions for reading, writing sequence files and manipulating files on the Hadoop Distributed File System (HDFS).

4.2 Serialization

4.2.1 rhsz

```
rhsz <- function(object)
```

Serializes a given R object. Currently the only objects that can be serialized are vectors of Raws,Numerics, Integers, Strings(including NA), Logical(including NA) and lists of these and lists of lists of these. Attributes are copied to(e.g names attributes). It appears objects like matrices, factors also get serialized and unserialized sucessfully.

4.2.2 rhuz

```
rhuz <- function(object)
```

Unserializes a raw object returned from `rhsz`

4.3 HDFS Related

4.3.1 rhsave

```
rhsave <- function(..., file)
```

Saves the objects in `...` to `file` on the HDFS. All other options are passed onto the R function `save`

4.3.2 rhsave.image

```
rhsave.image <- function(..., file)
```

Same as R's `save.image`, except that the file goes to the HDFS.

4.3.3 rhput

```
rhput <- function(src,dest,deleteDest=TRUE)
```

Copies the file in `src` to the `dest` on the HDFS, deleting destination if `deleteDest` is TRUE.

4.3.4 rhget

```
rhget <- function(src,dest)
```

Copies `src``(on the HDFS) to ``dest` on the local. If `src` is a directory and `dest` exists, `src` is copied inside `dest``(i.e a folder inside ``dest`).If not(i.e `dest` does not exist), `src`'s contents is copied to a new folder called `dest`. If `src` is a file, and `dest` is a directory `src` is copied inside `dest` . If `dest` does not exist, it is copied to that file

Wildcards allowed

OVERWRITES!

4.3.5 rhls

```
rhls <- function(dir)
```

Lists the path at `dir`. Wildcards allowed.

4.3.6 rhdel

```
rhdel <- function(dir)
```

Deletes file(s) at/in `dir`. Wildcards allowed.

4.3.7 rhwrite

```
rhwrite <- function(lo,f,n=NULL,...)
```

Writes the list `lo` to the file `f`. `n` is the number of sequence files to split the list into. The default value of `n` is `mapred.map.tasks` * `mapred.tasktracker.map.tasks.maximum` .

4.3.8 rhread

```
rhread <- function(files,max=NA,batch=100,length=1000)
```

Reads files(s) from `files` (which could be a directory). Wildcards allowed.

`max` is the maximum number of key-values to be read.

`batch` is how many to key-value pairs to request from Java in one go.

`length` is the initial size of the return list(a larger value will makes things faster if one is expecting to read in many items).

The latter two are important when it comes to reading sequence files with many values(100K+), set batch to a large number and `length` to an equally large number to reduce the number of JNI calls and vector resizes.

4.3.9 rhreadBin

```
rhreadBin <- function(filename, max=as.integer(-1), bf=as.integer(0))
```

Reads data outputed in 'binary' form. `max` is the maximum number to read, -1 is all. `bf` is the read buffer, 0 implies the os specified default `BUFSIZ`

CHAPTER

FIVE

USING RHIPE ON EC2

5.1 Introduction

We have release two AMIs(32 and 64bit). Both are based on Fedora 8 and have Hadoop 0.19.1,R 2.8 and RIPE (latest) installed. s3sync is also present.

32 bit ami-4b678122

64 bit ami-9f7492f6

The following describes the usage of the EC2 scripts.

5.2 Usage

- Get an Amazon EC2 account and confirm the ability to start and instance from the command line (using ec2-tools).
- Unzip the rhipe-ec2 distribution (see the downloads page)
- OPTIONS

In `bin/hadoop-ec2-env.sh` template there are several options:

AWS_ACCOUNT_ID fill this from the Amazon Account Identifiers

AWS_ACCESS_KEY_ID same as above

AWS_SECRET_ACCESS_KEY same as above

RSOPTS options to Rserve, default:

```
-max-nsize=1G --max-ppsize=100000 --RS-port 8888
```

R_USER_FILE a URL to an R script. This file is executed on machine boot up. Useful to install R packages. Read `bin/hadoop-ec2-env.sh.template` for details.

INSTANCE_TYPE choose the Amazon machine instance type. For details, go to http://aws.amazon.com/ec2/instance-types/

- Save the file as `bin/hadoop-ec2-env.sh`

5.3 Some launch commands

- launch

```
bin/hadoop-ec2 launch-cluster clustername number-of-workers
```

Replace clustername with the name of the cluster and number-of-workers with the number of workers. Use Elasticfox to check all the instances are running, this can some time.

- login

```
bin/hadoop-ec2 login clustername
```

- terminate

```
bin/hadoop-ec2 terminate-cluster clustername
```

- You can check the status of jobs at masterip:50030 in your web browser.

5.4 Useful tools

s3fox A S3 file browser that works within Firefox.

Elasticfox EC2 management tools, a Firefox add-on.

CHAPTER

SIX

EXAMPLES

6.1 rhlapply

6.1.1 Simple Example

Take a sample of 100 iid observations Xi from N(0,1). Compute the mean of the eight closest neighbours to X1. This is repeated 1,000,000 times.

```
nbrmean <- function(r){
  d <- matrix(rnorm(200),ncol=2)
  orig <- d[1,]
  ds <- sort(apply(d,1,function(r) sqrt(sum((r-orig)^2)))[-1])[1:8]
  mean(ds)
}
trials <- 1000000
```

One Machine

`trials` is 1,000,000

```
system.time({r <- sapply(1:trials, nbrmean)})
 user   system  elapsed
 1603.414    0.127 1603.789
```

Distributed, output to file

```
mapred <- list(mapred.map.tasks=1000)
r <- rhlapply(1000000, fun=nbrmean,ofolder="/test/one",mapred=mapred)
rhex(r)
```

Which took 7 minutes on a 4 core machine running 6 JVMs at once.

6.1.2 Using Shared Files and Side Effects

```
h=rhlapply(length(simlist)
  ,func=function(r){
    ## do something from data loaded from session.Rdata
    pdf("tmp/a.pdf")
    plot(animage)
    dev.off()},
```

```
  setup=expression({
    load("session.Rdata")
  }),
  hadoop=list(mapred.map.tasks=1000),
  shared.files=("/tmp/session.Rdata"))
```

Here `session.Rdata` is copied from HDFS to local temporary directories (making for faster reads). This is a useful idiom for loading code that the `rhlapply` function might depend on. For example, assuming the image is not *huge*

```
rhsave.image("/tmp/myimage.Rdata")
rhlapply(N,function(r) {
  object <- dataset[[r]]
  G(object)
},setup=expression({load("myimage.Rdata")}))
```

In the above example, I wish to apply the `G` to every element in `dataset`.

6.2 rhmr

6.2.1 Word Count

Generate the words, 1 word every line

```
rhlapply(10000,function(r) paste(sample(letters[1:10],5),collapse=""),output.folder='/tmp/words')
```

Word count using the sequence file

Run it

```
z <- rhmr(map=m,reduce=r,inout=c("sequence","sequence"),
       ifolder="/tmp/words",ofolder='/tmp/wordcount')
 rhex(z)
```

6.2.2 Subset a file

We can use this RHIPE to subset files. Setting `mapred.reduce.tasks` to 5 writes the subsetted data across 5 files (even though we haven't provided a reduce task)

```
m <- expression({
  for(x in map.values){
    y <- strsplit(x," +")[[1]]
    for(w in y) rhcollect(w,T)
  }})
z <- rhmr(map=m,inout=c("text","binary"),
    ifolder="X",ofolder='Y',mapred=list(mapred.reduce.tasks=5))
rhex(z)
```

CHAPTER

SEVEN

FAQ

1. Local Testing?

Easily enough. In `rhmr` or `rhlapply`, set `mapred.job.tracker` to 'local' in the `mapred` option of the respective command. This will use the local jobtracker to run your commands.

However keep in mind, `shared.files` will not work, i.e those files will not be copied to the working directory and side effect files will not be copied back.

1. Speed?

Similar to Hadoop Streaming. The bottlenecks are writing and reading to STDIN pipes and R.