

CS553 Programming Assignment 2

David Ghiurco - A20325178

Alexandru Iulian Orhean - A20386581

[1. Introduction](#)

[2. Shared Memory Implementation](#)

[2.1 Virtual Cluster \(1-node i3.large\)](#)

[2.2 Virtual Cluster \(1-node i3.4xlarge\)](#)

[3. Apache Hadoop MapReduce Implementation](#)

[3.1 Virtual Cluster \(1-node i3.large\)](#)

[3.2 Virtual Cluster \(1-node i3.4xlarge\)](#)

[3.3 Virtual Cluster \(8-nodes i3.large\)](#)

[4. Apache Spark Implementation](#)

[4.1 Virtual Cluster \(1-node i3.large\)](#)

[4.2 Virtual Cluster \(1-node i3.4xlarge\)](#)

[4.3 Virtual Cluster \(8-nodes i3.large\)](#)

[5. Performance Evaluation](#)

[6. Difficulties encountered](#)

[7. Conclusions](#)

1. Introduction

The purpose of this assignment is to implement external sort (larger-than-RAM) applications across a wide range of dataset sizes, hardware platforms, and software frameworks. The experiments described in this report include examination of various external sort implementations across 1-node setups, multi-node setups, small datasets (128 GB), large datasets (1 TB), all performed on top of Amazon EC2.

The steps taken in setting up and performing this assignment involved a learning curve in working with Apache Hadoop, and Apache Spark. A lot of time was spent on debugging configurations of various daemons of these frameworks, and getting all cluster components to orchestrate harmoniously was no trivial task. Programming in these frameworks was rather short, as the software stack is designed to provide convenient tools for use cases such as ours.

Our clusters have the following software stack:

- Linux Ubuntu 16.04 LTS 64-bit (Standard free AMI on Amazon EC2)
- Java (OpenJDK) 1.8.0_151
- Apache Hadoop 2.7.4
- Apache Spark 2.2.0 (Scala 2.11) built with Hadoop 2.7.4
- Gradle 4.3.1

Gradle is used to compile the Hadoop and Spark applications. No manual Gradle environment setup is required, as everything is scripted.

Note: No MPI implementation was performed in this assignment

2. Shared Memory Implementation

The shared memory solution is implemented in the C programming language, with the help of GCC built-in atomic operations and the POSIX threads library. It doesn't have any other dependencies and the application can be easily built through common Linux tools (make and gcc). The shared memory implementation has higher performance, when compared with the alternatives (Hadoop and Spark), and that is due to its design and the elimination of any kind of redundancy or checkpointing. The algorithm uses the concepts behind bucket sort in order to split the initial input file into smaller files (preferably having the size smaller than the RAM limit per thread), grouping the input into known ranges of values. After the "split" phase, each smaller input file is sorted independently in memory, using quick sort (implemented by hand, due to optimization reasons).

The bucket sort algorithm has its own pitfalls, especially with respect to the “split” process, that yields results highly dependant on the uniformity of the random distribution of elements that need to be sorted. In order to partially solve this problem, the implementation keeps track of the resulting split files through the use of a tree, that expands its leaf nodes, in order generate files that would fit into memory. This step creates multiple levels of “split” phases and can be tuned through the **FACTOR** defined value in the source code. For best performance, experimentally it has been observed that a value of 4/8 is required of the **FACTOR** variable.

As a bonus, the application contains auto-logging information, that is displayed at the end, containing the number of read and write operations executed in terms of GB and the time required to execute the application.

2.1 Virtual Cluster (1-node i3.large)

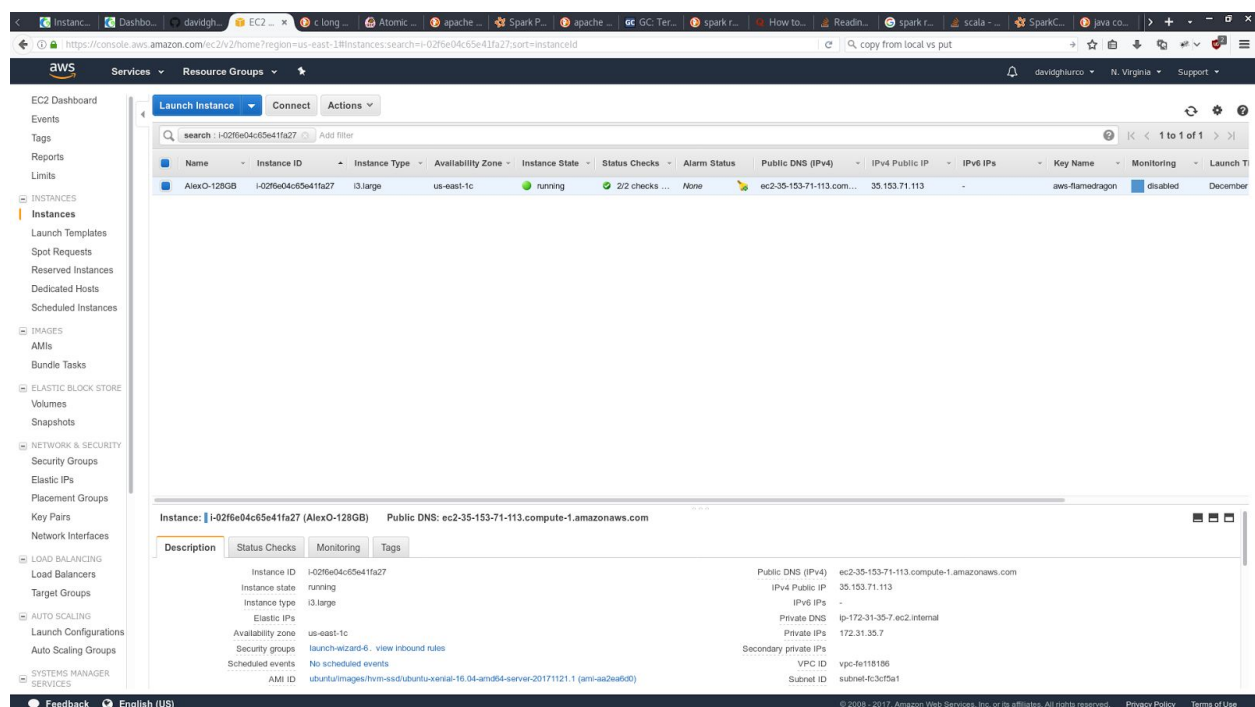


Figure 2.1.a AWS Instance running the 128GB sort experiment.

```
ubuntu@ip-172-31-35-7: /mnt/storage/ubuntu/cs553-2017-sort-benchmark/ptread-terasort 119x68

1 [|||||||||||||||||||||||||||||||||100.0%] Tasks: 35, 18 thr; 3 running
2 [|||||||||||||||||||||||||||||||||100.0%] Load average: 2.32 2.27 1.94
Mem[|||||||||||||||||5.15G/14.9G] Uptime: 02:19:43
Swp[|||||0K/0K]

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
8489 ubuntu 20 0 15.1G 5142M 1648 S 199.7 33.6 18:29.73 ./bin/ptreadsort.exe ../scripts/data/small.data 128000000
8496 ubuntu 20 0 15.1G 5142M 1648 R 99.7 33.6 5:46.21 ./bin/ptreadsort.exe ../scripts/data/small.data 128000000
8495 ubuntu 20 0 15.1G 5142M 1648 R 99.7 33.6 5:54.64 ./bin/ptreadsort.exe ../scripts/data/small.data 128000000
8493 ubuntu 20 0 24628 3448 2868 R 0.0 0.0 0:01.84 htop
1416 ubuntu 20 0 95368 3412 2440 S 0.0 0.0 0:01.07 sshd: ubuntu@pts/0
8009 ubuntu 20 0 29132 3496 2152 S 0.0 0.0 0:02.48 tmux
1177 root 10 -10 5724 3512 2424 S 0.0 0.0 0:00.67 /sbin/iscsid
1271 root 20 0 19608 2108 1804 S 0.0 0.0 0:00.28 /usr/sbin/irqbalance --pid=/var/run/irqbalance.pid
1176 root 20 0 5224 160 40 S 0.0 0.0 0:00.14 /sbin/iscsid
1130 root 20 0 266M 5176 4500 S 0.0 0.0 0:00.09 /usr/lib/accountsservice/accounts-daemon
1084 root 20 0 266M 5176 4500 S 0.0 0.0 0:00.10 /usr/lib/accountsservice/accounts-daemon
1 root 20 0 37716 3572 1784 S 0.0 0.0 0:02.84 /sbin/init
394 root 20 0 35276 3508 3188 S 0.0 0.0 0:00.76 /lib/systemd/systemd-journald
F1Help F2Setup F3Search F4Filter F5Free F6SortBy F7Nice F8Nice F9Kill F10Quit

Processed: 124570412300 Bytes
Processed: 124670420900 Bytes
Processed: 124770421000 Bytes
Processed: 124870560700 Bytes
Processed: 124970570300 Bytes
Processed: 125070576900 Bytes
Processed: 125170708500 Bytes
Processed: 125270708700 Bytes
Processed: 125370714200 Bytes
Processed: 125470714300 Bytes
Processed: 125570722900 Bytes
Processed: 125670875600 Bytes
Processed: 125770880300 Bytes
Processed: 125870967600 Bytes
Processed: 125971245400 Bytes
Processed: 126071245900 Bytes
Processed: 126171274300 Bytes
Processed: 126271277700 Bytes
Processed: 126371356300 Bytes
Processed: 126471460300 Bytes
Processed: 126571464400 Bytes
Processed: 126671512400 Bytes
Processed: 126771670100 Bytes
Processed: 126871672500 Bytes
Processed: 126971678400 Bytes
Processed: 127071715500 Bytes
Processed: 127171715700 Bytes
Processed: 127271719500 Bytes
Processed: 127371834200 Bytes
Processed: 127471930200 Bytes
Processed: 127572106600 Bytes
Processed: 127672115000 Bytes
Done splitting!
Starting parallel sort
Finished file ../scripts/data/small.data-17
Finished file ../scripts/data/small.data-16
Finished file ../scripts/data/small.data-18
Finished file ../scripts/data/small.data-19
Finished file ../scripts/data/small.data-20
Finished file ../scripts/data/small.data-21
Finished file ../scripts/data/small.data-22
Finished file ../scripts/data/small.data-23
Finished file ../scripts/data/small.data-25
Finished file ../scripts/data/small.data-24

[0] 0:htop "ip-172-31-35-7" 11:20 05-Dec-17
```

Figure 2.1.b Shared memory application running midway.

```

ubuntu@ip-172-31-35-7: /mnt/storage/ubuntu/cs553-2017-sort-benchmark/pthread-terasort 119x68

1 [ 0.0%] Tasks: 34, 16 thr; 1 running
2 [ 0.0%] Load average: 0.45 1.62 1.97
Mem[|||||] 119M/14.9G Uptime: 02:51:46
Swp[ ] 0K/0K

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
8009 ubuntu 20 0 29132 3496 2152 S 0.0 0.0 0:02.88 tmux
8493 ubuntu 20 0 24628 3456 2864 R 0.0 0.0 0:05.78 htop
1271 root 20 0 19608 2108 1804 S 0.0 0.0 0:00.37 /usr/sbin/irqbalance --pid=/var/run/irqbalance.pid
1177 root 10 -10 5724 3512 2424 S 0.0 0.0 0:00.84 /sbin/iscsid
1416 ubuntu 20 0 95368 3204 2232 S 0.0 0.0 0:01.15 sshd: ubuntu@pts/0
1176 root 20 0 5224 160 40 S 0.0 0.0 0:00.18 /sbin/iscsid
1130 root 20 0 266M 4852 4176 S 0.0 0.0 0:00.11 /usr/lib/accountsservice/accounts-daemon
1084 root 20 0 266M 4852 4176 S 0.0 0.0 0:00.12 /usr/lib/accountsservice/accounts-daemon
1146 syslog 20 0 254M 2904 2464 S 0.0 0.0 0:00.03 /usr/sbin/rsyslogd -n
1182 root 20 0 81404 13644 4280 S 0.0 0.1 0:00.01 /usr/lib/snapd/snapd
1160 root 20 0 28548 2588 2288 S 0.0 0.0 0:00.04 /lib/systemd/systemd-logind
1 root 20 0 37716 3568 1780 S 0.0 0.0 0:02.84 /sbin/init
394 root 20 0 35276 3508 3188 S 0.0 0.0 0:00.76 /lib/systemd/systemd-journald
F1Help F2Setup F3Search F4Filter F5Free F6Sort8 F7Nice F8Nice F9Kill F10Quit

../scripts/data/small.data-24
../scripts/data/small.data-25
../scripts/data/small.data-26
../scripts/data/small.data-27
../scripts/data/small.data-28
../scripts/data/small.data-29
../scripts/data/small.data-30
../scripts/data/small.data-31
../scripts/data/small.data-32
../scripts/data/small.data-33
../scripts/data/small.data-34
../scripts/data/small.data-35
../scripts/data/small.data-36
../scripts/data/small.data-37
../scripts/data/small.data-38
../scripts/data/small.data-39
../scripts/data/small.data-40
../scripts/data/small.data-41
../scripts/data/small.data-42
../scripts/data/small.data-43
../scripts/data/small.data-44
../scripts/data/small.data-45
../scripts/data/small.data-46
../scripts/data/small.data-47
../scripts/data/small.data-48
../scripts/data/small.data-49
../scripts/data/small.data-50
../scripts/data/small.data-51
../scripts/data/small.data-52
../scripts/data/small.data-53
../scripts/data/small.data-54
../scripts/data/small.data-55
../scripts/data/small.data-56
../scripts/data/small.data-57
../scripts/data/small.data-58
../scripts/data/small.data-59
../scripts/data/small.data-60
../scripts/data/small.data-61
../scripts/data/small.data-62
../scripts/data/small.data-63
Done sorting
Compute Time: 3093 sec
Data Read: 383 GB
Data Write: 383 GB
ubuntu@ip-172-31-35-7: /mnt/storage/ubuntu/cs553-2017-sort-benchmark/pthread-terasort$
[0] 0:htop* "ip-172-31-35-7" 11:53 05-Dec-17

```

Figure 2.1.c Shared memory application finishing the sort experiment.

```
ubuntu@ip-172-31-35-7: /mnt/storage/ubuntu/cs553-2017-sort-benchmark/pthread-terasort 119x68

1 [ 0.0%] Tasks: 34, 16 thr; 1 running
2 [ 0.0%] Load average: 0.86 1.29 1.66
Mem[||||||||||||||||||||||||||||||||||||| 118M/14.9G] Uptime: 02:58:44
Swp[||||| 0K/0K]

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
8493 ubuntu 20 0 24628 3456 2864 R 0.0 0.0 0:06.54 htop
8009 ubuntu 20 0 29132 3496 2152 S 0.0 0.0 0:03.02 tmux
1176 root 20 0 5224 160 40 S 0.0 0.0 0:00.19 /sbin/iscsid
1177 root 10 -10 5724 3512 2424 S 0.0 0.0 0:00.87 /sbin/iscsid
1271 root 20 0 19608 2108 1804 S 0.0 0.0 0:00.38 /usr/sbin/irqbalance --pid=/var/run/irqbalance.pid
1130 root 20 0 266M 4852 4176 S 0.0 0.0 0:00.12 /usr/lib/accounts-service/accounts-daemon

F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice +F9Kill F10Quit

Checksum: cd9ba3cbf3600f
Duplicate keys: 0
SUCCESS - all records are in order
Records: 26946500
Checksum: cd9538de51e74e
Duplicate keys: 0
SUCCESS - all records are in order
Records: 26938149
Checksum: cd83266de2296f
Duplicate keys: 0
SUCCESS - all records are in order
Records: 26945384
Checksum: cd95c4e6876f42
Duplicate keys: 0
SUCCESS - all records are in order
Records: 26944959
Checksum: cd99900fbffc18
Duplicate keys: 0
SUCCESS - all records are in order
Records: 26941130
Checksum: cd946b93a07fb3
Duplicate keys: 0
SUCCESS - all records are in order
Records: 26944714
Checksum: cd954029e877e8
Duplicate keys: 0
SUCCESS - all records are in order
Records: 26942878
Checksum: cd9a011343f3f6
Duplicate keys: 0
SUCCESS - all records are in order
Records: 26945968
Checksum: cd99a02297e2b5
Duplicate keys: 0
SUCCESS - all records are in order
Records: 26946331
Checksum: cda12aa2f78365
Duplicate keys: 0
SUCCESS - all records are in order
Records: 26950853
Checksum: cd9a160e0fe4b2
Duplicate keys: 0
SUCCESS - all records are in order
Records: 13473056
Checksum: 66c7b18b9f5ea2
Duplicate keys: 0
SUCCESS - all records are in order
Records: 1280000000
Checksum: 2625aa9ecff4f117
Duplicate keys: 0
SUCCESS - all records are in order
ubuntu@ip-172-31-35-7:/mnt/storage/ubuntu/cs553-2017-sort-benchmark/scripts$
[0] 0: bash* "ip-172-31-35-7" 11:59 05-Dec-17
```

Figure 2.1.d Validation of the shared memory sort results.

2.2 Virtual Cluster (1-node i3.4xlarge)

```
ubuntu@ip-172-31-44-240: /mnt/md0/cs553-2017-sort-benchmark/ptread-terasort 119x68

1  [|||||] 13.9% 5  [|||||] 19.5% 9  [|||||] 49.2% 13  [|||||] 0.7%
2  [|||||] 23.1% 6  [|||||] 17.1% 10 [|||||] 19.4% 14  [|||||] 0.0%
3  [|||||] 7.3% 7  [|||||] 50.0% 11 [|||||] 4.7% 15  [|||||] 71.0%
4  [|||||] 11.5% 8  [|||||] 3.6% 12 [|||||] 13.8% 16  [|||||] 48.2%
Mem [|||||] 3.38G/120G Tasks: 42, 556 thr; 8 running
Swp [|||||] 0K/0K Load average: 0.31 0.97 2.43
Uptime: 12:24:48

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
37546 ubuntu 20 0 582M 4056 1452 S 1011 0.0 0:45.71 ./bin/pterasort.exe ../scripts/data/big.data 1000000000
37552 ubuntu 20 0 582M 4056 1452 R 134. 0.0 0:05.78 ./bin/pterasort.exe ../scripts/data/big.data 1000000000
37549 ubuntu 20 0 582M 4056 1452 R 133. 0.0 0:06.01 ./bin/pterasort.exe ../scripts/data/big.data 1000000000
37551 ubuntu 20 0 582M 4056 1452 R 128. 0.0 0:05.48 ./bin/pterasort.exe ../scripts/data/big.data 1000000000
37550 ubuntu 20 0 582M 4056 1452 R 127. 0.0 0:05.82 ./bin/pterasort.exe ../scripts/data/big.data 1000000000
37547 ubuntu 20 0 582M 4056 1452 R 126. 0.0 0:05.81 ./bin/pterasort.exe ../scripts/data/big.data 1000000000
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice + F9Kill F10Quit

Processed: 2100351400 Bytes
Processed: 2200369900 Bytes
Processed: 2300389700 Bytes
Processed: 2400417800 Bytes
Processed: 2500751400 Bytes
Processed: 2600771800 Bytes
Processed: 2700780900 Bytes
Processed: 2800796400 Bytes
Processed: 2900815300 Bytes
Processed: 3000833200 Bytes
Processed: 3100851100 Bytes
Processed: 3200913400 Bytes
Processed: 3301205900 Bytes
Processed: 3401218200 Bytes
Processed: 3401219700 Bytes
Processed: 3501230400 Bytes
Processed: 3601244500 Bytes
Processed: 3701251700 Bytes
Processed: 3801266600 Bytes
Processed: 3901281100 Bytes
Processed: 4001294100 Bytes
Processed: 4101314400 Bytes
Processed: 4201336700 Bytes
Processed: 4301496600 Bytes
Processed: 4401518100 Bytes
Processed: 4501533700 Bytes
Processed: 4601583500 Bytes
Processed: 4701595200 Bytes
Processed: 4801609200 Bytes
Processed: 4901620600 Bytes
Processed: 5001634100 Bytes
Processed: 5101658300 Bytes
Processed: 5201666600 Bytes
Processed: 5301684700 Bytes
Processed: 5401699400 Bytes
Processed: 5501721600 Bytes
Processed: 5601739200 Bytes
Processed: 5701755100 Bytes
Processed: 5801770200 Bytes
Processed: 5901788600 Bytes
Processed: 6001803700 Bytes
Processed: 6101908400 Bytes
Processed: 6201930300 Bytes
Processed: 6301943300 Bytes
Processed: 6401956800 Bytes
Processed: 6501973100 Bytes
Processed: 6601986700 Bytes
Processed: 6701999800 Bytes

[2] 0:./bin/pterasort.exe* "ip-172-31-44-240" 17:22 05-Dec-17
```

Figure 2.2.a Starting the shared memory application for 1TB.

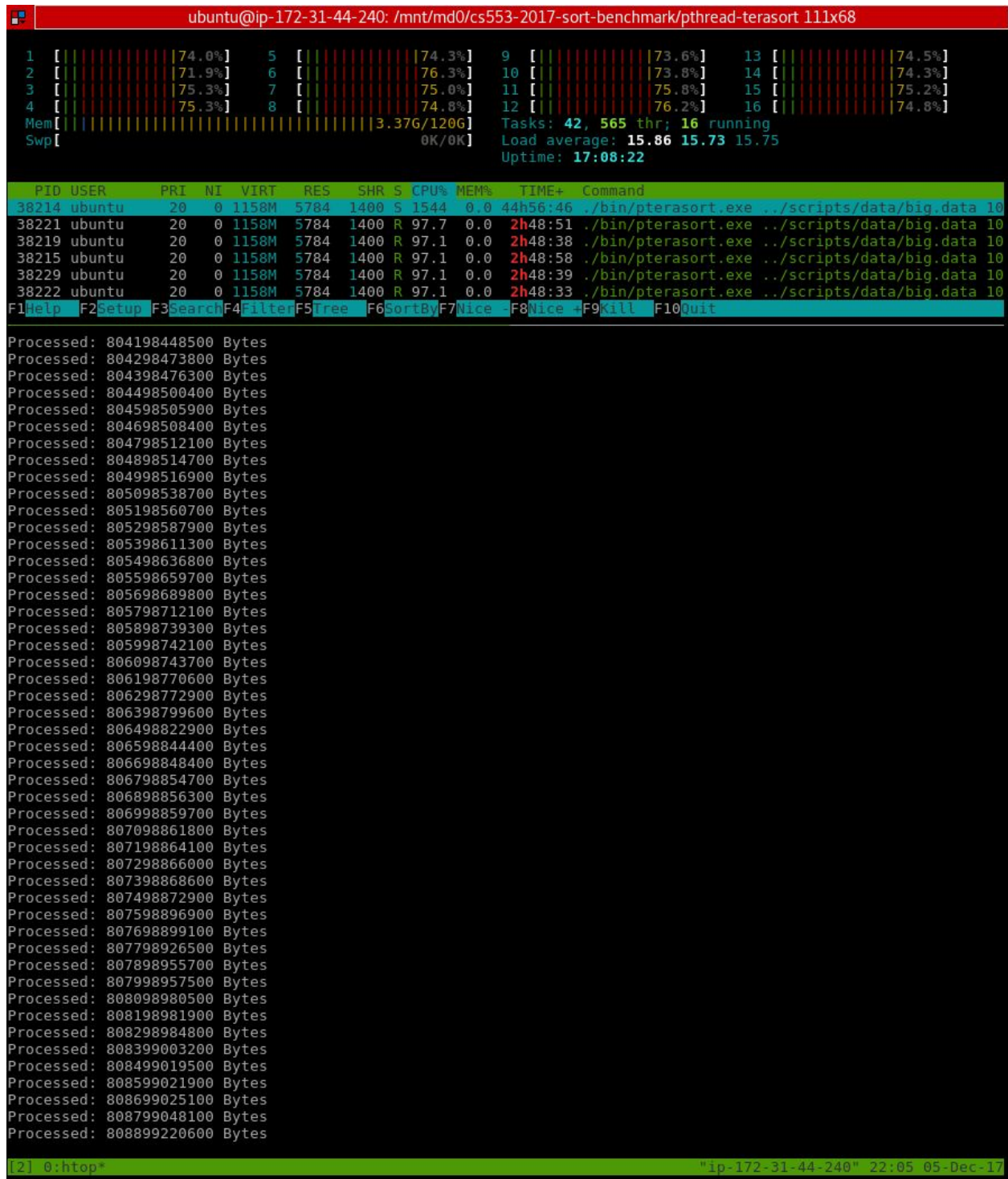


Figure 2.2.b Shared memory application running midway for 1TB.


```
ubuntu@ip-172-31-44-240: /mnt/md0/cs553-2017-sort-benchmark/pthread-terasort 119x68

1 [          0.0%] 5 [          0.0%] 9 [          0.0%] 13 [          0.0%]
2 [          0.0%] 6 [          0.0%] 10 [          0.0%] 14 [          0.0%]
3 [          0.0%] 7 [          0.7%] 11 [          0.0%] 15 [          0.0%]
4 [          0.0%] 8 [          0.0%] 12 [          0.0%] 16 [          0.0%]
Mem[||||]          3.37G/120G Tasks: 41, 550 thr; 1 running
Swp[          0K/0K] Load average: 0.03 0.14 3.35
Uptime: 18:26:57

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
 38560 ubuntu    20   0 26484  4980  2400  R   1.3   0.0   1:43.39 htop
105487 ubuntu    20   0 2828M  355M  4304  S   1.3   0.3  12:56.95 /usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java -Dproc_d
105943 ubuntu    20   0 2954M  519M  4212  S   0.7   0.4   2:57.51 /usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java -Dproc_r
106105 ubuntu    20   0 2920M  1119M  2532  S   0.7   0.9  30:51.59 /usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java -Dproc_n
105315 ubuntu    20   0 2795M  539M  4496  S   0.7   0.4   1:13.60 /usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java -Dproc_n
106301 ubuntu    20   0 2954M  519M  4212  S   0.7   0.4   0:00.62 /usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java -Dproc_r
F1Help F2Setup F3Search F4Filter F5Free F6SortBy F7Nice F8Nice F9Kill F10Quit

Processed file ../scripts/data/big.data-418
Processed file ../scripts/data/big.data-435
Processed file ../scripts/data/big.data-443
Processed file ../scripts/data/big.data-439
Processed file ../scripts/data/big.data-422
Processed file ../scripts/data/big.data-437
Processed file ../scripts/data/big.data-426
Processed file ../scripts/data/big.data-434
Processed file ../scripts/data/big.data-423
Processed file ../scripts/data/big.data-430
Processed file ../scripts/data/big.data-457
Processed file ../scripts/data/big.data-461
Processed file ../scripts/data/big.data-466
Processed file ../scripts/data/big.data-470
Processed file ../scripts/data/big.data-441
Processed file ../scripts/data/big.data-462
Processed file ../scripts/data/big.data-445
Processed file ../scripts/data/big.data-454
Processed file ../scripts/data/big.data-480
Processed file ../scripts/data/big.data-469
Processed file ../scripts/data/big.data-449
Processed file ../scripts/data/big.data-438
Processed file ../scripts/data/big.data-450
Processed file ../scripts/data/big.data-446
Processed file ../scripts/data/big.data-465
Processed file ../scripts/data/big.data-442
Processed file ../scripts/data/big.data-484
Processed file ../scripts/data/big.data-453
Processed file ../scripts/data/big.data-473
Processed file ../scripts/data/big.data-488
Processed file ../scripts/data/big.data-477
Processed file ../scripts/data/big.data-492
Processed file ../scripts/data/big.data-478
Processed file ../scripts/data/big.data-494
Processed file ../scripts/data/big.data-474
Processed file ../scripts/data/big.data-482
Processed file ../scripts/data/big.data-496
Processed file ../scripts/data/big.data-498
Processed file ../scripts/data/big.data-486
Processed file ../scripts/data/big.data-481
Processed file ../scripts/data/big.data-493
Processed file ../scripts/data/big.data-489
Processed file ../scripts/data/big.data-490
Processed file ../scripts/data/big.data-485
Sorting finished!
Compute Time: 16920 sec
Data Read: 2139 GB
Data Write: 2139 GB
ubuntu@ip-172-31-44-240: /mnt/md0/cs553-2017-sort-benchmark/pthread-terasort$
[2] 0:bash*
```

Figure 2.2.c Shared memory application finishing the sort experiment for 1TB.

3. Apache Hadoop MapReduce Implementation

3.1 Virtual Cluster (1-node i3.large)

	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
	davidghiurco	i-0abef69b12bb4477f	i3.large	us-east-1c	running	2/2 checks ...	None	ec2-54-84-86-198.com...	54.84.86.198

Here is the teragen job that created the 128 GB dataset:

```
17/12/05 05:09:06 INFO mapreduce.Job: Job job_1512448939295_0001 completed successfully
17/12/05 05:09:07 INFO mapreduce.Job: Counters: 31
  File System Counters
    FILE: Number of bytes read=0
    FILE: Number of bytes written=241526
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=170
    HDFS: Number of bytes written=128000000000
    HDFS: Number of read operations=8
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=4
  Job Counters
    Launched map tasks=2
    Other local map tasks=2
    Total time spent by all maps in occupied slots (ms)=2727749
    Total time spent by all reduces in occupied slots (ms)=0
    Total time spent by all map tasks (ms)=2727749
    Total vcore-milliseconds taken by all map tasks=2727749
    Total megabyte-milliseconds taken by all map tasks=2793214976
  Map-Reduce Framework
    Map input records=1280000000
    Map output records=1280000000
    Input split bytes=170
    Spilled Records=0
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ms)=20895
    CPU time spent (ms)=2178840
    Physical memory (bytes) snapshot=559861760
    Virtual memory (bytes) snapshot=3948421120
    Total committed heap usage (bytes)=382205952
  org.apache.hadoop.examples.terasort.TeraGen$Counters
    CHECKSUM=2748814493656638320
  File Input Format Counters
    Bytes Read=0
  File Output Format Counters
    Bytes Written=128000000000
```

The Hadoop application was run with 2 Reducer tasks, one for each VCPU:

```
ubuntu@ip-172-31-36-148:/mnt/storage/cs553-2017-sort-benchmark$ hadoop jar hadoop-terasort/build/libs/htera.jar $
Dmapred.reduce.tasks=2 /input /output
17/12/05 06:26:15 INFO client.RMPProxy: Connecting to ResourceManager at ip-172-31-36-148/172.31.36.148:8032
17/12/05 06:26:15 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement$
the Tool interface and execute your application with ToolRunner to remedy this.
17/12/05 06:26:16 INFO input.FileInputFormat: Total input paths to process : 2
Spent 238ms computing base-splits.
Spent 6ms computing TeraScheduler splits.
17/12/05 06:26:16 INFO mapreduce.JobSubmitter: number of splits:954
17/12/05 06:26:17 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1512455154927_0001
17/12/05 06:26:17 INFO impl.YarnClientImpl: Submitted application application_1512455154927_0001
17/12/05 06:26:17 INFO mapreduce.Job: The url to track the job: http://ip-172-31-36-148:8088/proxy/application_1$
12455154927_0001/
17/12/05 06:26:17 INFO mapreduce.Job: Running job: job_1512455154927_0001
17/12/05 06:26:25 INFO mapreduce.Job: Job job_1512455154927_0001 running in uber mode : false
17/12/05 06:26:25 INFO mapreduce.Job: map 0% reduce 0%
17/12/05 06:27:26 INFO mapreduce.Job: map 1% reduce 0%
17/12/05 06:28:49 INFO mapreduce.Job: map 2% reduce 0%
17/12/05 06:30:10 INFO mapreduce.Job: map 3% reduce 0%
```

Unfortunately, our Hadoop program never terminates on large datasets.

Although sorting was verified on small, test datasets, the job never terminates on the dataset magnitudes required for the assignment.

```
17/12/05 21:12:05 INFO mapreduce.Job: map 100% reduce 35%
17/12/05 21:12:20 INFO mapreduce.Job: map 100% reduce 36%
17/12/05 21:12:38 INFO mapreduce.Job: map 100% reduce 37%
17/12/05 21:13:05 INFO mapreduce.Job: map 100% reduce 38%
17/12/05 21:13:20 INFO mapreduce.Job: map 100% reduce 39%
17/12/05 21:13:41 INFO mapreduce.Job: map 100% reduce 40%
17/12/05 21:14:02 INFO mapreduce.Job: map 100% reduce 41%
17/12/05 21:14:20 INFO mapreduce.Job: map 100% reduce 42%
17/12/05 21:14:44 INFO mapreduce.Job: map 100% reduce 43%
17/12/05 21:15:02 INFO mapreduce.Job: map 100% reduce 44%
17/12/05 21:15:20 INFO mapreduce.Job: map 100% reduce 45%
```

The reduce phase gets stuck at either 33% or 45% (in this case). We suspect this has to do with the infrastructure stack that we are using. Hadoop is not optimized to run on virtual machines (like the simple Ubuntu AMI we are using on Amazon) and as such, Apache recommends either running Hadoop baremetal, or using specialized VM instances (those AMIs are both not free, and also explicitly forbidden by the assignment).

Additionally, the ResourceManager from Yarn indicates that the application is no longer running:

```
[detached (from session 0)]
ubuntu@ip-172-31-44-85:~$ yarn application -list
17/12/05 23:55:12 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-44-85/172.31.44.85:8032
Total number of applications (application-types: [] and states: [SUBMITTED, ACCEPTED, RUNNING]):1
  Application-Id      Application-Name      Application-Type      User      Queue
  State      Final-State      Progress      Tracking-URL
application_1512500242697_0002      HTerasort      MAPREDUCE      ubuntu      default
ACCEPTED      UNDEFINED      0%      N/A
ubuntu@ip-172-31-44-85:~$
```

This happens on the other Hadoop deployments/datasets as well. We were unable to debug it.

3.2 Virtual Cluster (1-node i3.4xlarge)

<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP
<input type="checkbox"/>	DavidG-1TB	i-02f852954daf20d60	i3.4xlarge	us-east-1c	running	2/2 checks ...	None	ec2-34-207-103-92.co...	34.207.103.92

Here is the teragen job that created the 1 TB dataset

```
17/12/05 09:05:07 INFO mapreduce.Job: Job job_1512458895359_0001 completed successfully
17/12/05 09:05:07 INFO mapreduce.Job: Counters: 31
  File System Counters
    FILE: Number of bytes read=0
    FILE: Number of bytes written=241818
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=173
    HDFS: Number of bytes written=1000000000000
    HDFS: Number of read operations=8
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=4
  Job Counters
    Launched map tasks=2
    Other local map tasks=2
    Total time spent by all maps in occupied slots (ms)=9182381
    Total time spent by all reduces in occupied slots (ms)=0
    Total time spent by all map tasks (ms)=9182381
    Total vcore-milliseconds taken by all map tasks=9182381
    Total megabyte-milliseconds taken by all map tasks=9402758144
  Map-Reduce Framework
    Map input records=100000000000
    Map output records=100000000000
    Input split bytes=173
    Spilled Records=0
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ms)=96796
    CPU time spent (ms)=11138570
    Physical memory (bytes) snapshot=688504832
    Virtual memory (bytes) snapshot=4064907264
    Total committed heap usage (bytes)=343408640
  org.apache.hadoop.examples.terasort.TeraGen$Counters
    CHECKSUM=3028416809717741100
  File Input Format Counters
    Bytes Read=0
  File Output Format Counters
    Bytes Written=1000000000000
```

The Hadoop application was run with 16 Reducer tasks, one for each VCPU:

```
ubuntu@ip-172-31-44-240:/mnt/md0/cs553-2017-sort-benchmark$ hadoop jar hadoop-terasort/build/libs/htera.jar -Dmap
red.reduce.tasks=16 /input /output
17/12/05 09:08:30 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-44-240/172.31.44.240:8032
17/12/05 09:08:30 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement
the Tool interface and execute your application with ToolRunner to remedy this.
17/12/05 09:08:30 INFO input.FileInputFormat: Total input paths to process : 2
Spent 235ms computing base-splits.
Spent 22ms computing TeraScheduler splits.
17/12/05 09:08:30 INFO mapreduce.JobSubmitter: number of splits:7452
17/12/05 09:08:30 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1512458895359_0002
17/12/05 09:08:31 INFO impl.YarnClientImpl: Submitted application application_1512458895359_0002
17/12/05 09:08:31 INFO mapreduce.Job: The url to track the job: http://ip-172-31-44-240:8088/proxy/application_15
12458895359_0002/
17/12/05 09:08:31 INFO mapreduce.Job: Running job: job_1512458895359_0002
17/12/05 09:08:36 INFO mapreduce.Job: Job job_1512458895359_0002 running in uber mode : false
17/12/05 09:08:36 INFO mapreduce.Job: map 0% reduce 0%
```

This job run also got stuck at reduce phase 33% and never terminated. We tried debugging it for a couple of days but have come up with nothing. This is the case with all our hadoop jobs in this assignment. Thus we are unable to get final performances results, although we have firm confirmation that the sort application WORKS on small datasets.

3.3 Virtual Cluster (8-nodes i3.large)

We have almost automated the entire multi-node deployment process. We spent a very long time writing the scripts, but we had a lot of trouble getting the system completely loaded. Multiple YARN NodeManager daemons constantly refused to load and we could not get a working cluster even though we got a quota increase for 8 i3.large nodes

4. Apache Spark Implementation

4.1 Virtual Cluster (1-node i3.large)

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs
DavidG-128GBSpark	i-0f35f0cb61ca0b7a4	i3.large	us-east-1c	running	2/2 checks ...	None	ec2-35-153-133-156.co...	35.153.133.156	-

Spark was run this instance with 2 partitions (for 2 VCPUs)

```
17/12/05 21:39:24 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:25 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:26 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:27 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:28 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:29 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:30 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:31 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:32 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:33 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:34 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:35 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:36 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:37 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:38 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:39 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:40 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:41 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:42 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:43 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:44 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:45 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:46 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:47 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:48 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:49 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:50 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:51 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:52 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:53 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:54 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:55 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:56 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:57 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:58 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:39:59 INFO yarn.Client: Application report for application_1512480559181_0004 (state: RUNNING)
17/12/05 21:40:00 INFO yarn.Client: Application report for application_1512480559181_0004 (state: FINISHED)
17/12/05 21:40:00 INFO yarn.Client:
client token: N/A
diagnostics: N/A
ApplicationMaster host: 172.31.43.252
ApplicationMaster RPC port: 0
queue: default
start time: 1512504394705
final status: SUCCEEDED
tracking URL: http://ip-172-31-43-252:8088/proxy/application_1512480559181_0004/
user: ubuntu
```

Here is the output of the spark application run on 128 GB

```
ubuntu@ip-172-31-43-252:/mnt/storage/cs553-2017-sort-benchmark$ hadoop fs -ls /output2
Found 2 items
-rw-r--r-- 1 ubuntu supergroup 30665318899 2017-12-05 21:40 /output2/part-00000
-rw-r--r-- 1 ubuntu supergroup 27283596802 2017-12-05 21:36 /output2/part-00001
```

Again, we have issues with the output format of the flushed sorted output in Spark. The output shown above is actually non-sensical binary when run on the 128 GB dataset.

However, we ran the same Spark application on Chameleon on a small data set and it sorted successfully:

```
cc@flamedragon:~/output$ rm *.sum
cc@flamedragon:~/output$ ./valsort -o 0.sum part-00000
Records: 62879
Checksum: 7b121ea3b17f
Duplicate keys: 0
SUCCESS - all records are in order
cc@flamedragon:~/output$ ./valsort -o 1.sum part-00001
Records: 65121
Checksum: 7ee759b68c09
Duplicate keys: 0
SUCCESS - all records are in order
cc@flamedragon:~/output$ cat 0.sum 1.sum > all.sum
cc@flamedragon:~/output$ ./valsort -s all.sum
Records: 128000
Checksum: f9f9785a3d88
Duplicate keys: 0
SUCCESS - all records are in order
cc@flamedragon:~/output$ S
```

Unfortunately, this means that we do not have conclusive performance data of our Spark application on large datasets.

4.2 Virtual Cluster (1-node i3.4xlarge)

4.3 Virtual Cluster (8-nodes i3.large)

We were unable to get a working multi-node deployment for Spark for the same reasons as above with the Hadoop clusters. We configured Spark to work with YARN, and YARN refused to cooperate on multi-node deployments.

5. Performance Evaluation

The shared memory implementation got some very interesting results overall. In the case of the 1TB dataset/configuration the application generated several segmentation faults caused by the inherent limit of open stream files for one program (approximately 2024) at the same time. In order to quickly fix this problem, files are opened and closed on demand, instead of being open throughout the split phase, thus contributing to a deficiency in performance. The efficiency of the application is pretty low, around at half, in contrast to the read and write throughput supported by these devices (over 600 MB/s sequentially). The I/O system but also the bus were stressed throughout the shared memory execution, and due to un-optimized and unorganized data movement, the application achieved low efficiency. From the scalability point of view, the weak speedup shows that the application scales well on multi-core systems.

Since we managed to run only single node experiments, in this regard the shared memory implementation outperformed Hadoop and Spark, being able to sort 32GB in less than 2 minutes, while the latter were able to do it in tens of minutes and almost ten minutes respectively. On one node the shared memory implementation scales even better than Hadoop and Spark, making use of the shared memory capabilities, without having to merge results or intermediary data, since it was already shared. ON 8 nodes, probably Hadoop and Spark would have the advantage since they support distributed communication and synchronization. Moreover we have observed that the more nodes Hadoop had the faster it ran applications. Experimentally, it took more time to generate 128GB of data on a single node than to generate 1TB of data on 5 nodes.

Experiment (instance/dataset)	Shared Memory TeraSort	Hadoop TeraSort	Spark TeraSort
Compute Time (sec) [1xi3.large 128GB]	3093		
Data Read (GB) [1xi3.large 128GB]	383		
Data Write (GB) [1xi3.large 128GB]	383		
I/O Throughput (MB/sec) [1xi3.large 128GB]	247.66		
Compute Time (sec) [1xi3.4xlarge 1TB]	16920		
Data Read (GB) [1xi3.4xlarge 1TB]	2139		
Data Write (GB) [1xi3.4xlarge 1TB]	2139		
I/O Throughput (MB/sec) [1xi3.4xlarge 1TB]	252.83		
Compute Time (sec) [8xi3.large 1TB]	N/A		

Data Read (GB) [8xi3.large 1TB]	N/A		
Data Write (GB) [8xi3.large 1TB]	N/A		
I/O Throughput (MB/sec) [8xi3.large 1TB]	N/A		
Speedup (weak scale)	1.02		
Efficiency (weak scale)	40-50%		

Table : Performance evaluation of TeraSort

Line Endings

```
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ diff head1 head2
1,10c1,10
~      =2G^9[- 0000000000000000000000000809EE 5555DDDD1111CCCC9999BBBB0000BBBCCCCFFFCFCC44443333
!<fJ,X\] 000000000000000000000000000000051799 22222222666600000333FFFFFFFFFF555500000000888EEEEEEEE
!"qVvYlHH 00000000000000000000000000000057C1A 2222EEEE000099990000222233388886666FFFF9999AAAA7777
[Y>:#%# 00000000000000000000000000000002B28D 555511118888FFFF88884444EEEE222244444446666BBBBAAAA
~ #kIo {&j 0000000000000000000000000000000A12AD 11155550DDDD1111222DDDDFFF77775555BBB11119999AAAA
~ $dTzn,no 000000000000000000000000000000098539 4444BBBB9999999992221111444422222233388884444EEEE
~ %E6`ElC% 0000000000000000000000000000000BC9D9 5555AAAA9999922220000DDDD1111FFFD0DD11118888AAAAEEEE
~ %,wO>oO 0000000000000000000000000000000354D0 44443333BBB888888822220000999944443335555FFFF1111
~ %-\\_mi* 00000000000000000000000000000000ACFD0 AAAAFFFF7777AAAAABBB88881111DDDDBBB4444AAAAEEEEAAAA
~ '.LQL kv 00000000000000000000000000000008B8AS AAAA333388889999EEEEAAAA6666222222AAAA333366662222
---
~      =2G^9[- 0000000000000000000000000809EE 5555DDDD1111CCCC9999BBBB0000BBBCCCCFFFCFCC44443333
!<fJ,X\] 000000000000000000000000000000051799 22222222666600000333FFFFFFFFFF555500000000888EEEEEEEE
!"qVvYlHH 00000000000000000000000000000057C1A 2222EEEE000099990000222233388886666FFFF9999AAAA7777
[Y>:#%# 00000000000000000000000000000002B28D 555511118888FFFF88884444EEEE222244444446666BBBBAAAA
~ #kIo {&j 0000000000000000000000000000000A12AD 11155550DDDD1111222DDDDFFF77775555BBB11119999AAAA
~ $dTzn,no 000000000000000000000000000000098539 4444BBBB9999999992221111444422222233388884444EEEE
~ %E6`ElC% 0000000000000000000000000000000BC9D9 5555AAAA9999922220000DDDD1111FFFD0DD11118888AAAAEEEE
~ %,wO>oO 0000000000000000000000000000000354D0 44443333BBB888888822220000999944443335555FFFF1111
~ %-\\_mi* 00000000000000000000000000000000ACFD0 AAAAFFFF7777AAAAABBB88881111DDDDBBB4444AAAAEEEEAAAA
~ '.LQL kv 00000000000000000000000000000008B8AS AAAA333388889999EEEEAAAA6666222222AAAA333366662222
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ ls
head1 head2 part-self part-teragen
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ vim part-self
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ vim part-teragen
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ vim part-self
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ vim part-teragen
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ ls
head1 head2 part-self part-teragen
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ ls
head1 head2 part-self part-teragen
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ wc -c < part-self
990000000
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ wc -c < part-teragen
100000000
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ wc -c < head1
990
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ wc -c < head2
1000
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ ls
head1 head2 part-self part-teragen
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ rm head*
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ ls
part-self part-teragen
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ head -n 1 part-self
=2G^9[- 0000000000000000000000000809EE 5555DDDD1111CCCC9999BBBB0000BBBCCCCFFFCFCC44443333
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ head -n 1 part-teragen
=2G^9[- 0000000000000000000000000809EE 5555DDDD1111CCCC9999BBBB0000BBBCCCCFFFCFCC44443333
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ head -n 1 part-self >> head1
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ head -n 1 part-teragen >> head2
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ ls
head1 head2 part-self part-teragen
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ diff head1 head2
1c1
~      =2G^9[- 0000000000000000000000000809EE 5555DDDD1111CCCC9999BBBB0000BBBCCCCFFFCFCC44443333
---
~      =2G^9[- 0000000000000000000000000809EE 5555DDDD1111CCCC9999BBBB0000BBBCCCCFFFCFCC44443333
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ cmp head1 head2
head1 head2 differ: byte 99, line 1
```



```
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ file head1
head1: ASCII text
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ file head2
head2: ASCII text, with CRLF line terminators
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ '
```

Hadoop TeraGen documentation indicates that this is the format for each row in the generated data:

```
<10 bytes key><10 bytes rowid><78 bytes filler>\r\n
```

Experimentation determined that this is the case for gensort as well.

We fixed this issue by using the TeraInputFormat & TeraOutputFormat classes that are part of the Apache TeraSort example library. This way, the file formats of our jobs are exactly like our validators (teravalidate and/or valsort) expect. Note, the sort implementation is still our own, and does not have any dependency to TeraSort directly.

These line differences and slight variations in file formats were causing the following vsort validation problems:

```
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ ./valsort part-tera
Records: 100
Checksum: 365ed3f3e1
Duplicate keys: 0
SUCCESS - all records are in order
cc@flamedragon:~/cs553-2017-sort-benchmark/diff$ ./valsort part-self
sump pump fatal error: pfunc_get_rec: partial record of 8 bytes found at end of input
```

In the end, we ended up not using the Tera formats anymore because we suspected that they were causing our hang issues on Hadoop and Spark. It ended up not mattering, because even after we reverted the changes, the jobs still refuse to complete.

Hadoop & Spark Infrastructure management

Both our Hadoop and Spark implementations were tested on small datasets that would not take a long time to complete. However, when running on large datasets, a lot of errors were encountered, too many to screenshot here.

Multinode deployment was also a nightmare. We couldn't get it to work after full days of trying to configure and tune it to work with our applications (or any applications in general).

Spark sort validation

Our implementation clearly sorts the rows produced by gensort correctly:

```
NPTNTX7;fGg 0000000000000000000000000000CD4  
NPtXKipyR[K 000000000000000000000000000072CB  
NPYqIKczDq 000000000000000000000000000053BE  
NP` }7A'BD` 000000000000000000000000000038BB  
NPG0lpZ:HI 00000000000000000000000000004BIA  
NPqyJ98zn1 00000000000000000000000000005E92  
NPtF;q@:!x 000000000000000000000000000005D3B  
NQ&u+PU".^ 000000000000000000000000000001F33  
NQ+`jTriVe 000000000000000000000000000000604  
NQ,yx=?oU# 000000000000000000000000000011EE  
NQ/:vvWh/K 000000000000000000000000000007EEE  
NQ/Mp3nX^( 0000000000000000000000000000007E09  
NQ=riH."c~j 000000000000000000000000000001C757  
NQ?v2`2?is 000000000000000000000000000015AB2  
NQFnft=K_5 000000000000000000000000000001A645  
NQZRyOjv=A 0000000000000000000000000000001959B  
NQemTo8`9: 000000000000000000000000000000948E  
NQg,>4/^X? 0000000000000000000000000000074D3  
NQm>1EVH7p 0000000000000000000000000000018D79  
NQr~os".u6 0000000000000000000000000000014D96  
NQts\AoPn" 0000000000000000000000000000063DA  
NQwwx?bPlc 0000000000000000000000000000003D17  
NQzb_0vcV6 00000000000000000000000000000019DA4  
NQ-7%1%Lyb 0000000000000000000000000000009C6C  
NR%M>/l2$A 000000000000000000000000000001C84B  
NR.SlCN;x4 000000000000000000000000000001D7D6  
NR!>LiQ|}cx 000000000000000000000000000001BF5F  
NR10d`[f]Z 00000000000000000000000000000155A4  
NR5^!/H9#' 0000000000000000000000000000099F8  
NR6BLMQ^t? 0000000000000000000000000000014239  
NR?C-A48jQ 000000000000000000000000000001702B  
NR@:_k]3wR 000000000000000000000000000000373F  
NRJC.LUC5$ 00000000000000000000000000000CD2B  
NRP@HVS8_T 000000000000000000000000000005B50  
NRPa`vrh^Q 000000000000000000000000000001DB1  
NRuihw:-NQ 00000000000000000000000000000125F9  
NR[YLa2FzZ 0000000000000000000000000000017EDE  
NRa*b)>E\ 00000000000000000000000000000152FA  
NRi=xdt>H5 00000000000000000000000000000C1BB  
NRkdYiIK{4 000000000000000000000000000001015A  
NRn#g=Hf+j 0000000000000000000000000000006F36  
NS"8>5Z#>/ 0000000000000000000000000000018A9E  
NS`FrIdY0q 0000000000000000000000000000013D3A  
NSGHlvT(Z! 00000000000000000000000000000E8AE
```

This output is from the output directory of the spark job running on top of HDFS.

We tried running the TeraValidate validator, but we get the following error(s):


```

cc@flamedragon:~/cs553-2017-sort-benchmark$ hadoop jar hadoop-mapreduce-examples.jar teravalidate /output /report
17/12/05 12:32:21 INFO client.RMPProxy: Connecting to ResourceManager at flamedragon/127.0.1.1:8032
17/12/05 12:32:21 INFO input.FileInputFormat: Total input paths to process : 2
Spent 28ms computing base-splits.
Spent 2ms computing TeraScheduler splits.
17/12/05 12:32:21 INFO mapreduce.JobSubmitter: number of splits:2
17/12/05 12:32:21 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1512463451507_0024
17/12/05 12:32:22 INFO impl.YarnClientImpl: Submitted application application_1512463451507_0024
17/12/05 12:32:22 INFO mapreduce.Job: The url to track the job: http://flamedragon:8088/proxy/application_1512463451507_0024/
17/12/05 12:32:22 INFO mapreduce.Job: Running job: job_1512463451507_0024
17/12/05 12:32:29 INFO mapreduce.Job: Job job_1512463451507_0024 running in uber mode : false
17/12/05 12:32:29 INFO mapreduce.Job: map 0% reduce 0%
17/12/05 12:32:38 INFO mapreduce.Job: map 100% reduce 0%
17/12/05 12:32:38 INFO mapreduce.Job: Task Id : attempt_1512463451507_0024_m_000000_0, Status : FAILED
Error: java.io.EOFException: read past eof
    at org.apache.hadoop.examples.terasort.TeraInputFormat$TeraRecordReader.nextKeyValue(TeraInputFormat.java:261)
    at org.apache.hadoop.mapred.MapTask$NewTrackingRecordReader.nextKeyValue(MapTask.java:556)
    at org.apache.hadoop.mapreduce.task.MapContextImpl.nextKeyValue(MapContextImpl.java:80)
    at org.apache.hadoop.mapreduce.lib.map.WrappedMapper$Context.nextKeyValue(WrappedMapper.java:91)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:145)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:787)
    at org.apache.hadoop.mapred.MapTask.run(MapTask.java:341)
    at org.apache.hadoop.mapred.YarnChild$2.run(YarnChild.java:164)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:422)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1746)
    at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:158)

Container killed by the ApplicationMaster.
Container killed on request. Exit code is 143
Container exited with a non-zero exit code 143

17/12/05 12:32:38 INFO mapreduce.Job: Task Id : attempt_1512463451507_0024_m_000001_0, Status : FAILED
Error: java.io.EOFException: read past eof
    at org.apache.hadoop.examples.terasort.TeraInputFormat$TeraRecordReader.nextKeyValue(TeraInputFormat.java:261)
    at org.apache.hadoop.mapred.MapTask$NewTrackingRecordReader.nextKeyValue(MapTask.java:556)
    at org.apache.hadoop.mapreduce.task.MapContextImpl.nextKeyValue(MapContextImpl.java:80)
    at org.apache.hadoop.mapreduce.lib.map.WrappedMapper$Context.nextKeyValue(WrappedMapper.java:91)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:145)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:787)
    at org.apache.hadoop.mapred.MapTask.run(MapTask.java:341)
    at org.apache.hadoop.mapred.YarnChild$2.run(YarnChild.java:164)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:422)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1746)
    at org.apache.hadoop.mapred.YarnChild.main(YarnChild.java:158)

```

The error has something to do with the file output format. We have been unable to debug this problem. We are certain that our sort is correct, but there is a very hard-to-debug problem with our output format and we are unsure how to debug it within Spark.

Therefore, we decided to take the more manual route of generating local data using gensort, copying it to HDFS, run the application, pull down the output directory back to the local filesystem, and run a validator there.

7. Conclusions

What conclusions can you draw? Which seems to be best at 1 node scale? How about 8 nodes? Can you predict which would be best at 100 node scale? How about 1000 node scales? Compare your results with those from the Sort Benchmark [9], specifically the winners in 2013 and 2014 who used Hadoop and Spark. Also, what can you learn from the CloudSort benchmark, a report can be found at [10].

2013, 1.42 TB/min

Hadoop

102.5 TB in 4,328 seconds

2100 nodes x

(2 2.3Ghz hexcore Xeon E5-2630, 64 GB memory, 12x3TB disks)

Thomas Graves

Yahoo! Inc.

Their scaled results (adjusted for the number of nodes) is: $102.5 \text{ TB} / 4328 \text{ seconds} / 2100 \text{ nodes} = 102500 \text{ GB} / 4328 \text{ seconds} / 2100 \text{ nodes} = 0.1128 \text{ GB} / \text{s} / \text{node}$

2014, 4.27 TB/min

Apache Spark

100 TB in 1,406 seconds

207 Amazon EC2 i2.8xlarge nodes x

(32 vCores - 2.5Ghz Intel Xeon E5-2670 v2, 244GB memory, 8x800 GB SSD)

Reynold Xin, Parviz Deyhim, Xiangrui Meng,

Ali Ghodsi, Matei Zaharia

Databricks

Their scaled results (adjusted for the number of nodes) is: $100 \text{ TB} / 1406 \text{ seconds} / 207 \text{ nodes} = 100000 / 1406 \text{ seconds} / 207 \text{ nodes} = 0.3436 \text{ GB} / \text{s} / \text{node}$

As far as CloudSort is concerned, the report indicates that it is a benchmark used to test the cost of public clouds for computation on a fixed dataset size (e.g. 100 TB). The reason external sort is used is twofold: 1) It's easy to implement and 2) it is representative of many real world workloads, since it stresses various parts of the compute ecosystem: CPU, as well as OS, storage and network IO. The applications are simple and easy to port to other systems when

they get upgraded. The importance of CloudSort is also that it can be used to highlight the deficiencies in IO-intensive workloads in the virtualized cloud, which generally perform better on in-house data centers.

Screenshots

Here is the entirety of the instances we used on AWS throughout this assignment. Some have been terminated to save costs. But this is was the entirety of our ecosystem:

	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	Key Name	Monitoring	Launch Time
<input type="checkbox"/>	AlexO-slave4	i-0779f12611b081b30	i3.large	us-east-1c	terminated		None		-	-	aws-flamedragon	disabled	December 5, 2017 at 2:42
<input type="checkbox"/>	DavidG-128GBSpark	i-0f590cb61ca0b7a4	i3.large	us-east-1c	running	2/2 checks ...	None	ec2-35-153-133-156.co...	35.153.133.156	-	aws-flamedragon	disabled	December 5, 2017 at 6:55
<input type="checkbox"/>	AlexO-slave3	i-06f51e4a444a1e3b0	i3.large	us-east-1c	terminated		None		-	-	aws-flamedragon	disabled	December 5, 2017 at 2:42
<input type="checkbox"/>	AlexO-128GB-Hadoop	i-06dec0084af319eed	i3.large	us-east-1c	running	2/2 checks ...	None	ec2-54-242-59-75.com...	54.242.59.75	-	aws-flamedragon	disabled	December 5, 2017 at 11:0
<input type="checkbox"/>	David-Hadoop-128GB	i-051613517ade14665	i3.large	us-east-1c	running	2/2 checks ...	None	ec2-34-204-92-185.co...	34.204.92.185	-	aws-flamedragon	disabled	December 5, 2017 at 12:4
<input type="checkbox"/>	AlexO-slave2	i-04c3c12c808cfe55	i3.large	us-east-1c	terminated		None		-	-	aws-flamedragon	disabled	December 5, 2017 at 2:42
<input type="checkbox"/>	AlexO-slave1	i-041275505ef9fc3ce	i3.large	us-east-1c	terminated		None		-	-	aws-flamedragon	disabled	December 5, 2017 at 2:42
<input type="checkbox"/>	DavidG-1TB	i-02852954daf20d60	i3.4xlarge	us-east-1c	running	2/2 checks ...	None	ec2-34-207-103-92.co...	34.207.103.92	-	aws-flamedragon	disabled	December 4, 2017 at 10:5
<input type="checkbox"/>	AlexO-master	i-003923c8fb10e989	i3.large	us-east-1c	terminated		None		-	-	aws-flamedragon	disabled	December 5, 2017 at 2:42