**Stats 781 Project - HMR Package Enhancement**

Introduction

Description of Linux, Bash, VM

Description of Hadoop & HDFS

Description of HMR high performance map reduce package using iotools

Store data in environments

Processing efficiently by reading in binary

1st Task – Detection of the formatter

Take input into R and infer types

If the formatter is omitted then the format is taken from input object (if it has one) or the default formatter (mstrsplit with '\t' as key separator, '|' as column separator) is used. If formatter is a function then the same formatter is used for both the map and reduce steps. If separate formatters are required, the formatter can be a list with the entries map and/or reduce specifying the corresponding formatter function.

**Two approaches:**

1. Advance detection approach.

#formatter function

guess <- function(path, sep='|', nsep='\t', nrowsClasses=25L, header=TRUE) {

f <- pipe(paste("hadoop fs -cat", path), "rb")

cr <- chunk.reader(f)

r <- read.chunk(cr, 1e6)

subset = mstrsplit(r, sep=sep, nsep=nsep, nrows=nrowsClasses, skip=header)

colClasses = rep(NA\_character\_, ncol(subset))

for (i in 1:ncol(subset))

colClasses[i] = class(type.convert(subset[,i],as.is=TRUE))

# If all NA's, R makes it logical; better to be character

index = which(apply(!is.na(subset), 2, sum) == 0)

if (length(index))

colClasses[index] = "character"

if (header) {

col\_names = mstrsplit(r, sep=sep, nsep=nsep, nrows=1)

if ((length(col\_names) - 1 == length(colClasses)) && !is.na(nsep))

col\_names = col\_names[-1]

names(colClasses) = col\_names

}

close(f)

rm(list=c("cr", "r", "f"))

function(x) dstrsplit(x, colClasses, sep=sep, nsep=nsep, skip=header)

}

Do the detection prior to processing map reduce. Take a chunk of the input (i.e. 1mb) and detect the format of the data and output as a function with the format included. Function is then passed to the formatter in HMR.

Advantages:

Only require one formatter across the process.

Can adjust the format for different data structure/separators

1. Dynamic detection approach.

colClassses = function(path, sep=",", nsep=NA, header=TRUE) {

r <- readBin(path, raw(), 1e6)

nr <- r[1:tail(which(r==10),1)]

# convert into character matrix

s = mstrsplit(nr, sep=sep, skip=header)

# vectorized computation to find column type

cols = apply(s, 2, function(x) class(type.convert(x, as.is = TRUE))) # type convert for date/time?

if (header) {

col\_names = mstrsplit(r, sep=sep, nrows=1)

if ((length(col\_names) - 1 == ncol(cols)) && !is.na(nsep))

col\_names = col\_names[-1]

names(cols) = col\_names

}

rm(list=c("nr", "r")) # necessary?

#dstrsplit(path, cols, sep=sep, nsep=nsep, header=TRUE)

list(cols) # or return?

}

Do the detection of the format during the map reduce process. Read a section of the input to detect the format of the data and output a list of column types for the input data. The list is passed to the specified function dstrsplit in the HMR formatter.

Advantages:

Won’t run into header issue if there are multiple files in path.

Can auto detect format for multiple files in path

Issues with different file structures – can only specify one sep?