

**CS 553 - Cloud Computing**  
**Programing Assignment 2 - Part A**

**Bhagyashree Bagwe (A20399761)**

### **Problem statement :**

“Perform external memory sort on single node with shared memory in a multi-threaded approach.”

### **Methodology:**

External memory sort is utilized for sorting larger than memory files. When data do not fit in main memory (RAM), external (or secondary) memory is used for storing chunks of the file temporarily.

In our experiment we are using two input data files one of size 2GB and another of size 20GB generated using gensort program, containing ASCII data.

I have developed two separate programs for sorting 2GB and 20GB data files. The flow of execution for both the programs goes as follows:

#### **2GB external sort:**

- Divide 2GB input file into 10 temporarily files of 200MB each and store them on disk temporarily
- Read one 200MB chunk at a time into the main memory
- Sort it and write it back to the disk
- Repeat above 2 steps for all temp input files
- When all the 10 files are sorted, merge all the sorted chunks in a multi threaded fashion by dividing chunks between threads

#### **20GB external sort:**

- Divide 20GB file into 10 chunks of 2GB each
- Pass each 2GB file as a input to above 2GB external sort program
- Merge all the sorted chunks in a multi threaded fashion by dividing chunks between threads for merging

### **Language of implementation :**

Java

### **Runtime Environment:**

Programs are executed on a Neutron cluster consisting of login and compute nodes connected via NFS.

The login nodes have capacity of 3GB RAM and compute nodes have capacity of 8GB RAM and 80GB of SSD storage which is essentially used for storing all the intermediary chunks.

While executing the source code is placed on the login node and is executed on the compute nodes using the slurm job scheduling.

```
bbagwe@neutron: ~/bhagya/cs553-pa2a
bbagwe@neutron:~/bhagya/cs553-pa2a$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list:   0-7
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             8
NUMA node(s):         1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                94
Model name:            Intel Core Processor (Skylake)
Stepping:              3
CPU MHz:               2099.998
BogoMIPS:              4199.99
Virtualization:        VT-x
Hypervisor vendor:     KVM
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              4096K
L3 cache:              16384K
NUMA node0 CPU(s):    0-7
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss syscall nx pdpe1gb
rdtscp lm constant_tsc rep_good nopl xtopology eagerfpu pni pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_t
mer aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invpcid_single rsb_ctxsw retpoline kaiser tpr_shadow vnmi flexpriority ept_v
pid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm rdseed adx smap xsaveopt arat
bbagwe@neutron:~/bhagya/cs553-pa2a$
```

### **Performance Evaluation:**

Experiment	Shared Memory (1VM 2 GB)	Linux Sort (1VM 2GB)	Shared Memory (1VM 20GB)	Linux Sort (1VM 20GB)
Compute Time (sec)	125	34.98	1141.25	439.024
Data Read (GB)	7.60	~4	116	~40
Data Write (GB)	8.80	~4	128	~40
I/O throughput (MB/sec)	130	~230	210	~180

Since access to disk drives is much slower than access to RAM, analysis of external-memory algorithms usually focuses on the number of disk accesses (I/O operations)

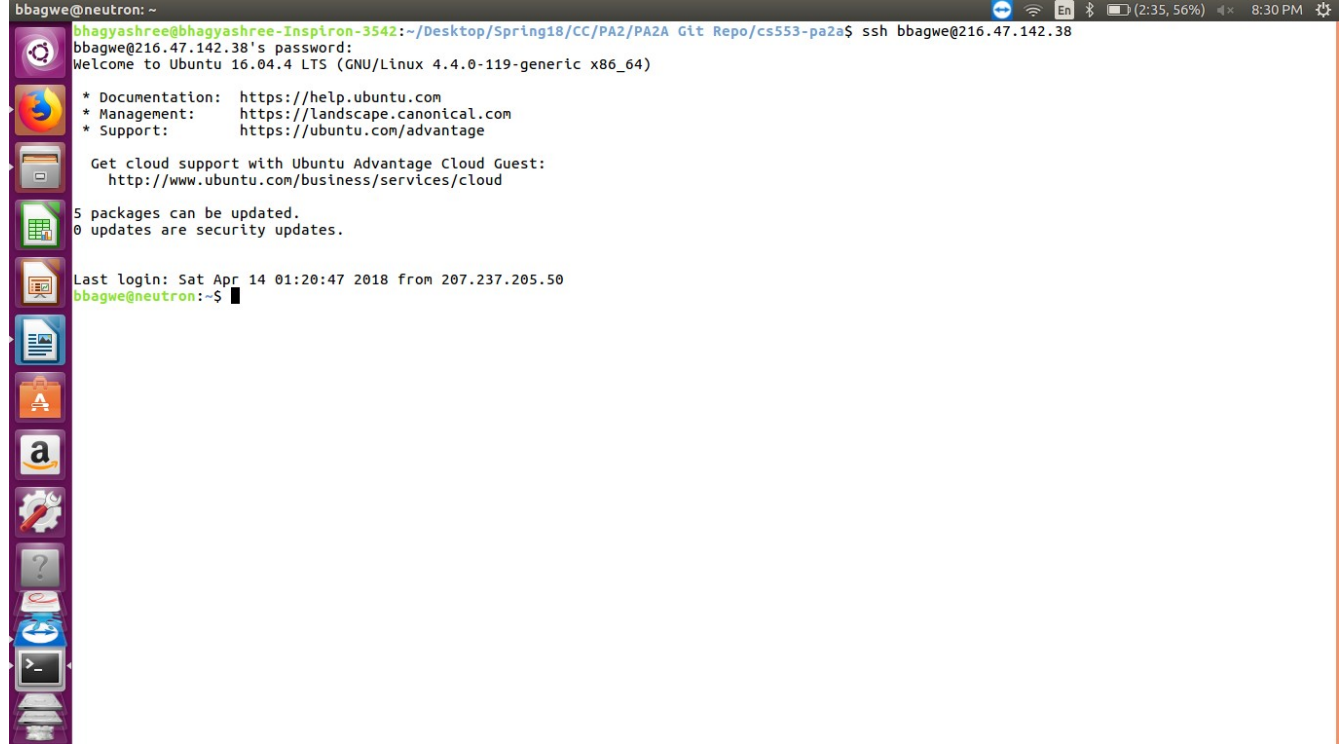
We are comparing comparing results of our experiments with UNIX sort command results.

For 2GB program, the throughput is less compared to linux sort. Since I have chosen a smaller chunk size for the available RAM, the program has to go back to disk more frequently and overall it ends up reading and writing more data from disk as compared to linux sort.

For 20GB program, since I have added an extra overhead of sending each chunk to 2GB program and merging them, even if it achieves more throughput as compared to linux sort, it compromises a lot on the response time.

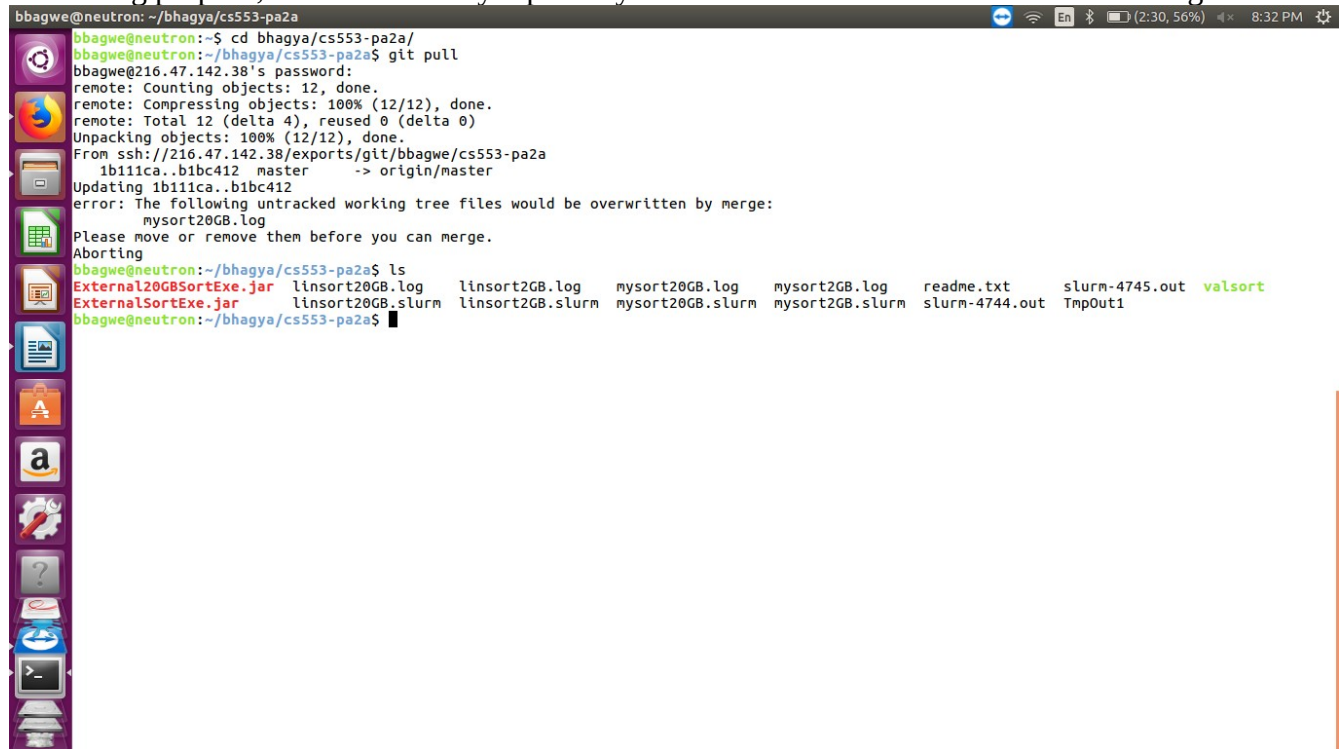
## Demo of execution

Login to cluster using `ssh bbagwe@216.47.142.38` command as shown in the figure:



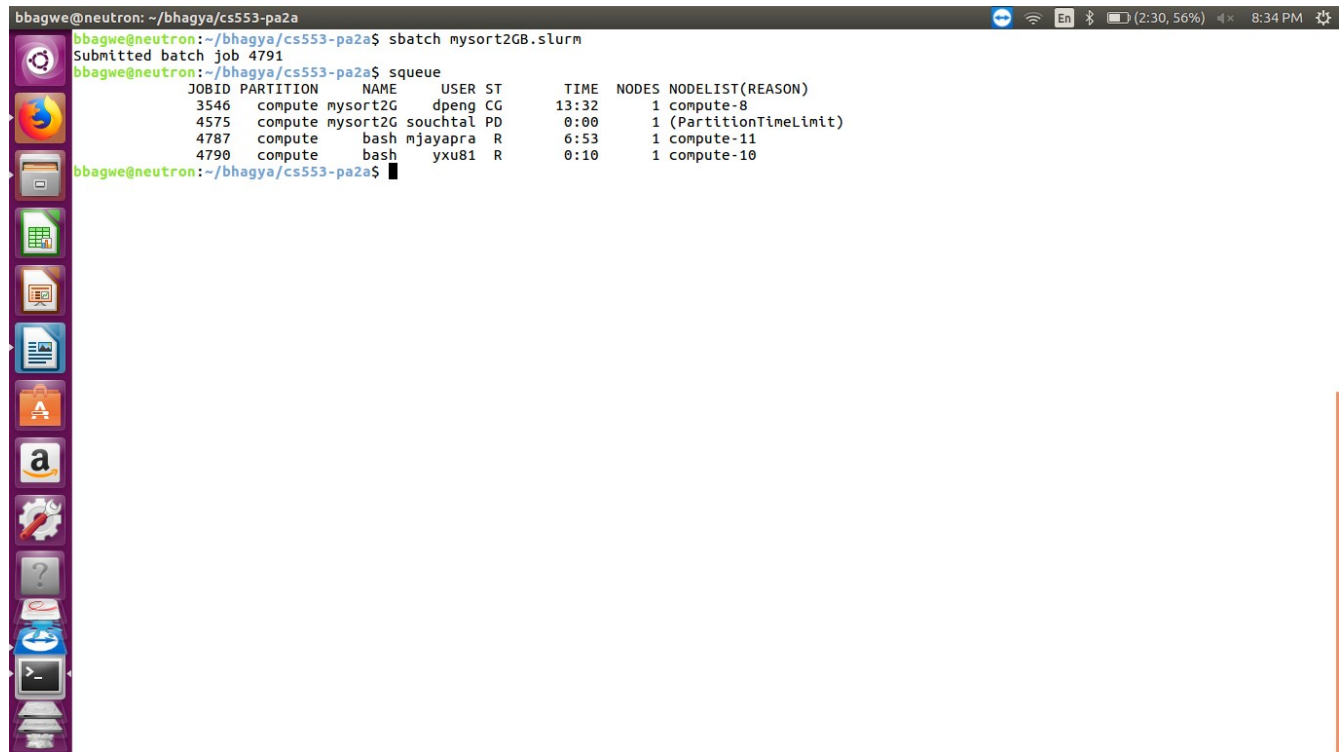
```
bbagwe@neutron: ~  
bhagyashree@bhagyashree-Inspiron-3542: ~/Desktop/Spring18/CC/PA2/PA2A Git Repo/cs553-pa2a$ ssh bbagwe@216.47.142.38  
bbagwe@216.47.142.38's password:  
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-119-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
Get cloud support with Ubuntu Advantage Cloud Guest:  
http://www.ubuntu.com/business/services/cloud  
  
5 packages can be updated.  
0 updates are security updates.  
  
Last login: Sat Apr 14 01:20:47 2018 from 207.237.205.50  
bbagwe@neutron:~$
```

For testing purpose, I have cloned my repository to a different location as shown in below figure:



```
bbagwe@neutron: ~/bhagya/cs553-pa2a  
bbagwe@neutron:~$ cd bhagya/cs553-pa2a/  
bbagwe@neutron:~/bhagya/cs553-pa2a$ git pull  
bbagwe@216.47.142.38's password:  
remote: Counting objects: 12, done.  
remote: Compressing objects: 100% (12/12), done.  
remote: Total 12 (delta 4), reused 0 (delta 0)  
Unpacking objects: 100% (12/12), done.  
From ssh://216.47.142.38/exports/git/bbagwe/cs553-pa2a  
 1b111ca..b1bc412  master    -> origin/master  
Updating 1b111ca..b1bc412  
error: The following untracked working tree files would be overwritten by merge:  
      mysort20GB.log  
Please move or remove them before you can merge.  
Aborting  
bbagwe@neutron:~/bhagya/cs553-pa2a$ ls  
External20GBSortExe.jar  linsort20GB.log  linsort20GB.log  mysort20GB.log  mysort20GB.log  readme.txt  slurm-4745.out  valsort  
ExternalSortExe.jar     linsort20GB.slurm  linsort20GB.slurm  mysort20GB.slurm  mysort20GB.slurm  slurm-4744.out  TmpOut1  
bbagwe@neutron:~/bhagya/cs553-pa2a$
```

To execute say 2GB program, submit a slurm job using command `sbatch mysort2GB.slurm`. We can check status of our job using `squeue` command after submission as shown in the figure:

A terminal window titled 'bbagwe@neutron: ~/bhagya/cs553-pa2a' shows the execution of 'sbatch mysort2GB.slurm', which submits batch job 4791. Subsequently, the 'squeue' command is run, displaying a table of active jobs. The table has columns: JOBID, PARTITION, NAME, USER, ST, TIME, NODES, and NODELIST(REASON). It lists four jobs: 3546 (mysort2G, dpeng, CG, 13:32, 1 node), 4575 (mysort2G, souchtal, PD, 0:00, 1 node), 4787 (bash, mjayapra, R, 6:53, 1 node), and 4790 (bash, yxu81, R, 0:10, 1 node). The terminal window includes a sidebar with application icons and a top status bar showing battery level and time.

```
bbagwe@neutron: ~/bhagya/cs553-pa2a$ sbatch mysort2GB.slurm
Submitted batch job 4791
bbagwe@neutron: ~/bhagya/cs553-pa2a$ squeue
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
3546	compute	mysort2G	dpeng	CG	13:32	1	compute-8
4575	compute	mysort2G	souchtal	PD	0:00	1	(PartitionTimeLimit)
4787	compute	bash	mjayapra	R	6:53	1	compute-11
4790	compute	bash	yxu81	R	0:10	1	compute-10

```
bbagwe@neutron: ~/bhagya/cs553-pa2a$
```

Once the job completes execution, a log file named `mysort2GB.log` will be generated in the same path as the executable. We use `cat` command to view its content.

#### References:

- <http://vkundeti.blogspot.com/2008/03/tech-algorithmic-details-of-unix-sort.html>
- <https://pdfs.semanticscholar.org/9d8d/14d9e7cc3f05e943934e8e473461e4b89477.pdf>