ps5

Hangyu Huang

17 October 2017

0.1 Partner

YUE HU

0.2 Problem 2

In this problem, the double precision floating point representations for some integers from 2^0 to 2^53 are shown.

From 2^53 to 2^54, the spacing between integers that can be represented exactly is 2.

From 2^54 to 2^55, the spacing between integers that can be represented exactly is 4.

```
library(pryr)
```

```
## Warning: package 'pryr' was built under R version 3.4.2
```

```
bits(2^53-1)
```

```
## [1] "01000011 00111111 11111111 11111111 11111111 11111111 11111111 11111111"
```

```
bits(2^53)
```

```
## [1] "01000011 01000000 00000000 00000000 00000000 00000000 00000000 00000000"
```

```
bits(2^53+1)
```

```
## [1] "01000011 01000000 00000000 00000000 00000000 00000000 00000000 00000000"
```

```
bits(2^53+2)
```

```
## [1] "01000011 01000000 00000000 00000000 00000000 00000000 00000000 00000001"
```

```
bits(2^54)
```

```
## [1] "01000011 01010000 00000000 00000000 00000000 00000000 00000000 00000000"
```

```
bits(2^54+1)
```

```
## [1] "01000011 01010000 00000000 00000000 00000000 00000000 00000000 00000000"
```

```
bits(2^54+2)
```

```
## [1] "01000011 01010000 00000000 00000000 00000000 00000000 00000000 00000000"
```

```
bits(2^54+3)
```

```
## [1] "01000011 01010000 00000000 00000000 00000000 00000000 00000000 00000001"
```

```
bits(2^54+4)
```

```
## [1] "01000011 01010000 00000000 00000000 00000000 00000000 00000000 00000001"
```

1: (-1)^0 x 1 x (2)^(1023-1023)

2: (-1)^0 x 1 x (2)^(1024-1023)

3: (-1)^0 x (1.1) x (2)^(1024-1023)

$2^{53-2:(-1)}$0 x (1.11...111110) x (2)^(1075-1023) # 11...11111: total 51 "1"

$2^{53-1:(-1)}$0 x (1.11...11111) x (2)^(1075-1023) # 11....111111: total 52 "1"

2^53: (-1)^0 x (1) x (2)^(1076-1023)

2^53+2: (-1)^0 x (1.000....00001) x (2)^(1076-1023) #000...0000: total 51 "0"

As the fraction has only 52bits,

From 2^52 to 2^53, the representable numbers are exactly the integers,total 2^52 number of intergers;

From 2^53 to 2^54, everything is multipulied by 2, so the representable numbers are all the even numbers, the spacing is 2, such as 2^53 = (2^52)x2, $2^{53+2=(2}$52+1)x2. Therefore, 2^53+1 cannot be represented,

From 2^54 to 2^55, everything is multipulied by 2^2=4, so the representable numbers are the numbers with spacing 4, such as 2^54 = (2^52)x4, 2^54+4= (2^52+1)x4.

0.3 Problem 3

(a) "system.time()"is used to compare the time used of copying a large vector of integers to the time of copying a large vector of numeric numbers.

Based on the results, it can be concluded that it is faster to copy a large vector of integers than a numeric vector of the same length in R.

(b) Based on the results, it can be concluded that it is slightly faster to take a subset of size k ??? n/2 from an integer vector of size n than from a numeric vector of size n.

```
#integer case
Int<- c(1:1e8)
object.size(Int)
```

```
## 400000040 bytes
```

```
IntNew <- Int
print(system.time(IntNew[40] <- 10L))
```

```
##    user  system elapsed
##    0.07    0.06    0.15
```

```
# As Intnew is now different to Int, a real vevtor of integers copy is made.
print(system.time(IntNew[1:5e7]))
```

```
##    user  system elapsed
##    0.34    0.07    0.40
```

```
#it shows the time to take a subset of size k ??? n/2 from an integer vector.

# numeric case
Numrc <- Int
print(system.time(Numrc[40] <- 10))
```

```
##    user  system elapsed
##    0.26    0.12    0.39
```

```
# As Numrc is now different to Int, a real vevtor of numeric numbers copy is made.
print(system.time(Numrc[1:5e7]))
```

```
##    user  system elapsed
##    0.27    0.14    0.41
```

```
# it shows the time to take a subset of size k ??? n/2 from a numeric vector of size n.
```

0.4 Problem 4

(a)

Both memory usage and the ammount of communication need to be considered. In this case, for many tasks that each take similar time, you want to preschedule the tasks to reduce communication. However, the communication overhead of starting and stopping the task will reduce efficiency.

(b)

Approach A:

Memory Usage:

Each worker will use X (n x n) and submatrix of Y (n x m) with a result of nxm.

For P wokers, the total memory is P x (nˆ2+2nm)

Amount of Communication:

P x (nˆ2+2mn) = nnp + 2nn

Approach B:

Memory Usage:

Each worker will use X (m x n) and submatrix of Y (n x m) with a result of nxm. For P wokers, the total memory is P x (2nm+mˆ2) Amount of Communication: (P x P)x (mˆ2+2mn) =nn +2pnn

Based on the result, we can see that B uses less memory than A, however, A takes less Commication cost than B.

In conclusion, Approch A is better for minimizing communication, Approach B is better for minimizing memory.