STA242 - HW04 - Working with Large files

Hugh Crockford

March 3, 2013

1 Using Shell tools

Using a combination of grep and uniq -c, I was able to get counts of flights leaving our 4 airports by scanning files by line, with no intermediate files. I tried piping output of bzip -dc (-dc outputs directly to stdout) directly into my grep/uniq command, and this worked well for the individual sipped files, however when I tried this on the combined decade long files it tripped up during the uncompression and threw an error. I was able to run the exact same command with no issues on the uncompressed files, So I'm assuming the error had something to do with the bzip uncompression, maybe in between the multiple files?

To get the average and standard deviation, I had to put code into a shell script and pass it arguments from R. I used awk for string processing, which was fast and is fairly easy to write. The awk statement was reasonably easy to work out once the nececary fields were identified, however I struggled when trying to generalise my statement, as the nested quotes (double, single , and back) were been evaluated before passed down to the shell. I tried various methods (sprintf, paste, escaping quotes) in R but was unable to achieve a general result and ended up hardcoding into my script which is hence not generalisable. Using a case statement I was able to use one script to compute all variables recquired - counts, mean, and SD.

2 Using R

2.1 Using file connections

2.2 Using Databases

3 Using MySQL on Amazon EC2 instance.

I'd been playing around with an EC2 after setting up user on mysql 5.5 serer, I was able to establish a connection using dbconnect() After trying various ways to insert delays data into mysql (LOAD LOCAL INFILE, LOAD INFILE, BULK IMPORT etc.) I found it had to use –local-infile option to allow local adding - come funky permisisons error. Used scp to transfer R script files, and wget to pull data. Using the EC2 instance through ssh was good practice using a tmux session and forced me to use command line (vim, mysql, top etc) tools for everything. It was also usefuil to use tmux so if ssh pipe was broken, I could recover the session without any loss of data.

creating index on origin sped up avg calc from 0.74sec to 0.12 sec, a 600% increase MySQL also has a handy inbuilt function for Standard deviation, STD, which made this calculation easier although it did not affect the time taken. I tried to use entire dataset to compare speeds amoung db's but hit the space limit on my free usage ec2. Even just the 2008.csv blew up the MySQL I'd installed locally so found out I can get a free micro amazon DB, and loaded data from local machine to

kept getting process's killed, which after some googling looked like it was a memory issue on amazon, which auto kills hungry process.

4 MonetDB locally

I was intersted in the speed gains given my monetdb. The moetdb client didn't like the Year or Month variable, and threw an error relating to primary keys so I'd assume its a reserved word. To get the data into monetdb the header row had to be deleted, and all NA's deleted as they didnt match data types. I used tr to translate NA's to 0's, which satisfied both int and varchar recquirements. It was much faster to import data into a local monetdb than cloud mysql, but this was obviously due to the data upload.

5 Speed

I set up monetDB and MySQL on my machine to test the speed of each. A plot of the time taken by each db for the same function is below. As can be seen from this plot, monetDb outperformed all other dbs over all tests. One strange thing in my speed benchmarking was the effect of indexing. Using SQlite indexing on origin sped up computations by 230 percent, and monet sped up 240 percent, however using MySQL it nearly doubled the time taken.

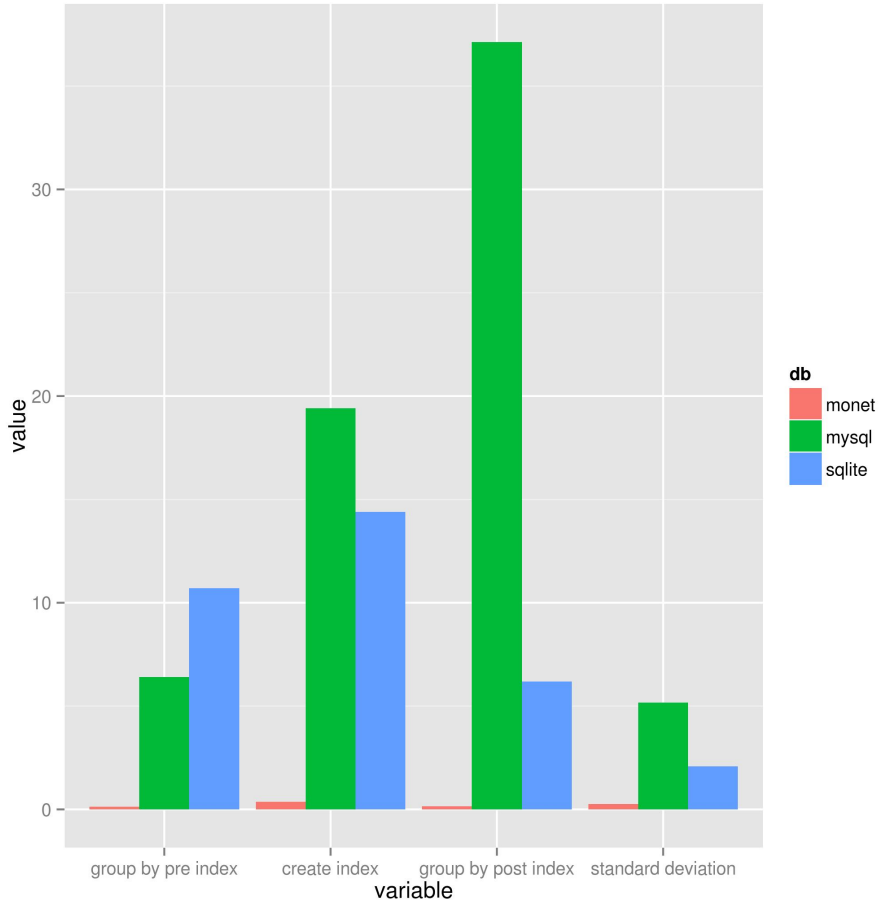

Figure 1: Comparison of time taken for various databases on same tasks

6 Git

My github repo can be found at: https://github.com/hughec1329/

7 CODE

7.1 R

```
# STA242 H04 - cli for large data
# 20130227 - HCrockford


###################################
## With shell tools
###################################

ports = c("LAX","OAK","SMF","SFO")
coll = paste(ports,"|",sep="", collapse = "") # lthrow
   together for grep.
pflat = substr(coll,1,nchar(coll)-1)
fill = "~/data/airports/open/1987.csv"
com = sprintf("cat %s | awk -F ',' '{print $17}'| grep
   -E '%s' | sort | uniq -c",fill,pflat)
ret = system(com,intern = TRUE)
foo = sapply(ret,function(i) unlist(strsplit(i," ")))
planes = structure(as.integer(lapply(foo,function(i) i[
   length(i)-1])) , names = lapply(foo,function(i) i[
   length(i)]))


### mean and SD of arrival delay times.

ports = c('"LAX"','"OAK"','"SMF"','"SFO"')        # damn
   quotes.

fill = "~/data/airports/open/1987.csv"
com = sprintf("cat %s | awk -F ',' '{if ($17 == %s) s+=
   $15} END {print s}'",fill,ports)
com = sprintf("cat %s | awk -F ',' '{if ($17 == "LAX")
   s+=$15} END {print s}'",fill,ports)       # damn
   quotes!
```

```
ret = system(com,intern = TRUE)
foo = sapply(ret,function(i) unlist(strsplit(i," ")))
planes = structure(as.numeric(lapply(nom,function(i) i[
    length(i)-1])) , names = lapply(nom,function(i) i[
    length(i)]))


com = paste("./plan.sh", "mean", fill)
ret = system(com,intern = TRUE)
foo = sapply(ret,function(i) unlist(strsplit(i," ")))
means = structure(as.numeric(foo[2,]),names = foo[1,])
means


#### SD

com = paste("./plan.sh", "sd", fill)
ret = system(com,intern = TRUE)
foo = sapply(ret,function(i) unlist(strsplit(i," ")))
sds = structure(as.numeric(foo[2,]),names = foo[1,])
sds


##################################
### within R - Cant do without messy slow loops?
##################################

fill = "~/data/airports/open/1987.csv"
con = file(fill)
nlin = as.integer(strsplit(system(paste("wc -l",fill),
    intern=TRUE)," ")[[1]][1])
block = 1000
arrdelay = numeric(1000000)
nplanes =0
passes = round(nlin/block)
# for(i in 1:passes){            # very slow with only
    1000 planes!!
        plan = readLines(con,block)
```

```
        lapply(strsplit(plan,","), function(i) {
                if(i[17] == "LAX"){
                        nplanes = nplanes+1
                        arrdelay[nplanes] = i[15]
                }
                men = sum(arrdelay)/nplanes
                sdd = sum((arrdelay - men)^2)
                }
        )
                # return(sdd)
# }

#################################
### with DB's
#################################


library(ggplot2)
library(reshape2)
specs = data.frame(db = c("sqlite","mysql","monet"),
                   loadtimes = c(0,124.37,0),
                   avggroupby = c(10.694,6.4,0.136),
                   createindex = c(14.401,19.41,0.369),
                   avggroupbyWindex = c
                      (6.178,37.12,0.153),     # slowed
                       down all except sqlite?
                   unconditionalstdev = c
                      (2.076,5.17,0.255)
                   )
names(specs) = c("db","load time","group by pre index",
    "create index","group by post index","standard
   deviation")
tim = melt(specs[,-2],id.vars="db")
ggplot(tim, aes(variable,value,fill=db),xlab = "command
   ", ylab = "time", main = "Comparison between
   Databases for running same commands") +geom_bar(
   position="dodge",main = "test")
ggsave("bench.jpg")
```

```
#### SQLITE

library(RSQLite)
con = dbConnect(SQLite(),dbname="~/data/airports/delays
    .db")
ports = c("'LAX'","'OAK'","'SMF'","'SFO'")

## COUNTS
com = sprintf("SELECT COUNT(arrd) FROM delays WHERE
    origin = %s",ports)
lapply(com,function(i) fetch(dbSendQuery(con,i)))

## AVG
com = sprintf("SELECT AVG(arrd) FROM delays WHERE
    origin = %s",ports)
ret = lapply(com,function(i) fetch(dbSendQuery(con,i)))
avg = structure(as.numeric(unlist(ret)),names = ports)
   #

## SD
com = sprintf("SELECT SUM((arrd-%s)*(arrd-%s))/(COUNT(
    arrd)-1) FROM delays WHERE origin = %s",avg,avg,
    ports)
ret = lapply(com,function(i) fetch(dbSendQuery(con,i)))
sd = structure(sqrt(as.numeric(unlist(ret))),names =
    ports) #

## check
# cat 1987.csv | awk -F "," "{if ($17 == 'LAX') print
    $15}" > lax ..From shell
lax = read.table("lax")
sd(lax,na.rm=TRUE)       # 33 - agrees w shell but not
    sql
# made test table to check calculation
# Close but no cigar??
```

```
#### MySQL

library(RSQLite)
con = dbConnect(SQLite(),dbname="~/data/airports/delays
    .db")
ports = c("'LAX'","'OAK'","'SMF'","'SFO'")

## COUNTS
com = sprintf("SELECT COUNT(arrd) FROM delays WHERE
    origin = %s",ports)
lapply(com,function(i) fetch(dbSendQuery(con,i)))

## AVG
com = sprintf("SELECT AVG(arrd) FROM delays WHERE
    origin = %s",ports)
ret = lapply(com,function(i) fetch(dbSendQuery(con,i)))
avg = structure(as.numeric(unlist(ret)),names = ports)
   #

## SD
com = sprintf("SELECT SUM((arrd-%s)*(arrd-%s))/(COUNT(
    arrd)-1) FROM delays WHERE origin = %s",avg,avg,
    ports)
ret = lapply(com,function(i) fetch(dbSendQuery(con,i)))
sd = structure(sqrt(as.numeric(unlist(ret))),names =
    ports) #

## check
# cat 1987.csv | awk -F "," "{if ($17 == 'LAX') print
    $15}" > lax ..From shell
lax = read.table("lax")
sd(lax,na.rm=TRUE)       # 33 - agrees w shell but not
    sql
# made test table to check calculation
# Close but no cigar??
```

```
#### Monetdb

library(RSQLite)
con = dbConnect(SQLite(),dbname="~/data/airports/delays
    .db")
ports = c("'LAX'","'OAK'","'SMF'","'SFO'")

## COUNTS
com = sprintf("SELECT COUNT(arrd) FROM delays WHERE
    origin = %s",ports)
lapply(com,function(i) fetch(dbSendQuery(con,i)))

## AVG
com = sprintf("SELECT AVG(arrd) FROM delays WHERE
    origin = %s",ports)
ret = lapply(com,function(i) fetch(dbSendQuery(con,i)))
avg = structure(as.numeric(unlist(ret)),names = ports)
   #

## SD
com = sprintf("SELECT SUM((arrd-%s)*(arrd-%s))/(COUNT(
    arrd)-1) FROM delays WHERE origin = %s",avg,avg,
    ports)
ret = lapply(com,function(i) fetch(dbSendQuery(con,i)))
sd = structure(sqrt(as.numeric(unlist(ret))),names =
    ports) #

## check
# cat 1987.csv | awk -F "," "{if ($17 == 'LAX') print
    $15}" > lax ..From shell
lax = read.table("lax")
sd(lax,na.rm=TRUE)       # 33 - agrees w shell but not
    sql
# made test table to check calculation
# Close but no cigar??
```

7.2 shell

```
#!/bin/bash
# How to pass in airline names as argument to script?
case "$1" in
        count)
                lax=`cat $2 | awk -F ',' '{if ($17 == "
                   LAX" ) s+=$15} END {print s}'`
                oak=`cat $2 | awk -F ',' '{if ($17 == "
                   OAK" ) s+=$15} END {print s}'`
                smf=`cat $2 | awk -F ',' '{if ($17 == "
                   SMF" ) s+=$15} END {print s}'`
                sfo=`cat $2 | awk -F ',' '{if ($17 == "
                   SFO" ) s+=$15} END {print s}'`
                echo "lax" $lax
                echo "oak" $oak
                echo "smf" $smf
                echo "sfo" $sfo
                ;;
        mean)
                lax=`cat $2 | awk -F ',' '{if ($17 == "
                   LAX" ) {sum+=$15;nrec+=1}} END {
                    print sum/nrec}'`
                oak=`cat $2 | awk -F ',' '{if ($17 == "
                   OAK" ) {sum+=$15;nrec+=1}} END {
                    print sum/nrec}'`
                smf=`cat $2 | awk -F ',' '{if ($17 == "
                   SMF" ) {sum+=$15;nrec+=1}} END {
                    print sum/nrec}'`
                sfo=`cat $2 | awk -F ',' '{if ($17 == "
                   SFO" ) {sum+=$15;nrec+=1}} END {
                    print sum/nrec}'`
                echo "lax" $lax
                echo "oak" $oak
                echo "smf" $smf
                echo "sfo" $sfo
                ;;
        sd)
```

```
                    lax=`cat $2 | awk -F ',' '{if ($17 == "
                       LAX" ) {sum+=$15;nrec+=1;array[nrec
                       ]=$15}} END {for(x=1;x<=nrec;x++)
                       sumsq+=(array[x]-(sum/nrec))^2 ;
                       print sqrt(sumsq/nrec)}'`
                    oak=`cat $2 | awk -F ',' '{if ($17 == "
                       OAK" ) {sum+=$15;nrec+=1;array[nrec
                       ]=$15}} END {for(x=1;x<=nrec;x++)
                       sumsq+=(array[x]-(sum/nrec))^2 ;
                       print sqrt(sumsq/nrec)}'`
                    smf=`cat $2 | awk -F ',' '{if ($17 == "
                       SMF" ) {sum+=$15;nrec+=1;array[nrec
                       ]=$15}} END {for(x=1;x<=nrec;x++)
                       sumsq+=(array[x]-(sum/nrec))^2 ;
                       print sqrt(sumsq/nrec)}'`
                    sfo=`cat $2 | awk -F ',' '{if ($17 == "
                       SFO" ) {sum+=$15;nrec+=1;array[nrec
                       ]=$15}} END {for(x=1;x<=nrec;x++)
                       sumsq+=(array[x]-(sum/nrec))^2 ;
                       print sqrt(sumsq/nrec)}'`
                    echo "lax" $lax
                    echo "oak" $oak
                    echo "smf" $smf
                    echo "sfo" $sfo
                    ;;
esac

#!/bin/sh
bzip2 -dc 1987.csv.bz2 # uncompress to stout
awk -F "," '{print $17}' samp.csv       # get just
   origin
grep LAX        # match airline
sc -l   # get # lines

cat samp.csv | awk -F "," '{print $17}'| grep 'LAX\|OAK
   \|SMF' | wc -l   # only gives total - need split by
   airport.
```

```
cat 1987.csv | awk -F "," '{print $17}'| grep 'LAX\|OAK
   \|SMF' | sort | uniq -c #DONE

time bzip2 -dc 1987.csv.bz2 | awk -F "," '{print $17}'|
    grep 'LAX\|OAK\|SMF' | sort | uniq -c   # to time
   it.

cat ~/data/airports/open/1987.csv | awk -F ',' '{print
   $17}'| grep -E 'LAX|OAK|SMF|SFO' | awk -F ',' '{s+=
   $15} END {print s}' |

cat ~/data/airports/open/1987.csv | awk -F ',' '{s+=$15
   } END {print s}' # to get sum, but need only those
   from lax etc.

cat ~/data/airports/open/1987.csv | awk -F ',' '{if (
   $17 == "LAX"||$17 == "OAK") s+=$15} END {print s}' #
    working but this is sum of all airports - need to
   do one at a time - no way to combine?

 cat ~/data/airports/open/1987.csv | awk -F ',' '{if (
    $17 == "LAX") l+=$15; else if ($17 == "OAK") o+=$15
    } END {print l,o}'    # can get multiple out.
```

7.3 Amazon RDS and EC2