# CS553 Assignment 6

Lintao Lu / A20415263 / llu25@hawk.iit.edu

**1. Setting:**

First, install Java on chameleon cloud by using: *sudo apt-get install default-jdk* (maybe you should update apt-get). There is Java file called Sort.java on my directory. To run it, you should first enter "javac ./Sort.java" on your Linux terminal, and then enter "java sort with 3 parameters". The first parameter is how many element (or lines) in each chunk; second one is how many chunks you plan to split the whole file; last parameter is how many threads you plan to create.

For example: if we have 10GB file and 2 threads, I should enter "java Sort 1000000 100 2". But since I use Java to for this assignment, it is hard to predict how many memory spaces will be used. So for large data and small memory, number of element in each chunks should use a small number.'

**2. Design**

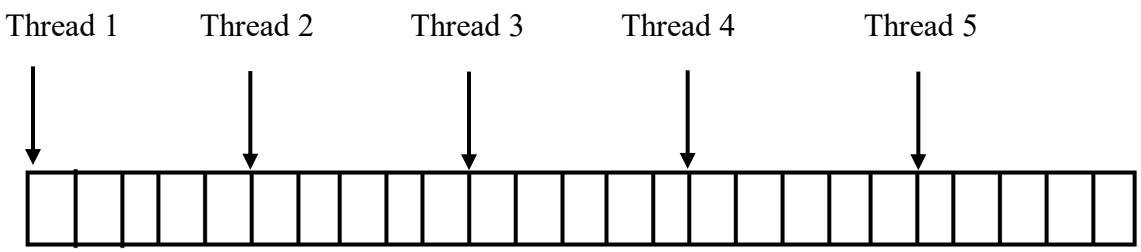
Since I use java to implement external sort, I got a very low performance. Firstly, Java cannot control memory directly; second, it's hard to know accurate memory usage; third JVM may cause more overhead. So my test result is expected slower than Linux sort.

Firstly, I split the whole data into several pieces in respect to how many threads we have. For example, if we have 10G data and 2 threads, what I did is splitting the whole data into 2 big chunks, so each thread take care of 5G. Number of elements is equal to how many data can be sorted in each time. In this case, each thread takes care of 5G but it cannot sort 5G for one time. So 1000000 means each thread can only sort 100M data each time. The reason why I just sort such small amount of data is because I cannot fully control memory. If I still have references in stack, Java garbage collector will not clean heap related objects.

Then, after first stage, I get a lot of sorted data, but the whole file still not be sorted. What I did next is to use K way merge. First I build a heap structure called index minimum priority queue. This structure save element itself and where is form, and always return the minimum element so far from the heap. Each time I read one element from disk to memory and push it into the priority queue. If the priority queue is not empty, I just pop element from it, push next one (form the same sorted data chunk as previous one) in to the queue, and save the popped element in a buffer that will be write into disk in the future. Here is a brief picture for such processes:

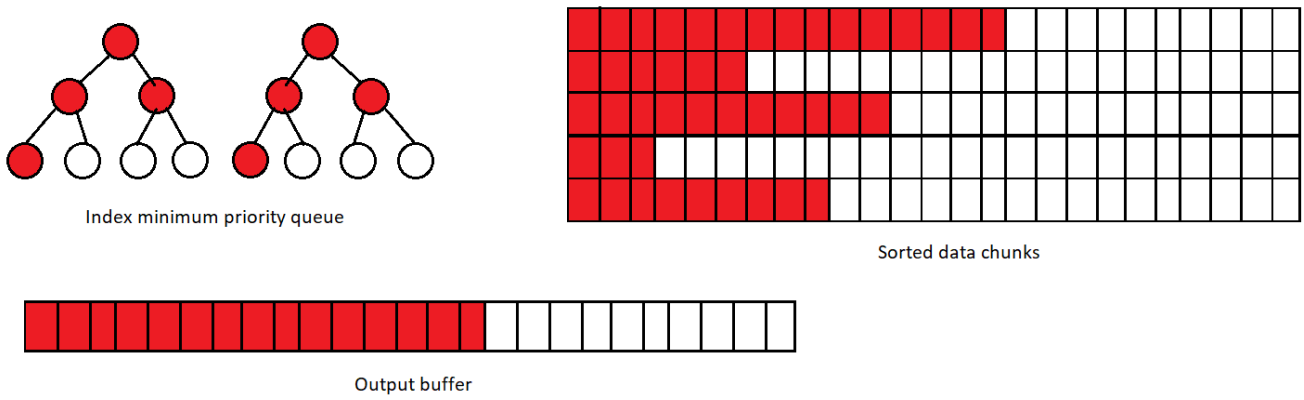
Stage 1: sort chunks

In this example, 5 threads to sort 25GB data. Each thread take care of 5G, but each time only read, sort and write 1G data.




Stage 2: K way merge

Left is index minimum priority queue structure, it saves elements and which data chunk it comes form. Right is the sorted data. Output buffer gets elements from index minimum priority queue. Once it is full, just write data to disk.



### 3. Test Result

Test 100 elements by using valsort on my local machine.

```
lintao@lintao-VirtualBox: ~
File  Edit  View  Search  Terminal  Help
        at Sort.main(Sort.java:145)
0 s
lintao@lintao-VirtualBox:~$ ./gensort -a 100 output
lintao@lintao-VirtualBox:~$ javac ./Sort.java
lintao@lintao-VirtualBox:~$ java Sort 50 2 2
Soring, please wait...
thread = 1; offset = 0; number of element = 50; rounds = 1
thread = 2; offset = 5000; number of element = 50; rounds = 1
2 chunks are sorted
0 s
lintao@lintao-VirtualBox:~$ ls
 Assignment6-New     Pictures                 Sort.java          llu25_keys.pem
 CS553               Public                   Videos             mHPL-1.0
 Desktop            'Sort$AsciiData.class'    ass-6              output
 Documents          'Sort$IndexMinPQ.class'   examples.desktop   pmbw-0.6.2
 Downloads          'Sort$QuickSort.class'    gensort            sortedFile
 ExternalSort.java  'Sort$ThreadSort.class'   homework6          valsort
 Music               Sort.class               linuxSort
lintao@lintao-VirtualBox:~$ ./valsort sortedFile
Records: 100
Checksum: 2f987241eb
Duplicate keys: 0
SUCCESS - all records are in order
lintao@lintao-VirtualBox:~$
```

1G, 2 cores machine, my code:

```
cc@hw6-llu25-medium: ~/ass6
File  Edit  View  Search  Terminal  Help
cc@hw6-llu25-medium:~/ass6$ ls
gensort  Sort$AsciiData.class  Sort$IndexMinPQ.class  Sort$QuickSort.class
output   Sort.class            Sort.java              Sort$ThreadSort.class
cc@hw6-llu25-medium:~/ass6$ java Sort 500000 20 2
Soring, please wait...
thread = 1; offset = 0; number of element = 500000; rounds = 10
thread = 2; offset = 500000000; number of element = 500000; rounds = 10
106 s
cc@hw6-llu25-medium:~/ass6$
```

1G, 2 cores machine, Linux sort:

cc@hw6-llu25-medium: ~/ass6

File Edit View Search Terminal Help

```
~~~%A NB_t  0000000000000000000000000003DE5EC  FFFF7777EEEEBBBB4444EEEEEEEE3333
9999DDDD999900005555
~~~-c-CQ(>  0000000000000000000000000832611  9999FFFF1111777733337777000011111
444444440000BBBB6666
~~~8Y}Fql*  0000000000000000000000000742A4C  BBBB1111CCCCEEEE8888000000007777
33333333DDDD22225555
~~~>Dd=QT]  0000000000000000000000000674C0F  9999DDDD000055556666CCCC22220000
FFFFEEEEDDDDFFFF0000
~~~]JA(}j$  0000000000000000000000000060BCBE  1111222233334444555599999AAAABBBB
9999FFFFDDDDBBBB3333
~~~]Zp.#/+  0000000000000000000000000003B9A5A  CCCC8888EEEEAAAAEEEE333333337777
0000FFFFCCCC66667777
~~~_jQepix  0000000000000000000000000011E5D4  1111999911115555BBBB111100002222
EEEE6666BBBB7777DDDD
~~~nt=ZH[N  0000000000000000000000000332A13  44441111BBBBBBBB33337777FFFF4444
5555555533330000CCCC
~~~s/Pq,-E  0000000000000000000000000006BE930  2222DDDDDDDD77771111EEEECCCC7777
BBBB4444888811111111
~~~zbA_ Tt  0000000000000000000000000007F9F4F  BBBBCCCC666655559999FFFF8888AAAA
11116666AAAABBBB0000

real    4m7.966s
user    0m16.644s
sys     0m50.260s
cc@hw6-llu25-medium:~/ass6$
```

10G, 2 cores machine, my code:

```
cc@hw6-llu25-medium: ~/ass6
File  Edit  View  Search  Terminal  Help
Soring, please wait...
thread = 1; offset = 0; number of element = 5000000; rounds = 10
thread = 2; offset = 5000000000; number of element = 5000000; rounds = 10
Exception in thread "Thread-1" java.lang.OutOfMemoryError: Java heap space
        at java.lang.StringCoding$StringEncoder.encode(StringCoding.java:300)
        at java.lang.StringCoding.encode(StringCoding.java:344)
        at java.lang.String.getBytes(String.java:918)
        at java.io.RandomAccessFile.open0(Native Method)
        at java.io.RandomAccessFile.open(RandomAccessFile.java:316)
        at java.io.RandomAccessFile.<init>(RandomAccessFile.java:243)
        at Sort.read(Sort.java:12)
        at Sort.sortChunks(Sort.java:55)
        at Sort$ThreadSort.run(Sort.java:172)
        at java.lang.Thread.run(Thread.java:748)
^Z
[1]+  Stopped                 java Sort 5000000 20 2
cc@hw6-llu25-medium:~/ass6$ ./gensort -a 100000000 output
cc@hw6-llu25-medium:~/ass6$ java Sort 1000000 100 2
Soring, please wait...
thread = 1; offset = 0; number of element = 1000000; rounds = 50
thread = 2; offset = 5000000000; number of element = 1000000; rounds = 50
100 chunks are sorted
2123 s
cc@hw6-llu25-medium:~/ass6$
```

10G, 2 cores machine, Linux sort:

```
cc@hw6-llu25-medium: ~/ass6
File  Edit  View  Search  Terminal  Help
cc@hw6-llu25-medium:~/ass6$ time sort output > linuxSort.log

real    4m55.719s
user    2m52.388s
sys     0m55.484s
cc@hw6-llu25-medium:~/ass6$
```

I cannot get enough resources form Chameleon cloud to test 40GB data. But I think in this case, I could enter “java sort 10000000 40 8”. But if it throws “out of memory” error, just decrease the size from 10000000 to 5000000 and don’t forget change 40 to 80. Keep doing so it should work.