How to speed up R code

Fibears

2015 年 10 月 17 日

Content

Abstract

- **R** is more and more popular in some fields such as Statistics. But the users then faces total execution times of his codes that are hard to work with: hours,days, even weeks.
- **Goal**: How to reduce the total execution time of your R codes

Chap 1.

- Your Computer
  - 1.1 HDD (The hard disk drive)
  - 1.2 RAM (The random-access memory)
  - 1.3 CPU (The central processing unit)

1.1 HDD

- The hard disk drive is where all the data of the computer is stored,even when it is powered down.
- Cheap to produce
- But,they are slow!

```
system.time({
    data <- rnorm(10^7)
    write.table(data, "data.txt")
})
```

1.1 HDD

- This file *data.txt* was 280.5 MBs,so we can roughly infer that a 4-GB dataset will require 500 seconds to be written down on a HDD.
- A rule when writing R code: only access the HDD at the beginning of your code(to load some data) and at the end(to save some results).
- Use SSD: You can store files you have to access quite often on the small SSD and files of lesser interest on the large HDD.

1.2 RAM

- The random-access memory is also used to store files.
  - read.table(): The dataset is copied from the HDD(or SSD) and pasted on the RAM.
  - RAM is a much faster data storage technology, but everything stored on the RAM is lost when the computer is powered down.
  - Expensive
- 32-bit R or 64-bit R?
  - 32-bit programs cannot use more than 2 GBs of RAM.
  - 64-bit programs are allowed to use as much as RAM as they need.

```
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin13.4.0 (64-bit)
```

1.3 CPU

- CPU: The central processing unit–the heart of your computer
- Multi-core: **R** is not natively able to use several cores at the same time! But,some packages allowing **R** to use more than one core at the same time.

Chap 2.

- One R instance, one core
  - 2.1 Writing your code right
  - 2.2 What part of my code is slow?
  - 2.3 Writing part of your code in C

2.1 Writing your code right

Two ways to measure how fast your code currently run.

```
time.start <- proc.time()[[3]]
#### Your code ####
time.end <- proc.time()[[3]]
time.end - time.start
```

```
system.time({
    ### Your code ###
})[[3]]
```

2.1 Writing your code right

- R works better with vectors, matrics and tables. Loops are, in general, more expensive, as well as dataframes and lists.
  - Example 1.

```
n <- 10000
system.time({
a <- vector(length=n); for(i in 1:n) a[i]=2*pi*sin(i)
})
```

```
##    user  system elapsed
##    0.05    0.00    0.06
```

```
system.time({
b <- NULL; for(i in 1:n) b[i]=2*pi*sin(i)
})
```

```
##    user  system elapsed
##    0.55    0.00    0.55
```

2.2 What part of my code is slow?

2.3 Writing part of your code in C

Chap 3.

- Using the others cores of your computer
  - 3.1 Several R instances, one core for each
  - 3.2 One R instance, several cores

3.1 Several R instances, one core for each

3.2 One R instance, several cores