Building intuition for performance

Roel Janssen

November 17, 2023

Wait a second

How much can a computer do in a second?

https://computers-are-fast.github.io/

Some numbers

execute typical instruction	1/1,000,000,000 sec = 1 nanosec
fetch from L1 cache memory	0.5 nanosec
branch misprediction	5 nanosec
fetch from L2 cache memory	7 nanosec
Mutex lock/unlock	25 nanosec
fetch from main memory	100 nanosec
send 2K bytes over 1Gbps network	20,000 nanosec
read 1MB sequentially from memory	250,000 nanosec
fetch from new disk location (seek)	8,000,000 nanosec
read 1MB sequentially from disk	20,000,000 nanosec
send packet US to Europe and back	150 milliseconds = 150,000,000 nanosec

Big-O notation

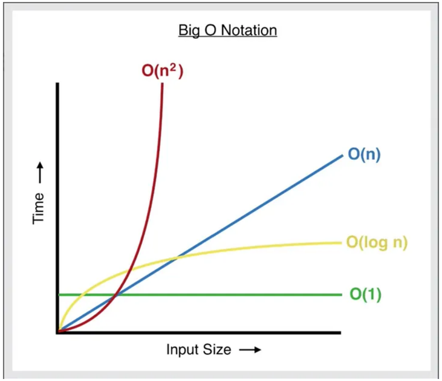


Figure: *Image source: medium.com/dataseries/how-to-calculate-time-complexity-with-big-o-notation-9afe33aa4c46*

Intuition test

```r
simple_call_1 <- function (number) { return (number * number) }
sapply(numbers_1, simple_call_1)

simple_call_2 <- function (numbers) {
    for (index in 1:length(numbers)) {
        numbers[index] <- numbers[index] * numbers[index]
    }
    return (numbers)
}
```

Intuition test

```r
numbers_1 <- runif(1000000) * 100
numbers_2 <- numbers_1

object.size(numbers_2)
# => 8000048 bytes
```

Execution time and space (1/2)

```
library(bench)
res_sapply <- bench::mark(numbers_3 <- sapply(numbers_1, simple_call_1))
res_for <- bench::mark(numbers_4 <- simple_call_2 (numbers_2))

res_sapply
res_for
```

Execution time and space (2/2)

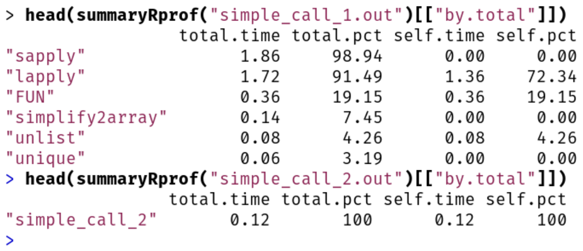
	memory allocated	**median time spent**
`simple_call_1`	30.9MB	2.37s
`simple_call_2`	7.65MB	136ms

Function call profiling (1/2)

```
Rprof("simple_call_1.out")
numbers_3 <- sapply(numbers_1, simple_call_1)
Rprof("simple_call_2.out")
numbers_4 <- simple_call_2 (numbers_2)
Rprof(NULL)

head(summaryRprof("simple_call_1.out")[["by.total"]])
head(summaryRprof("simple_call_2.out")[["by.total"]])
```

Function call profiling (2/2)



```
> head(summaryRprof("simple_call_1.out")[["by.total"]])
                 total.time total.pct self.time self.pct
"sapply"               1.86     98.94      0.00     0.00
"lapply"               1.72     91.49      1.36    72.34
"FUN"                  0.36     19.15      0.36    19.15
"simplify2array"       0.14      7.45      0.00     0.00
"unlist"               0.08      4.26      0.08     4.26
"unique"               0.06      3.19      0.00     0.00
> head(summaryRprof("simple_call_2.out")[["by.total"]])
                total.time total.pct self.time self.pct
"simple_call_2"       0.12       100      0.12      100
>
```

Memory allocation profiling (1/3)

```r
library(profmem)
capabilities("profmem") # Check if your R can do memory profiling

options(profmem.threshold = 0)
mem_details_sapply <- profmem({ numbers_3 <- sapply(numbers_1, simple_call_1) })
mem_details_for    <- profmem({ numbers_4 <- simple_call_2 (numbers_2) })

mem_details_sapply
mem_details_for
```

Memory allocation profiling (2/3)

```
> mem_details_sapply
...

Memory allocations:
Number of 'new page' entries not displayed: 9098
       what     bytes                                                      calls
1      alloc   8000048                                         sapply() -> lapply()
9100   alloc   4000048         sapply() -> simplify2array() -> unique() -> lengths()
9101   alloc   8388656 sapply() -> simplify2array() -> unique() -> unique.default()
9102   alloc   4000048 sapply() -> simplify2array() -> unique() -> unique.default()
9103   alloc   8000048                     sapply() -> simplify2array() -> unlist()
total         32388848
```

Memory allocation profiling (3/3)

```
> mem_details_for
...
Memory allocations:
       what   bytes           calls
1     alloc 8000048 simple_call_2()
total       8000048
> object.size(numbers_3)
8000048 bytes
>
```

More reading

Hadley Wickham on performance in R

http://adv-r.had.co.nz/Performance.html