R Performance (it's not R, it's you)

Tim Hoolihan

Presented September 2020

Bio

- Tim Hoolihan
- Local Service Offering Lead of Data & Analytics @ Centric Consulting
- Organizer of the Cleveland R User Group: **meetup.com/Cleveland-UseR-Group**
- Created a series of R machine learning videos: **packtpub.com/all?search=Hoolihan**

((CENTRIC))

Centric Consulting provides business and technical consulting services with unmatched client and employee experiences

- I am hiring a Sr Data Architect for the Cleveland Practice
  - https://cutt.ly/wfTo1AW

Purpose

R is often unfairly maligned for being slow, and not being able to deal with big data. This may have to do with the number of non-traditional programmers that right code in R due to support of various domains. Let's talk about:

- Tools for measuring performance
- Common performance issues
- Dealing with Large Data Sets
- Other tips

Time a process

```
start <- Sys.time()
Sys.sleep(0.5)
end <- Sys.time()
end - start
```

```
Time difference of 0.507865 secs
```

```
system.time({system("du -shc ~/workspace")})
```

```
   user  system elapsed
  0.043   0.765   1.736
```

What is user time? system time?

"User CPU time" gives the CPU time spent by the current process (i.e., the current R session) and "system CPU time" gives the CPU time spent by the kernel (the operating system) on behalf of the current process. The operating system is used for things like opening files, doing input or output, starting other processes, and looking at the system clock: operations that involve resources that many processes must share. Different operating systems will have different things done by the operating system.

- William Dunlap on R-Help Mailing List

Using rbenchmark package

RBenchmark Package

```
library(rbenchmark)

1:25
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13
14 15 16 17 18 19 20 21 22 23 24 25
```

```
benchmark((1:25)^2, replications = 500)
```

```
      test replications elapsed relative
user.self sys.self user.child
1 (1:25)^2          500   0.002        1
0.001        0           0
  sys.child
1         0
```

Benchmarking

Demo here

Other benchmarking alternatives

- tictoc package
- microbenchmark

Memory profiling

More info **here**

```
library(pryr)
```

Scaler (single value)

```
a = 1
object_size(a)
```

```
56 B
```

Vector (multiple values)

```
b = 1:1000
object_size(b)
```

```
4.05 kB
```

Memory profiling (cont)

All Objects

```
ls()
```

```
[1] "a"     "b"     "end"   "start"
```

```
sapply(ls(), function (x)
object_size(eval(parse(text=x))))
```

```
    a     b   end start
   56  4048   344   344
```

Memory Management

Demo

A common performance problem

“If you wrote a for loop in R, you're most likely doing it wrong”

- Me

Vectors

Q: What is a vector in R?

A: Everything

```
i <- 9
print(paste('class:', class(i),
            'i[0]:', i[0],
            'i[1]:', i[1],
            'i[2]:', i[2]))
```

```
[1] "class: numeric i[0]:  i[1]: 9 i[2]: NA"
```

Q: What?

Vectorize operations

In other words, use R functions that operate on an entire vector of values, as opposed to looping through values and applying your function. Even if the underlying C code is using a for loop, trust that it is optimized to minimize the number of allocation steps.

> Vectorising is about taking a "whole object" approach to a problem, thinking about vectors, not scalars.
>
> - Hadley Wickham in ***Advanced R***

Well summarized in this **blog post by Noam Ross: http://bit.ly/RpBOYe**

Vectorize how?

```r
# loop
grades <- c(85, 90, 72)
for(g in grades) {
  print(g / 100)
}
```

```
[1] 0.85
[1] 0.9
[1] 0.72
```

```r
# vectorized division
print(grades / 100)
```

```
[1] 0.85 0.90 0.72
```

Vectorize operations

Demo

Isn't there a more difficult way?

Since the solution seems to involve using R functions that are implemented in C/C++…

RCPP

RCPP

Demo

RCPP Disclaimer

Compressed data

```
fh <- gzfile("data/flights14.csv.gz", open = "rt")
df <- read.csv(fh)
head(df)
```

```
  year month day dep_time dep_delay arr_time arr_delay cancelled carrier
1 2014     1   1      914        14     1238        13         0      AA
2 2014     1   1     1157        -3     1523        13         0      AA
3 2014     1   1     1902         2     2224         9         0      AA
4 2014     1   1      722        -8     1014       -26         0      AA
5 2014     1   1     1347         2     1706         1         0      AA
6 2014     1   1     1824         4     2145         0         0      AA
  tailnum flight origin dest air_time distance hour min
1  N338AA      1    JFK  LAX      359     2475    9  14
2  N335AA      3    JFK  LAX      363     2475   11  57
3  N327AA     21    JFK  LAX      351
```

```
2475   19   2
4  N3EHAA     29    LGA  PBI      157
1035    7  22
5  N319AA    117    JFK  LAX      350
2475   13  47
6  N3DEAA    119    EWR  LAX      339
2454   18  24
```

data.table

CRAN package page

Description: “Fast aggregation of large data (e.g. 100GB in RAM), fast ordered joins, fast add/modify/delete of columns by group using no copies at all, list columns, a fast friendly file reader and parallel file writer. Offers a natural and flexible syntax, for faster development.”

data.table

Demo

data.table takes a shell command

Filter csv columns when loading

```
library(data.table)
dt <- fread("cut -f1,2,3,9 -d','
data/flights14.csv")
dt
```

```
        year month day carrier
     1: 2014     1   1      AA
     2: 2014     1   1      AA
     3: 2014     1   1      AA
     4: 2014     1   1      AA
     5: 2014     1   1      AA
    ---
253312: 2014    10  31      UA
253313: 2014    10  31      UA
253314: 2014    10  31      MQ
253315: 2014    10  31      MQ
253316: 2014    10  31      MQ
```

Chunking with ff, ffbase

When you're data won't fit in memory, but is still reasonable to work on a disk with.

- ff **cran.r-project.org/package=ff**
- ffbase **cran.r-project.org/package=ffbase**
  - adds functionality to ff objects

Demo

I have cores...

parallel Part of base R as of 2.14

Description: “Support for parallel computation, including by forking (taken from package multicore), by sockets (taken from package snow) and random-number generation.”

Parallel

Demo

Deep learning with a GPU

- **tensorflow R package**
  - **Talk by Bryan Lewis at CRUG**
- **keras R package**
  - **Talk by Tim Hoolihan at CRUG**
- Why not just use Python?

My data has it's own zip code...

Big Data comes into play when the CPU time for the calculation takes longer than the cognitive process of designing a model.

(Paraphrased) Hadley Wickham

- **Spark**
  - **Sparklyr**
- **Hadoop**
  - **r2mr**
  - **RHipe**
- **h20**

Asking for help

- **stackoverflow.com/questions/tagged/r**
- **community.rstudio.com**
- **R mailing list**
- #R on freenode irc
- Cleveland R User Group: **meetup.com/Cleveland-UseR-Group/**
- **How to Write a Reproducable Example - by Hadley Wickham**
  - **http://bit.ly/2FkmXIh**

Questions? / Contact

- Slides: **github.com/thoolihan/RPerformance**
- Twitter: **@thoolihan**
- Homepage: **hoolihan.net**
- Blog: **hoolihan.net/blog-tim**
- Videos: **youtube.com/c/TimHoolihan**
- ML Videos: **packtpub.com/all?search=Hoolihan**
- Cleveland R User Group: **meetup.com/Cleveland-UseR-Group/**
- Cleveland R User Group Talks: **youtube.com/channel/UC7C4YZ-9itQW7Nl4RVKDflg**