Solution to Homework 6 (CS 553)

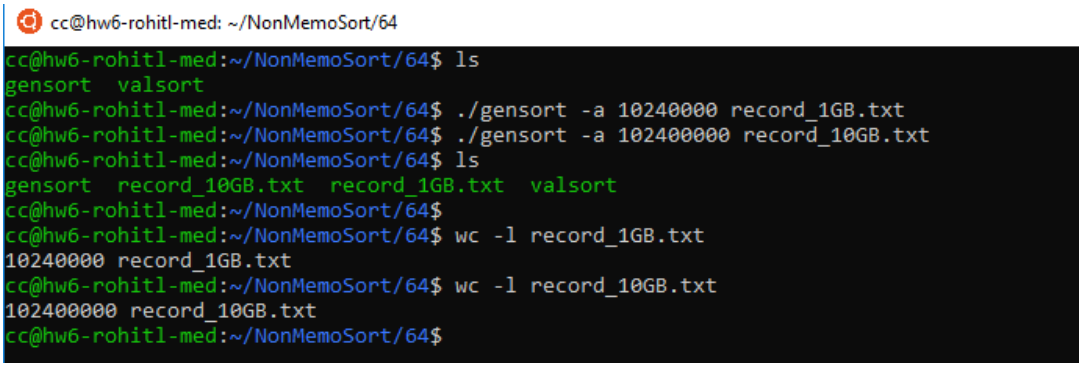
Rohit Lakde

1. Performance Table

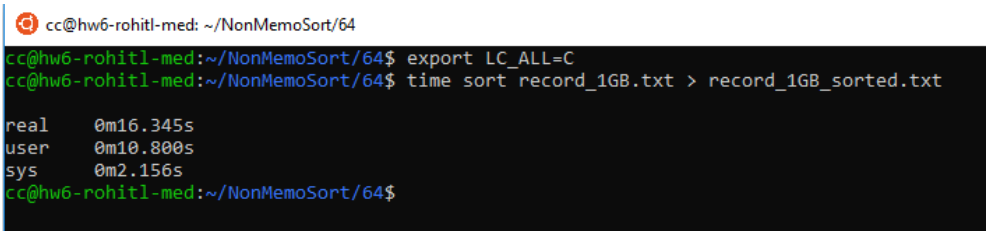
Experiment	Shared Memory (1GB)	Linux Sort (1GB)	Shared Memory (10GB)	Linux Sort (10GB)	Shared Memory (40GB)	Linux Sort (40GB)
Data Read (GB)	1GB	1GB	10GB	10GB	40GB	40GB
Data Write (GB)	1GB	1GB	10GB	10GB	40GB	40GB
Sort time (Sec)	25.33	16.34	613.86	425.57	1292.42	874.20
Overall, I/O Throughput (MB/sec)	78.95	122.39	32.58	46.99	61.89	91.51

2. 1GB Linux Sort validation:

Generating 1GB and 10 GB data using gensort algorithm



Time taken by linux sort



Valsort data validation

```
cc@hw6-rohitl-med: ~/NonMemoSort/64
cc@hw6-rohitl-med:~/NonMemoSort/64$ ./valsort record_1GB_sorted.txt
Records: 10240000
Checksum: 4e1d7bcdea349c
Duplicate keys: 0
SUCCESS - all records are in order
cc@hw6-rohitl-med:~/NonMemoSort/64$
```

Shared Memory Code

```
cc@hw6-rohitl-med: ~/NonMemoSort
cc@hw6-rohitl-med:~/NonMemoSort$ python NoMemSort1GB.py
Enter number of threads 2
thread t0 sorting done
thread t1 sorting done
Merging Done
Done! Time required is 25.3306591511Sec
Sorting output is stored in merge0.txt file
cc@hw6-rohitl-med:~/NonMemoSort$
```

Output validation using Valsort

```
cc@hw6-rohitl-med: ~/NonMemoSort/64
cc@hw6-rohitl-med:~/NonMemoSort/64$ ./valsort /home/cc/NonMemoSort/merge0.txt
Records: 10240000
Checksum: 4e1d7bcdea349c
Duplicate keys: 0
SUCCESS - all records are in order
cc@hw6-rohitl-med:~/NonMemoSort/64$
```

3. **10GB Linux Sort validation:**

Generating 10 GB data using gensort algorithm

```
cc@hw6-rohitl-med: ~/NonMemoSort/64
cc@hw6-rohitl-med:~/NonMemoSort/64$ ./gensort -a 102400000 record_10GB.txt
cc@hw6-rohitl-med:~/NonMemoSort/64$ wc -l record_10GB.txt
102400000 record_10GB.txt
cc@hw6-rohitl-med:~/NonMemoSort/64$
```

Time taken by linux sort

```
cc@hw6-rohitl-med: ~/NonMemoSort/64
cc@hw6-rohitl-med:~/NonMemoSort/64$ export LC_ALL=C
cc@hw6-rohitl-med:~/NonMemoSort/64$ time sort record_10GB.txt > record_10GB_sorted.txt

real    7m5.571s
user    2m22.060s
sys     0m43.916s
cc@hw6-rohitl-med:~/NonMemoSort/64$
```

Valsort data validation

```
cc@hw6-rohitl-med: ~/NonMemoSort/64
cc@hw6-rohitl-med:~/NonMemoSort/64$ ./valsort record_10GB_sorted.txt
Records: 102400000
Checksum: 30d3f9f9a56f3f3
Duplicate keys: 0
SUCCESS - all records are in order
cc@hw6-rohitl-med:~/NonMemoSort/64$
```

Shared memory Code:

```
cc@hw6-rohitl-med:~/NonMemoSort$ python NoMemSort_10GB.py
Enter number of threads 8
thread t0 sorting done
thread t1 sorting done
thread t2 sorting done
thread t3 sorting done
thread t4 sorting done
thread t5 sorting done
thread t6 sorting done
thread t7 sorting done
Done! Time required is 613.8658758234sec
Sorting output is sorted in merge7.txt file
cc@hw6-rohitl-med:~/NonMemoSort$
```

Valsort data validation

```
cc@hw6-rohitl-med:~/NonMemoSort/64$ ./valsort merge7.txt
Records: 102400000
Checksum: 30d3f9f9a56f3f3
Duplicate keys: 0
SUCCESS - all records are in order
```

4. **40 GB Data:**

Generating 40 GB random data

```
cc@hw6-rohitl-xlarge: ~/NonMemoSort/64
cc@hw6-rohitl-xlarge:~/NonMemoSort/64$ ./gensort -a 409600000 record_40GB.txt
cc@hw6-rohitl-xlarge:~/NonMemoSort/64$ ls
gensort  record_40GB.txt  valsort
cc@hw6-rohitl-xlarge:~/NonMemoSort/64$ wc -l record_40GB.txt
409600000 record_40GB.txt
cc@hw6-rohitl-xlarge:~/NonMemoSort/64$
```

Time required for linux sort

```
cc@hw6-rohitl-xlarge: ~/NonMemoSort/64
cc@hw6-rohitl-xlarge:~/NonMemoSort/64$ export LC_ALL=C
cc@hw6-rohitl-xlarge:~/NonMemoSort/64$ ls
gensort  record_40GB.txt  record_40GB_sorted.txt  valsort
cc@hw6-rohitl-xlarge:~/NonMemoSort/64$ time sort record_40GB.txt > record_40GB_sorted.txt

real    14m34.202s
user    12m17.504s
sys     1m58.648s
cc@hw6-rohitl-xlarge:~/NonMemoSort/64$
```

Valsort data validation

```
cc@hw6-rohitl-xlarge:~/NonMemoSort/64$ ./valsort record_40GB_sorted.txt
Records: 409600000
Checksum: c34f26ef1ef13fd
Duplicate keys: 0
SUCCESS - all records are in order
cc@hw6-rohitl-xlarge:~/NonMemoSort/64$
```

Shared memory code:

```
cc@hw6-rohitl-xlarge:~/NonMemoSort$ python NoMemSortt_40GB.py
Enter number of threads 8
thread t0 sorting done
thread t1 sorting done
thread t2 sorting done
thread t3 sorting done
thread t4 sorting done
thread t5 sorting done
thread t6 sorting done
thread t7 sorting done
Done! Time required is 1292.4278402187sec
Sorting output is stored in merge7.txt file
```

Valsort data validation

```
cc@hw6-rohitl-xlarge:~/NonMemoSort/64$ ./valsort merge7.txt
Records: 409600000
Checksum: c34f26ef1ef13fd
Duplicate keys: 0
SUCCESS - all records are in order
```

Theory:

1. Linux sort: Linux performs in memory sorting that's why time required to sort is very short. That's why if we compare any time required for given files, it is always lower than shared memory sorting.
2. For this assignment we have created random data by using generate sort for that we have installed one package containing gensort and valsort. Valsort is used to verify if sorted data is correct or not.
3. ASCI inputs are used everywhere in the implementation and below is the command to generate random files with given size "./gensort -a 10240000 random1gbdata.txt"
4. Command "sort random1gbdata.txt" is used to sort the data using default linux sort.
5. Command "./valsort random1gbdata.txt" is used to verify sorted data

6. In the implementation I have sliced list into eight equal parts and gave each part to one thread, each thread sorted data in each slice.
7. After that all threads are terminated except one, which does merging task.