bench.fct <- function(N = 2500000, type.file = 'rds', type.hd = 'HDD') {

# Function for timing read and write operations

#

# INPUT: N - Number of rows in dataframe to be read and write

# type.file - format of output file (rds, csv, fst)

# type.hd - where to save (hdd or ssd)

#

# OUTPUT: A dataframe with results

require(tidyverse)

require(fst)

my.df <- data\_frame(x = runif(N),

char.vec = sample(letters, size = N,

replace = TRUE))

path.file <- switch(type.hd,

'SSD' = '~',

'HDD' = '/mnt/HDD/')

my.file <- file.path(path.file,

switch (type.file,

'rds-base' = 'temp\_rds.rds',

'rds-readr' = 'temp\_rds.rds',

'fst' = 'temp\_fst.fst',

'csv-readr' = 'temp\_csv.csv',

'csv-base' = 'temp\_csv.csv'))

if (type.file == 'rds-base') {

time.write <- system.time(saveRDS(my.df, my.file, compress = FALSE))

time.read <- system.time(readRDS(my.file))

} else if (type.file == 'rds-readr') {

time.write <- system.time(write\_rds(x = my.df, path = my.file, compress = 'none'))

time.read <- system.time(read\_rds(path = my.file ))

} else if (type.file == 'fst') {

time.write <- system.time(write.fst(x = my.df, path = my.file))

time.read <- system.time(read\_fst(my.file))

} else if (type.file == 'csv-readr') {

time.write <- system.time(write\_csv(x = my.df, path = my.file))

time.read <- system.time(read\_csv(file = my.file, col\_types = cols(x = col\_double(),

char.vec = col\_character())))

} else if (type.file == 'csv-base') {

time.write <- system.time(write.csv(x = my.df, file = my.file))

time.read <- system.time(read.csv(file = my.file))

}

# clean up

file.remove(my.file)

# save output

df.out <- data\_frame(type.file = type.file,

type.hd = type.hd,

N = N,

type.time = c('write',

'read'),

times = c(time.write[3],

time.read[3]))

return(df.out)

}

Now that we have my function, its time to use it for all combinations  
between number of rows, the formats of the file and type of drive:

library(purrr)

df.grid <- expand.grid(N = seq(1, 500000, by = 50000),

type.file = c('rds-readr', 'rds-base', 'fst', 'csv-readr', 'csv-base'),

type.hd = c('HDD', 'SSD'), stringsAsFactors = F)

l.out <- pmap(list(N = df.grid$N,

type.file = df.grid$type.file,

type.hd = df.grid$type.hd), .f = bench.fct)

df.res <- do.call(what = bind\_rows, args = l.out)

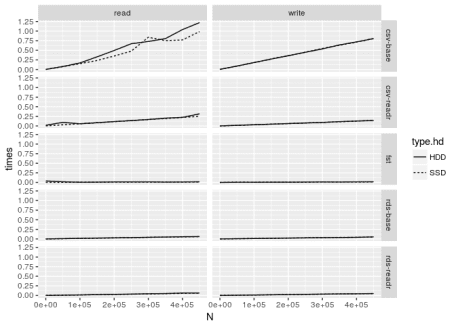
Lets check the result in a nice plot:

library(ggplot2)

p <- ggplot(df.res, aes(x = N, y = times, linetype = type.hd)) +

geom\_line() + facet\_grid(type.file ~ type.time)

print(p)



As you can see, the csv-base format is messing with the y axis. Let’s  
remove it for better visualization:

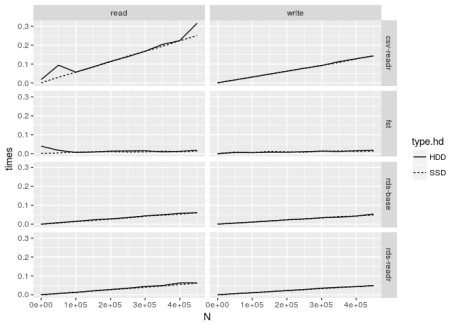
library(ggplot2)

p <- ggplot(filter(df.res, !(type.file %in% c('csv-base'))),

aes(x = N, y = times, linetype = type.hd)) +

geom\_line() + facet\_grid(type.file ~ type.time)

print(p)



When it comes to the file format, we learn:

* **By far, the fst format is the best**. It takes less time to read  
  and write than the others. However, it’s probably unfair to compare  
  it to csv and rds as it uses many of the 16 cores of my  
  computer.
* **readr is a great package for writing and reading *csv* files**.  
  You can see a large difference of time from using the base  
  functions. This is likely due to the use of low level functions to  
  write and read the text files.
* **When using the *rds* format, the base function do not differ much  
  from the readr functions**.

As for the effect of using SSD, its clear that it **DOES NOT** effect  
the time of reading and writing. The differences between using HDD and  
SSD looks like noise. Seeking to provide a more robust analysis, let’s  
formally test this hypothesis using a simple t-test for the means:

tab <- df.res %>%

group\_by(type.file, type.time) %>%

summarise(mean.HDD = mean(times[type.hd == 'HDD']),

mean.SSD = mean(times[type.hd == 'SSD']),

p.value = t.test(times[type.hd == 'SSD'],

times[type.hd == 'HDD'])$p.value)

print(tab)

## # A tibble: 10 x 5

## # Groups: type.file [?]

## type.file type.time mean.HDD mean.SSD p.value

##

## 1 csv-base read 0.554 0.463 0.605

## 2 csv-base write 0.405 0.405 0.997

## 3 csv-readr read 0.142 0.126 0.687

## 4 csv-readr write 0.0711 0.0706 0.982

## 5 fst read 0.015 0.0084 0.0584

## 6 fst write 0.00900 0.00910 0.964

## 7 rds-base read 0.0321 0.0303 0.848

## 8 rds-base write 0.0253 0.025 0.969

## 9 rds-readr read 0.0323 0.0304 0.845

## 10 rds-readr write 0.0251 0.0247 0.957

As we can see, the null hypothesis of equal means easily fails to be  
rejected for almost all types of files and operations at 10%. The  
exception was for the *fst* format in a reading operation. In other  
words, statistically, it does not make any difference in time from using  
SSD or HDD to read or write files in different formats.

I am very surprised by this result. Independently of the type of format,  
I expected a large difference as SSD drives are much faster within an  
OS. Am I missing something? Is this due to the OS being in the SSD? What  
you guys think?