STAT 242

Assignment 4

Zhewen HU

912490355

Introduction

In this assignment, we should write a function to move cars in C language. In this assignment, we will know how to use C to improve the bottleneck of R code. Also, while compiling C or calling C in R, we will know how to handle some problems when using C. And we will know the bad side and good side of using U.

Write crunBMLGrid() in C

For solving this problem, we will focus on two steps. First we should find the cars we need to move. The second is make them move simultaneously. If we move them sequentially, it will waste a lot of time. For the first step, we will use R function to get the location of the red or blue cars, then we update the location. In this way, we can find the car quickly instead of looping over all the cells to locate the car. For the second step, we use two grids in our computations. We set one original grid and one updated grid. When finishing moving all the cars in a literation, we update the original grid.

Comparison of Performance ( Function with C code vs Original Funciton)

We first test the crunBML() to see the system time it will consume. We simply test the same sized grid (100\*100, density = 0.5). It costs 38.1s, which is much faster than R code.

Test Different Grid Size

Test Different Density

Discussion

Is it worth rewriting the createBMLGrid() in C language. I think it is not worth it. First, it is not the bottleneck for our R function. Second, it is not that easy for me to write C function, for I am totally new to C. It is not easy for me to figure out the grammar of C. Also, there are some small details which are quite tough for me. (I have listed in the next section.)

Problems I have faced

1. Compile in windows
2. Is.loaded
3. .dll or .so
4. Learn too many programming language at the same time?
5. Statistics and Computer science
6. U take time to learn something new and make the process faster.
7. Importance of version control. I mess up when I have several versions.

In addition: Warning messages:

1: In matrix("", r, c) :

Reached total allocation of 3915Mb: see help(memory.size)

2: In matrix("", r, c) :

Reached total allocation of 3915Mb: see help(memory.size)

3: In matrix("", r, c) :

Reached total allocation of 3915Mb: see help(memory.size)

4: In matrix("", r, c) :

Reached total allocation of 3915Mb: see help(memory.size)

> summaryRprof(tmp)

$by.self

self.time self.pct total.time total.pct

"!=" 4219.48 32.59 4219.48 32.59

"structure" 3597.90 27.79 3598.22 27.79

"crunBMLGrid" 1556.70 12.02 12947.38 100.00

"==" 1127.66 8.71 1127.66 8.71

"getCarLocations" 760.22 5.87 8867.50 68.49

"matrix" 413.14 3.19 662.76 5.12

"sample.int" 284.50 2.20 284.54 2.20

"match" 249.56 1.93 249.60 1.93

"sample" 166.74 1.29 526.68 4.07

".C" 137.24 1.06 137.24 1.06

"row" 105.08 0.81 105.08 0.81

"col" 93.46 0.72 93.46 0.72

"cbind" 91.04 0.70 91.04 0.70

":" 75.28 0.58 75.28 0.58

"createBMLGrid" 67.72 0.52 853.20 6.59

"rownames" 0.30 0.00 0.42 0.00

"[[" 0.26 0.00 0.26 0.00

"c" 0.26 0.00 0.26 0.00

"nrow" 0.16 0.00 0.16 0.00

"Rprof" 0.14 0.00 0.26 0.00

".External" 0.12 0.00 0.12 0.00

"as.name" 0.12 0.00 0.12 0.00

"dimnames" 0.12 0.00 0.12 0.00

"getNamespace" 0.10 0.00 0.22 0.00

"length" 0.10 0.00 0.10 0.00

"sum" 0.10 0.00 0.10 0.00

"<Anonymous>" 0.08 0.00 0.08 0.00

"as.integer" 0.08 0.00 0.08 0.00

"class<-" 0.06 0.00 0.06 0.00

"gridInterger" 0.04 0.00 403.72 3.12

"is.atomic" 0.04 0.00 0.04 0.00

"\*" 0.02 0.00 0.02 0.00

"ceiling" 0.02 0.00 0.02 0.00

"class" 0.02 0.00 0.02 0.00

"dim" 0.02 0.00 0.02 0.00

"list" 0.02 0.00 0.02 0.00

"ncol" 0.02 0.00 0.02 0.00

"numeric" 0.02 0.00 0.02 0.00

$by.total

total.time total.pct self.time self.pct

"crunBMLGrid" 12947.38 100.00 1556.70 12.02

"getCarLocations" 8867.50 68.49 760.22 5.87

"!=" 4219.48 32.59 4219.48 32.59

"structure" 3598.22 27.79 3597.90 27.79

"==" 1127.66 8.71 1127.66 8.71

"createBMLGrid" 853.20 6.59 67.72 0.52

"matrix" 662.76 5.12 413.14 3.19

"sample" 526.68 4.07 166.74 1.29

"gridInterger" 403.72 3.12 0.04 0.00

"sample.int" 284.54 2.20 284.50 2.20

"match" 249.60 1.93 249.56 1.93

".C" 137.24 1.06 137.24 1.06

"row" 105.08 0.81 105.08 0.81

"col" 93.46 0.72 93.46 0.72

"cbind" 91.04 0.70 91.04 0.70

":" 75.28 0.58 75.28 0.58

"rownames" 0.42 0.00 0.30 0.00

"[[" 0.26 0.00 0.26 0.00

"c" 0.26 0.00 0.26 0.00

"Rprof" 0.26 0.00 0.14 0.00

"getNamespace" 0.22 0.00 0.10 0.00

"nrow" 0.16 0.00 0.16 0.00

".External" 0.12 0.00 0.12 0.00

"as.name" 0.12 0.00 0.12 0.00

"dimnames" 0.12 0.00 0.12 0.00

"length" 0.10 0.00 0.10 0.00

"sum" 0.10 0.00 0.10 0.00

"<Anonymous>" 0.08 0.00 0.08 0.00

"as.integer" 0.08 0.00 0.08 0.00

"class<-" 0.06 0.00 0.06 0.00

"%in%" 0.06 0.00 0.00 0.00

"is.atomic" 0.04 0.00 0.04 0.00

"\*" 0.02 0.00 0.02 0.00

"ceiling" 0.02 0.00 0.02 0.00

"class" 0.02 0.00 0.02 0.00

"dim" 0.02 0.00 0.02 0.00

"list" 0.02 0.00 0.02 0.00

"ncol" 0.02 0.00 0.02 0.00

"numeric" 0.02 0.00 0.02 0.00

$sample.interval

[1] 0.02

$sampling.time

[1] 12947.94



