

# STAT243-PS5

Jinhui Xu

October 2017

## 1 Other students

I discuss some problems with Weijie Xu

## 2 Question 2

### 2.1 (a)

First, we look at the format  $(-1)^S * 1.d * 2^{e-1023}$ . As d is represented as 52 bits and e is represented as 11 bits, we can write the format as

$$(-1)^S * 1.d_1 d_2 \dots d_{52} * 2^{e-1023} = (-1)^S * (1 + d_1 * 2^{-1} + d_2 * 2^{-2} \dots + d_{52} * 2^{-52}) * 2^{e-1023}$$

where  $S, d_i \in \{0, 1\}$  and  $(e - 1023) \in \{-1023, -1022, \dots, 1024, 1025\}$

Then let  $2^p - m$  is any one number among  $\{1, 2, 3, \dots, 2^{53} - 2, 2^{53} - 1\}$  where  $m \in \{1, 2, \dots, 2^p - 1\}$ . Obviously,  $S=0$  and we can find a group of  $d_i$  that makes

$$(1 + d_1 * 2^{-1} + d_2 * 2^{-2} \dots + d_{52} * 2^{-52}) * 2^{p-1} = 2^p - m$$

It is because the range of the left of the equation is  $[2^{p-1}, 2^p - 2^{p-53}]$  and the precision is  $2^{p-53}$

For example

$$2^{53} - 2 = (1 + 1 * 2^{-1} + \dots + 1 * 2^{-50} + 1 * 2^{-51} + 0 * 2^{-52}) * 2^{52}, \text{ which means } d_{52} = 0 \text{ and other } d_i = 1$$
$$2^{52} - 3 = (1 + 1 * 2^{-1} + \dots + 0 * 2^{-50} + 1 * 2^{-51} + 0 * 2^{-52}) * 2^{51}, \text{ which means } d_{50,52} = 0 \text{ and other } d_i = 1$$

### 2.2 (b)

If we want to store  $2^{53}, 2^{53} + 1, 2^{53} + 2, \dots$ , from (a), we can know that  $p=54$ . In this way, the precision equals to  $2^{p-53} = 2$

It is the same situation when we store  $2^{54}, 2^{54} + 1, 2^{54} + 2, \dots$ . In this situation,  $p=55$ . Then the precision equals to  $2^{p-53} = 4$

```
options(digits=22)
print(2^53-1)

## [1] 9007199254740991

print(2^53)

## [1] 9007199254740992

print(2^53+1)

## [1] 9007199254740992
```

We can find that  $2^{53} + 1$  is stored same as  $2^{53}$  in R. Because the precision is 2.

### 3 Question 3

#### 3.1 (a)

Copy two vectors and it is obviously that copying numeric vector takes more time. It is because that storing each int in R takes 4 bytes while storing each numeric in R takes 8 bytes. In plot, I use log function to the data.

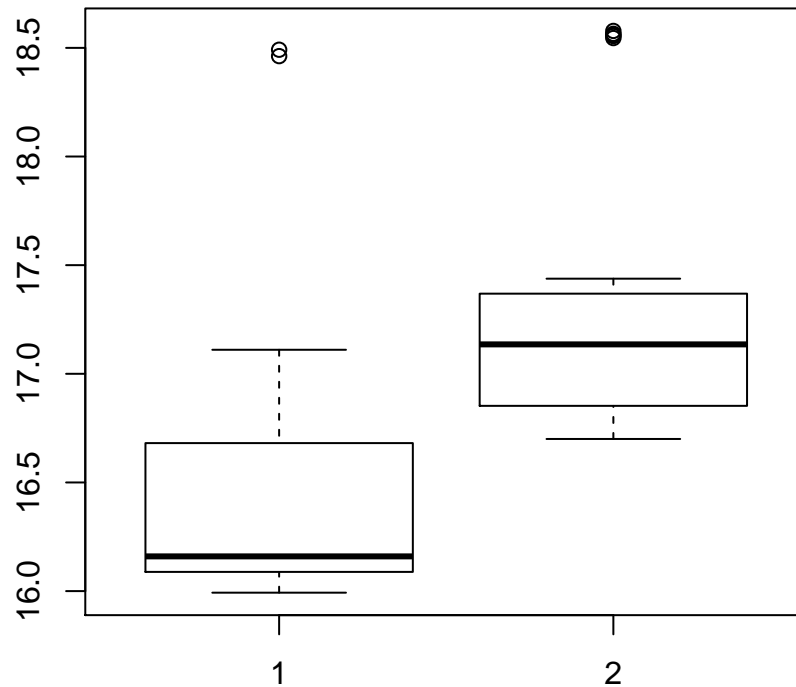
```
library(data.table)
library(microbenchmark)

#the function to compare the time used to copy two item.
compare_copytime=function(x,y){
  timex=microbenchmark(copy(x))$time      #get 100 data of time
  timey=microbenchmark(copy(y))$time
  if (mean(timex)>mean(timey)) print("copying the first one takes more time")
  else print('copying the second one cost more time')
  boxplot(log(timex),log(timey))
}
#generate the numeric vector
numvec<-rnorm(1e7)

#generate the integer vector
intvec<-as.integer(numvec)

#get the comparation and plot it
compare_copytime(intvec,numvec)

## [1] "copying the second one cost more time"
```



### 3.2 (b)

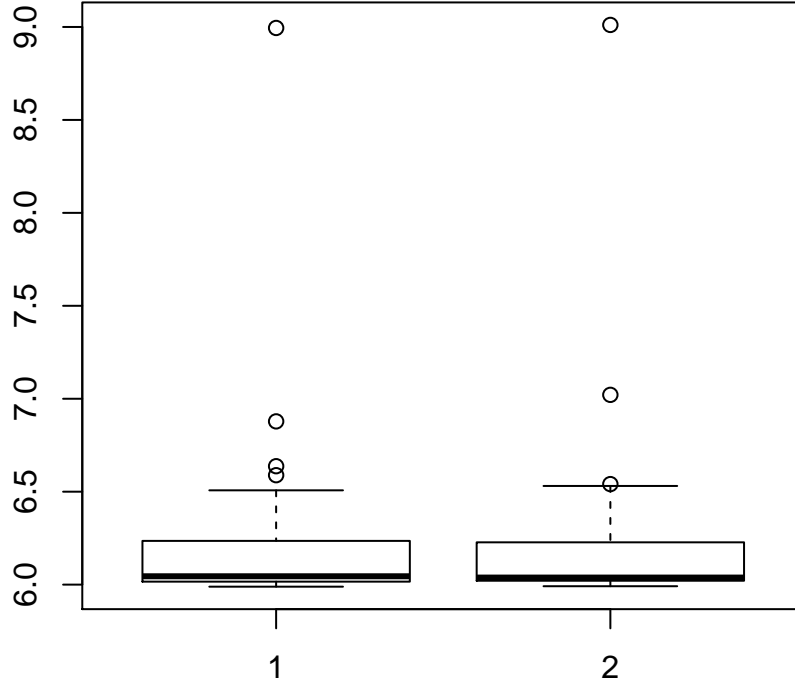
I find that the time used to take a subset of these two vectors is nearly same. I do several times comparison, the results are random.

```
#the function to compare the time used to take a subset of two items.

compare_subtime=function(x,y){
  timex=microbenchmark(xsub<-x[1:5*1e6])$time           #get 100 data of time
  timey=microbenchmark(ysub<-y[1:5*1e6])$time
  if (mean(timex)>mean(timey)) print("taking a subset of the first one takes more time")
  else print('taking a subset of the second one cost more time')
  boxplot(log(timex),log(timey))
}

#get the comparation and plot it
compare_subtime(intvec,numvec)

## [1] "taking a subset of the first one takes more time"
```



## 4 Question4

### 4.1 (a)

Breaking  $Y$  into  $n$  individual column-wise computations sometimes may not speed up a lot. Because if there is one task needs much more time than any other one, it still takes much time even though we breaking  $Y$  into  $n$  individual columns. In this way, it can not speed up a lot while it costs much more communication.

Therefore sometimes it is better to break up  $Y$  into  $p$  blocks of  $m = n/p$  columns rather than into  $n$  individual column-wise computations

### 4.2 (b)

1. In approach one, each worker deals with a  $\frac{n}{p} \times n$  matrix and a  $n \times n$  matrix, plus the memory used to store a  $\frac{n}{p} \times n$  result matrix. Then the total memory used in a single moment is

$$2 \times \frac{n}{p} \times n \times p + p \times n^2 = (2 + p)n^2$$

In approach two, each worker deals with two  $\frac{n}{p} \times n$  matrixes, plus the memory used to store a  $\frac{n}{p} \times \frac{n}{p}$

result matrix. Then the total memory used in a single moment is

$$2 \times \frac{n}{p} \times n \times p + \frac{n}{p} \times \frac{n}{p} \times p = 2n^2 + \frac{n^2}{p}$$

Therefore, the second approach use less memories.

2. In approach one, each worker takes in and out numbers only once and it takes in a  $n \times \frac{n}{p}$  matrix plus a  $n \times n$  matrix, and takes out a  $n \times \frac{n}{p}$  matrix. so the total number of numbers needed to be passed is

$$(2n \times \frac{n}{p} + n \times n) \times p = 2n^2 + pn^2$$

In approach two, each worker takes in and out numbers p times, it takes in two  $\frac{n}{p} \times n$  matrixes and takes out one  $\frac{n}{p} \times \frac{n}{p}$  matrix in one time. so the total number of numbers needed to be passed is

$$(2 \times \frac{n}{p} \times n + \frac{n}{p} \times \frac{n}{p}) \times p \times p = 2pn^2 + n^2$$

Therefore, the first approach is better for minimizing the communication.

## 5 Question5

From the question2, we know numerics are written as the following format:

$$(-1)^S * 1.d_1d_2...d_{52} * 2^{e-1023} = (-1)^S * (1 + d_1 * 2^{-1} + d_2 * 2^{-2}... + d_{52} * 2^{-52}) * 2^{e-1023}$$

I only discuss the situation the number is smaller than 1, without loss of generality.

It is obviously that only  $n = \sum_{i=1}^{52} a_i * 2^{-i}$  can be accurately stored.

And other numbers are stored as  $m = \sum_{i=1}^{52} a_i * 2^{-i}$ , where  $a_i$  minimizes  $\|m - \sum_{i=1}^{52} a_i * 2^{-i}\|$

Then we take example of 0.2+0.3. If  $0.2 = \sum_{i=1}^{52} a_i * 2^{-i}$ ,  $a_i$  are fixed there, it shows that  $a_i$  minimizes

$$\|0.2 - \sum_{i=1}^{52} a_i * 2^{-i}\|$$

And

$$\|0.2 - \sum_{i=1}^{52} a_i * 2^{-i}\| = \|0.3 - (2^{-1} - \sum_{i=1}^{52} a_i * 2^{-i})\|$$

So 0.3 is store as  $2^{-1} - \sum_{i=1}^{52} a_i * 2^{-i}$  in R. Then 0.2+0.3=0.5.

However, if the result can not be written as binary format, like 0.1+0.2=0.3. Then we can not guarantee that it is true.

```
0.2+0.3==0.5
```

```
## [1] TRUE
```

```
0.1+0.4==0.5
```

```
## [1] TRUE
```

```
0.1+0.2==0.3
```

```
## [1] FALSE
```