Manisha Siddartha Nalla

Assignment -1

R11471258
manisha-siddartha.nalla@ttu.edu

1 Mapply() Vs Loop

1.1 Definition :

mapply stands for multivariate apply. It performs operations on multiple lists/vectors/matrices...

1.2 Code description :

mapply takes input as two vectors, one contains number of times an element has to be repeated and other containes the elements that are to be repeated.

1.3 Example :

- i/p : vec1 = 1:3 and vec2 = 3:1
- o/p : 111 22 3

1.4 Code Snippet

github link : `http://tex.stackexchange.com`

```
c1<-1:10000
c2<-10000:1
f1<-mapply(rep,c1,c2)
store <-list()
f2<-function(c1,c2){
  for(i in 1:length(c1)){
    vec<-rep(NA,c2[i])
    store[[i]]<-list(rep(NA,c2[i]))
    x <- 1
    for(j in 1:c2[i]){
      vec[x]<-i
      x <- x+1
    }
    store[[i]] <- vec
    vec <- NULL
  }
  #print(store)
}
ggplot2::autoplot(microbenchmark(f1,f2,times=10L))
```

1.5 Performance Comparision

The diagram shows the plots of execution times for mapply()function (f1) and loop (f2) for 10 repetations. It is clear that mapply() reduces the execution time than loop for larger inputs.

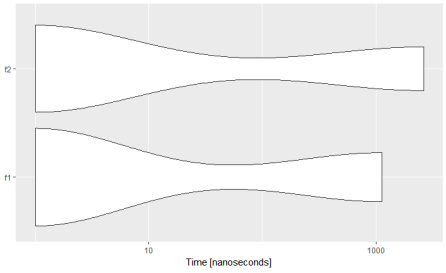

Figure 1:

2 Rapply() Vs Loop

2.1 Definition :

rapply stands for recursive apply, and as the name suggests it is used to apply a function to all elements of a list recursively.

2.2 Code description :

It squares all the elements in the list

2.3 Example :

- i/p : list1 = [1, 2, 3]
- o/p : 1 4 9

2.4 Code Snippet

github link : `http://tex.stackexchange.com`

```
li <- list(1:1000,1001:2000,2001:3000)
f1<- rapply(li, function(x) x^2, how = "list", classes = "integer")
f2<-function(li){
  for(i in li){
    i<-i^2
    #print(i)
  }
}
ggplot2::autoplot(microbenchmark(f1,f2,times=20L))
```

2.5 Performance Comparision

The diagram shows the plots of execution times for rapply()function (f1) and loop (f2) for 20 repetations. It is clear that rapply() reduces the execution time

than loop for larger inputs.

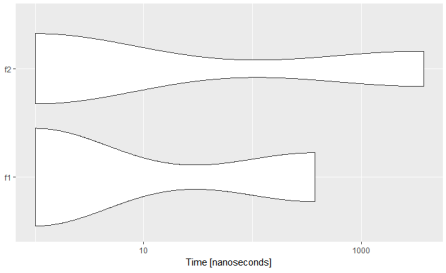

Figure 2:

3 tapply() Vs Loop

3.1 Definition :

Apply a function to each cell of a ragged array, that is to each (non-empty) group of values given by a unique combination of the levels of certain factors.

3.2 Code description :

Program calculates the elements at each level in factor to corresponding vector. That implies in the following code snippet 1st level contains elements 3,6,8 whose sum is 17

3.3 Example :

- i/p : vec1 = 1 2 3 4 5 6 7 8 9 10 fac = 2 3 1 3 2 1 3 1 3 2 , levels = 1 2 3
- o/p :
  1 2 3
  17 16 22

3.4 Code Snippet

github link : `http://tex.stackexchange.com`

```
vec <- 1:10000
fac <- factor(c(sample(1:10,10000,replace=T)))
f1 <- tapply(vec, fac, sum)
res <- c()
f2<- function(fac,vec){
        for(n in strtoi(levels(fac))){
              sum1<-c(0)
```

```
                    for(j in vec){
                        if(fac[j] == n)
                            sum1 <- sum(sum1,vec[j])
                    }
                    res[n] <- c(sum1)
        }
  #print(res)
}
ggplot2::autoplot(microbenchmark(f1,f2,times=15L))
```

3.5 Performance Comparision

The diagram shows the plots of execution times for tapply()function (f1) and loop (f2) for 15 repetations. It is clear that tapply() reduces the execution time than loop for larger inputs.

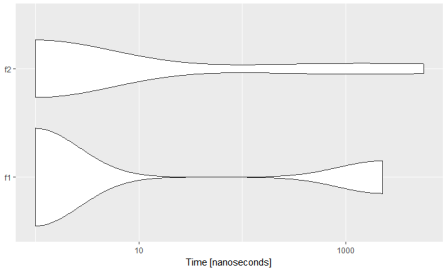

Figure 3: