OPTIMIZING R CODE

Jacalyn Huband

UVA Research Computing

What is Code Optimization?

- The process of making code more efficient (either with time or memory)

- Important caveats:

  - The correctness of the code must be preserved.
  - The code should run faster "on average".
  - There is a tradeoff between your efforts and the computer's efforts. (**Should you spend a week trying to make a program faster by 1 sec?**)

Vectorization

- **Avoid using loops.**

  Take advantage of applying operations to entire arrays/lists.

Before:

```
total <- 0.0

for (x in values){
  total = total + sin(x)^2 – 3*x
}
```

After:

```
total <- sum( sin(values)^2 – 3*values )
```

Tidyverse

- **Use *tidyverse* to manipulate data**

  tidyverse is a collection of packages designed specifically for ease of manipulating data. Designed by Hadley Wickham, it has a different approach to its syntax.

Before:

```
myData <- read.csv(filename)
quality <- sort(unique(myData$quality))

N <- length(quality) ;   avg_chl <- rep(0.0, N)
for (i in 1:N) {
  qual <- quality[i]
  ndx <- which(myData$quality == qual)
  avg_chl[i] <- mean(myData$chlorides[ndx])
}
df <- data.frame(quality, avg_chl)
```

After:

```
library(tidyverse)
 myData <- read_csv(filename)
 myData %>% group_by(quality) %>%
                        summarize(avg_chl = mean(chlorides))
```

Pre-allocate Memory

- **Don’t “grow” a list or array.**

  If you know the size and type of the list or array, reserve memory for the entire list/array before doing calculations.

Before:

```
y = c( )

for (x in values){
   if ( x > 0.5) {
      y  <- c(y, sin(x))
   } else {
       y <- c(y, cos(x))
   }
}
```

After:

```
y <-  rep(0.0, length(values);  i = 1

for (x in values){
   if ( x > 0.5) {
      y[i] <- sin(x)
   } else {
       y[i] <- cos(x)
   }
  i = i + 1
}
```

Avoid Overuse of Parentheses

- **R treats parentheses as function calls.**

  The contents inside parentheses are evaluated and stored in a special list structure.
  There is overhead for allocating the list, storing, and retrieving the results.

Before:

```
y <- ( (x)^2 + 3*( sin(x) ) )
```

After:

```
y <- x^2 + 3*sin(x)
```

Use Exponentials

- **Squaring a number is okay.**

  When squaring a value or expression, exponentiation is okay.

Before:

```
y <- x*x
z <- x*x*x*x
```

After:

```
y <- x^2
z <- (x^2)^2
```

ifelse

- **Avoid ifelse for vectors.**

  Although it is expected to be more concise, the ifelse statement can be less efficient than a more detailed approach.

Before:

```
x <- runif(numValues, min=1, max=20)

y <- ifelse(x > 10, 1, -1)
```

After:

```
x <- runif(numValues, min=1, max=20)

y <- rep(-1, length(x))
y[x > 10] <- 1
```

FINAL COMMENTS

Optimization Strategy

- During optimization, your goal is to minimize the amount of work the computer is required to do. The strategy we recommend is a two-step approach:

1. Write code that is efficient from the start (e.g., use vectorizations instead of loops)

2. After your code is debugged and working, try more aggressive optimization techniques (e.g., manipulating the mathematical formulas to reduce calls to built-in math functions).

Disadvantages?

- Optimizing code is time consuming

  Do not waste weeks optimizing code that will run once for 1 hour.

- Some optimizations can make the code harder to read and debug.

- Be aware that different architectures can respond in different ways.

  Just because code is optimized on your laptop does not necessarily mean that it is optimized on your colleague's computer.

- Some optimizations can adversely affect parallel scaling.

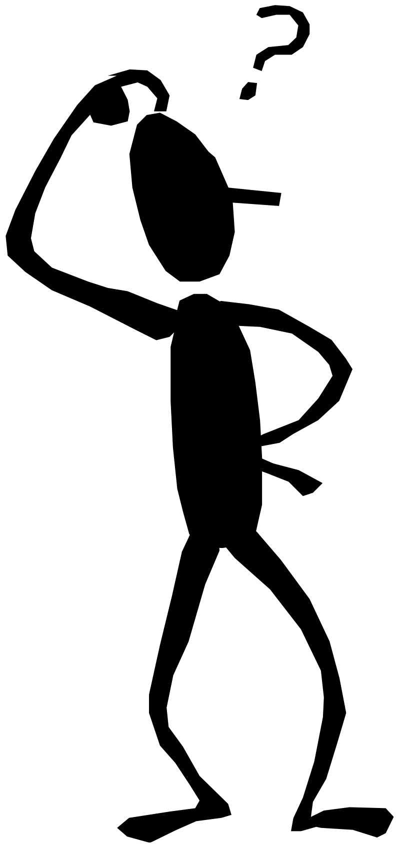
When to optimize?

- Code optimization is an iterative process requiring time, energy and thought. It is recommended for:
    - Codes that will be widely distributed and used often by the research community.
    - Projects that have limited allocation, so that you can maximize the available time on the compute resources.

When optimization isn’t enough

- When you have done everything possible to optimize your code, and it still isn’t fast enough, you can
  - Find a better algorithm (if one exists).
  - Look into parallelizing your code.

Questions?



University of Virginia
Advanced Research Computing Services