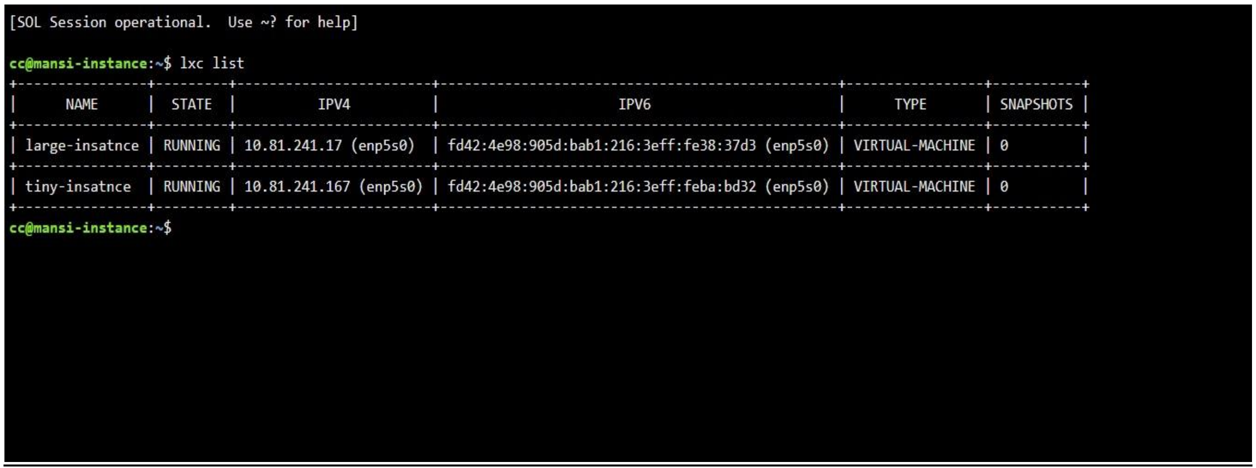
CS553 homework-5

A20556560
Mansi Dinesh

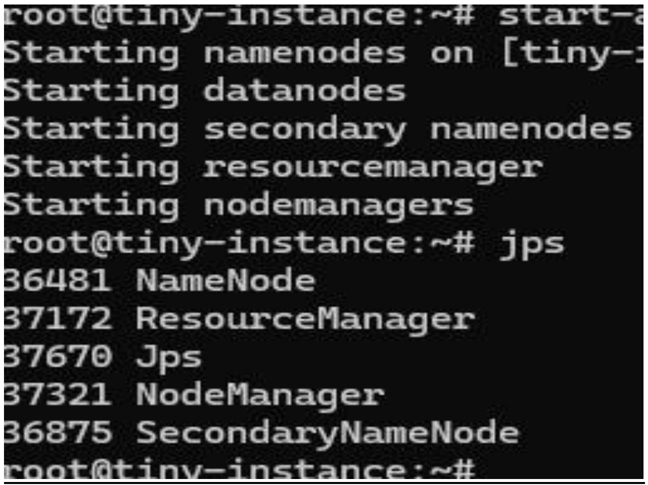
VM’s



```
[SOL Session operational.  Use ~? for help]

cc@mansi-instance:~$ lxc list
+----------------+---------+-----------------------+-----------------------------------------------+-----------------+-----------+
|      NAME      |  STATE  |         IPV4          |                     IPV6                      |      TYPE       | SNAPSHOTS |
+----------------+---------+-----------------------+-----------------------------------------------+-----------------+-----------+
| large-insatnce | RUNNING | 10.81.241.17 (enp5s0)  | fd42:4e98:905d:bab1:216:3eff:fe38:37d3 (enp5s0) | VIRTUAL-MACHINE | 0         |
+----------------+---------+-----------------------+-----------------------------------------------+-----------------+-----------+
| tiny-insatnce  | RUNNING | 10.81.241.167 (enp5s0) | fd42:4e98:905d:bab1:216:3eff:feba:bd32 (enp5s0) | VIRTUAL-MACHINE | 0         |
+----------------+---------+-----------------------+-----------------------------------------------+-----------------+-----------+
cc@mansi-instance:~$
```

MASTER



```
root@tiny-instance:~# start-a
Starting namenodes on [tiny-i
Starting datanodes
Starting secondary namenodes
Starting resourcemanager
Starting nodemanagers
root@tiny-instance:~# jps
36481 NameNode
37172 ResourceManager
37670 Jps
37321 NodeManager
36875 SecondaryNameNode
root@tiny-instance:~#
```

LARGE INSTANCE

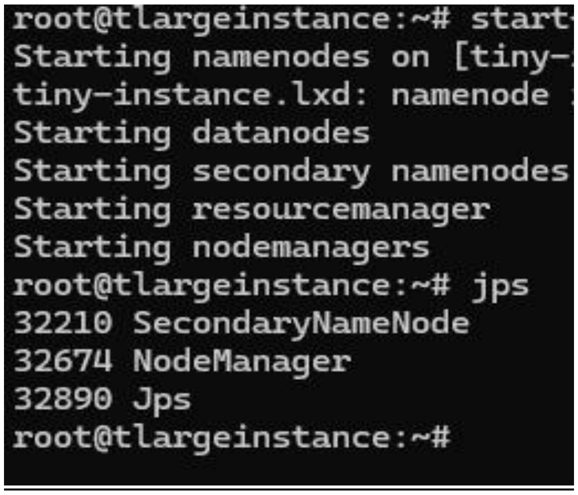
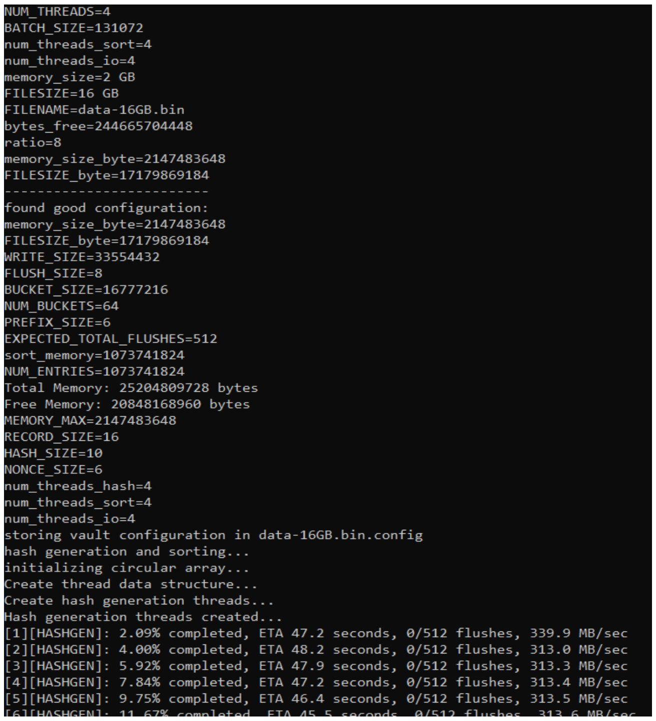

```
root@tlargeinstance:~# start
Starting namenodes on [tiny-
tiny-instance.lxd: namenode
Starting datanodes
Starting secondary namenodes
Starting resourcemanager
Starting nodemanagers
root@tlargeinstance:~# jps
32210 SecondaryNameNode
32674 NodeManager
32890 Jps
root@tlargeinstance:~#
```

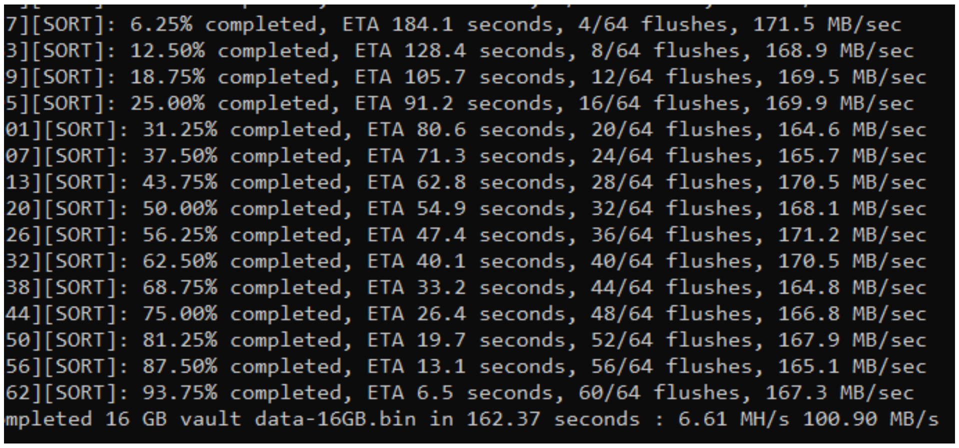
Table 1: Performance (measured in seconds); each instance needs a tiny.instance for the name node

Expeiment	Hashgen	Vault	Hadoop Sort	Spark Sort
1 small.instance, 16GB dataset, 2GB RAM	365.2	162.37	NA	NA
1 small.instance, 32GB dataset, 2GB RAM	582.12	340.83	NA	NA
1 small.instance, 64GB dataset, 2GB RAM	NA	NA	NA	NA
1 large.instance, 16GB dataset, 16GB RAM	1154.2	415.29		
1 large.instance, 32GB dataset, 16GB RAM	1852.2	341.6		
1 large.instance, 64GB dataset, 16GB RAM	2378.76	642.98		
6 small.instances, 16GB dataset	NA	NA		
6 small.instances, 32GB dataset	NA	NA		
6 small.instances, 32GB dataset	NA	NA		

1 small instance, 16GB dataset, 2GB RAM

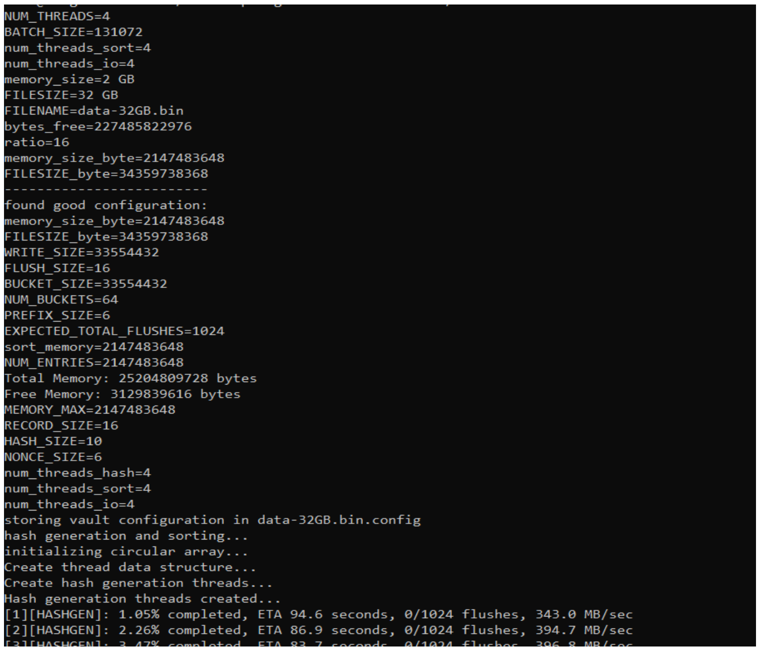


```
NUM_THREADS=4
BATCH_SIZE=131072
num_threads_sort=4
num_threads_io=4
memory_size=2 GB
FILESIZE=16 GB
FILENAME=data-16GB.bin
bytes_free=244665704448
ratio=8
memory_size_byte=2147483648
FILESIZE_byte=17179869184
-------------------------
found good configuration:
memory_size_byte=2147483648
FILESIZE_byte=17179869184
WRITE_SIZE=33554432
FLUSH_SIZE=8
BUCKET_SIZE=16777216
NUM_BUCKETS=64
PREFIX_SIZE=6
EXPECTED_TOTAL_FLUSHES=512
sort_memory=1073741824
NUM_ENTRIES=1073741824
Total Memory: 25204809728 bytes
Free Memory: 20848168960 bytes
MEMORY_MAX=2147483648
RECORD_SIZE=16
HASH_SIZE=10
NONCE_SIZE=6
num_threads_hash=4
num_threads_sort=4
num_threads_io=4
storing vault configuration in data-16GB.bin.config
hash generation and sorting...
initializing circular array...
Create thread data structure...
Create hash generation threads...
Hash generation threads created...
[1][HASHGEN]: 2.09% completed, ETA 47.2 seconds, 0/512 flushes, 339.9 MB/sec
[2][HASHGEN]: 4.00% completed, ETA 48.2 seconds, 0/512 flushes, 313.0 MB/sec
[3][HASHGEN]: 5.92% completed, ETA 47.9 seconds, 0/512 flushes, 313.3 MB/sec
[4][HASHGEN]: 7.84% completed, ETA 47.2 seconds, 0/512 flushes, 313.4 MB/sec
[5][HASHGEN]: 9.75% completed, ETA 46.4 seconds, 0/512 flushes, 313.5 MB/sec
[6][HASHGEN]: 11.67% completed, ETA 45.5 seconds, 0/512 flushes, 313.6 MB/sec
```

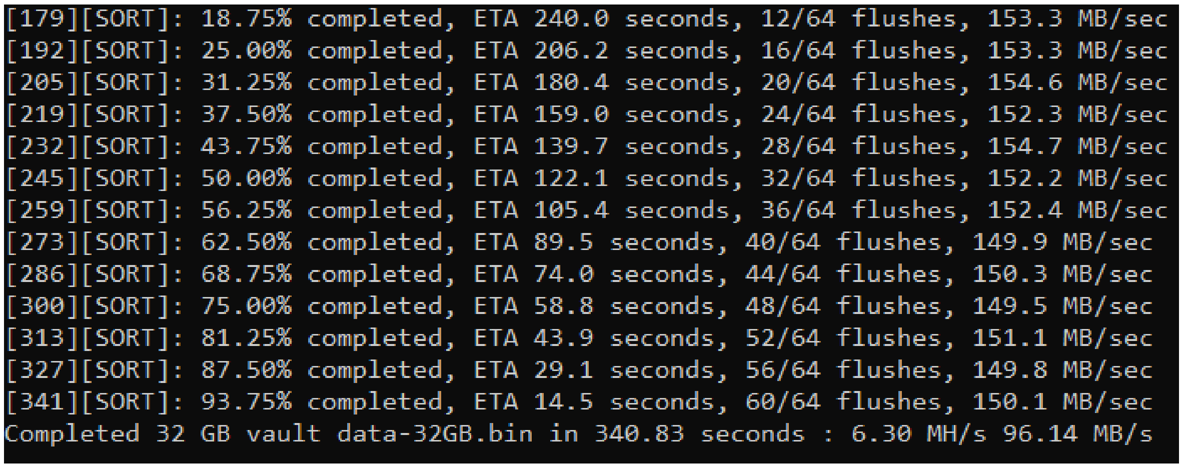


```
7][SORT]: 6.25% completed, ETA 184.1 seconds, 4/64 flushes, 171.5 MB/sec
3][SORT]: 12.50% completed, ETA 128.4 seconds, 8/64 flushes, 168.9 MB/sec
9][SORT]: 18.75% completed, ETA 105.7 seconds, 12/64 flushes, 169.5 MB/sec
5][SORT]: 25.00% completed, ETA 91.2 seconds, 16/64 flushes, 169.9 MB/sec
01][SORT]: 31.25% completed, ETA 80.6 seconds, 20/64 flushes, 164.6 MB/sec
07][SORT]: 37.50% completed, ETA 71.3 seconds, 24/64 flushes, 165.7 MB/sec
13][SORT]: 43.75% completed, ETA 62.8 seconds, 28/64 flushes, 170.5 MB/sec
20][SORT]: 50.00% completed, ETA 54.9 seconds, 32/64 flushes, 168.1 MB/sec
26][SORT]: 56.25% completed, ETA 47.4 seconds, 36/64 flushes, 171.2 MB/sec
32][SORT]: 62.50% completed, ETA 40.1 seconds, 40/64 flushes, 170.5 MB/sec
38][SORT]: 68.75% completed, ETA 33.2 seconds, 44/64 flushes, 164.8 MB/sec
44][SORT]: 75.00% completed, ETA 26.4 seconds, 48/64 flushes, 166.8 MB/sec
50][SORT]: 81.25% completed, ETA 19.7 seconds, 52/64 flushes, 167.9 MB/sec
56][SORT]: 87.50% completed, ETA 13.1 seconds, 56/64 flushes, 165.1 MB/sec
62][SORT]: 93.75% completed, ETA 6.5 seconds, 60/64 flushes, 167.3 MB/sec
mpleted 16 GB vault data-16GB.bin in 162.37 seconds : 6.61 MH/s 100.90 MB/s
```

1 small instance, 32GB dataset, 2GB RAM

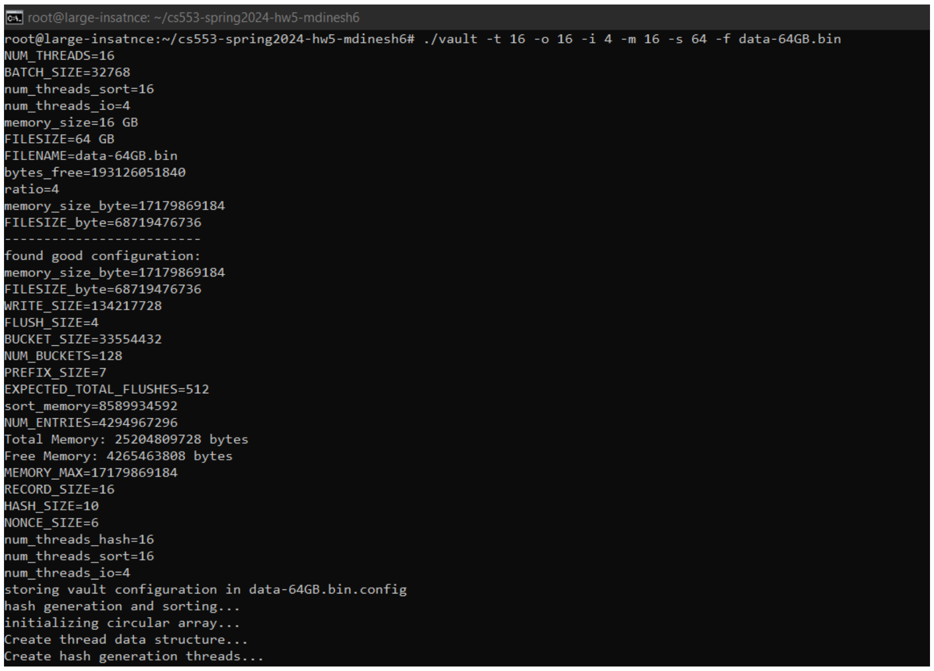


```
NUM_THREADS=4
BATCH_SIZE=131072
num_threads_sort=4
num_threads_io=4
memory_size=2 GB
FILESIZE=32 GB
FILENAME=data-32GB.bin
bytes_free=227485822976
ratio=16
memory_size_byte=2147483648
FILESIZE_byte=34359738368
--------------------------
found good configuration:
memory_size_byte=2147483648
FILESIZE_byte=34359738368
WRITE_SIZE=33554432
FLUSH_SIZE=16
BUCKET_SIZE=33554432
NUM_BUCKETS=64
PREFIX_SIZE=6
EXPECTED_TOTAL_FLUSHES=1024
sort_memory=2147483648
NUM_ENTRIES=2147483648
Total Memory: 25204809728 bytes
Free Memory: 3129839616 bytes
MEMORY_MAX=2147483648
RECORD_SIZE=16
HASH_SIZE=10
NONCE_SIZE=6
num_threads_hash=4
num_threads_sort=4
num_threads_io=4
storing vault configuration in data-32GB.bin.config
hash generation and sorting...
initializing circular array...
Create thread data structure...
Create hash generation threads...
Hash generation threads created...
[1][HASHGEN]: 1.05% completed, ETA 94.6 seconds, 0/1024 flushes, 343.0 MB/sec
[2][HASHGEN]: 2.26% completed, ETA 86.9 seconds, 0/1024 flushes, 394.7 MB/sec
[3][HASHGEN]: 3.47% completed, ETA 83.7 seconds, 0/1024 flushes, 396.8 MB/sec
```



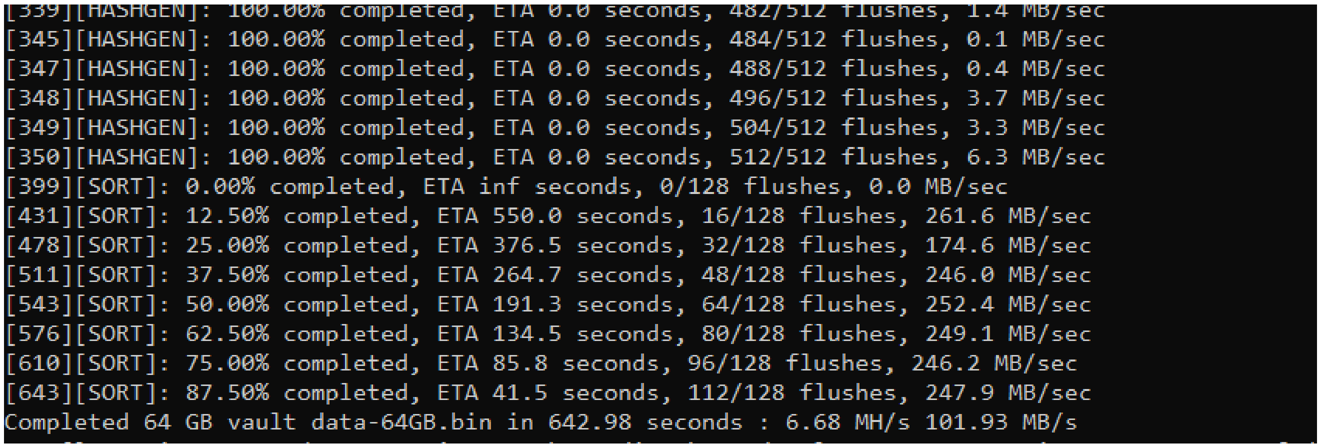
```
[179][SORT]: 18.75% completed, ETA 240.0 seconds, 12/64 flushes, 153.3 MB/sec
[192][SORT]: 25.00% completed, ETA 206.2 seconds, 16/64 flushes, 153.3 MB/sec
[205][SORT]: 31.25% completed, ETA 180.4 seconds, 20/64 flushes, 154.6 MB/sec
[219][SORT]: 37.50% completed, ETA 159.0 seconds, 24/64 flushes, 152.3 MB/sec
[232][SORT]: 43.75% completed, ETA 139.7 seconds, 28/64 flushes, 154.7 MB/sec
[245][SORT]: 50.00% completed, ETA 122.1 seconds, 32/64 flushes, 152.2 MB/sec
[259][SORT]: 56.25% completed, ETA 105.4 seconds, 36/64 flushes, 152.4 MB/sec
[273][SORT]: 62.50% completed, ETA 89.5 seconds, 40/64 flushes, 149.9 MB/sec
[286][SORT]: 68.75% completed, ETA 74.0 seconds, 44/64 flushes, 150.3 MB/sec
[300][SORT]: 75.00% completed, ETA 58.8 seconds, 48/64 flushes, 149.5 MB/sec
[313][SORT]: 81.25% completed, ETA 43.9 seconds, 52/64 flushes, 151.1 MB/sec
[327][SORT]: 87.50% completed, ETA 29.1 seconds, 56/64 flushes, 149.8 MB/sec
[341][SORT]: 93.75% completed, ETA 14.5 seconds, 60/64 flushes, 150.1 MB/sec
Completed 32 GB vault data-32GB.bin in 340.83 seconds : 6.30 MH/s 96.14 MB/s
```

1 large instance, 64GB dataset, 16GB RAM



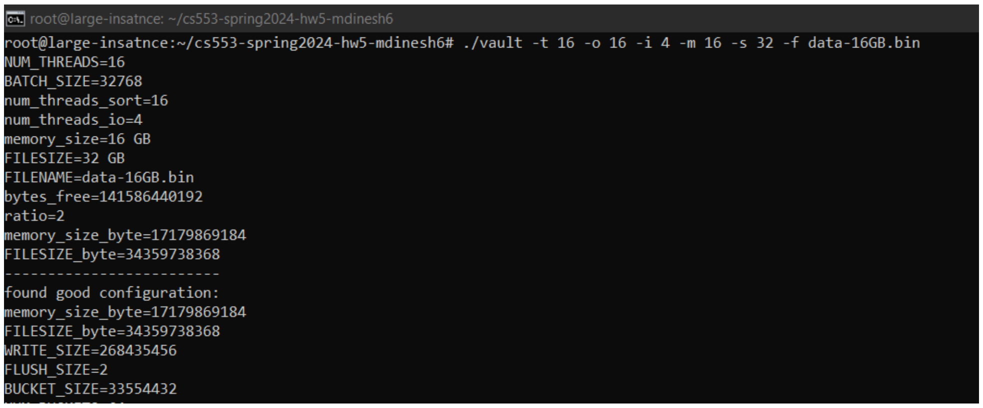
root@large-insatnce: ~/cs553-spring2024-hw5-mdinesh6

```
root@large-insatnce:~/cs553-spring2024-hw5-mdinesh6# ./vault -t 16 -o 16 -i 4 -m 16 -s 64 -f data-64GB.bin
NUM_THREADS=16
BATCH_SIZE=32768
num_threads_sort=16
num_threads_io=4
memory_size=16 GB
FILESIZE=64 GB
FILENAME=data-64GB.bin
bytes_free=193126051840
ratio=4
memory_size_byte=17179869184
FILESIZE_byte=68719476736
------------------------
found good configuration:
memory_size_byte=17179869184
FILESIZE_byte=68719476736
WRITE_SIZE=134217728
FLUSH_SIZE=4
BUCKET_SIZE=33554432
NUM_BUCKETS=128
PREFIX_SIZE=7
EXPECTED_TOTAL_FLUSHES=512
sort_memory=8589934592
NUM_ENTRIES=4294967296
Total Memory: 25204809728 bytes
Free Memory: 4265463808 bytes
MEMORY_MAX=17179869184
RECORD_SIZE=16
HASH_SIZE=10
NONCE_SIZE=6
num_threads_hash=16
num_threads_sort=16
num_threads_io=4
storing vault configuration in data-64GB.bin.config
hash generation and sorting...
initializing circular array...
Create thread data structure...
Create hash generation threads...
```



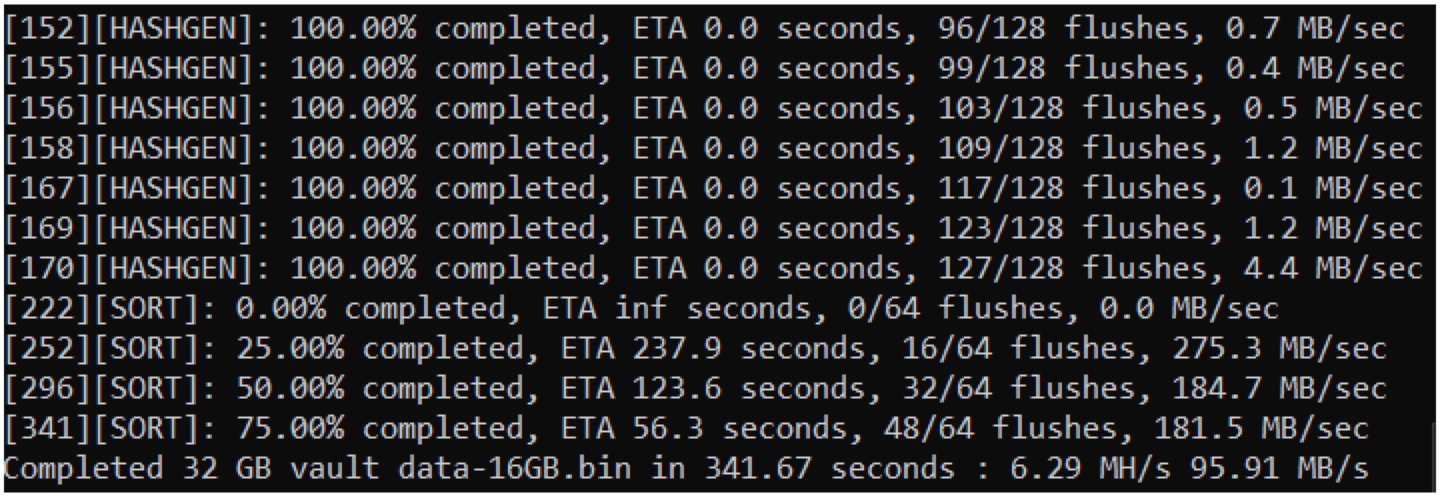
```
[339][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 482/512 flushes, 1.4 MB/sec
[345][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 484/512 flushes, 0.1 MB/sec
[347][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 488/512 flushes, 0.4 MB/sec
[348][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 496/512 flushes, 3.7 MB/sec
[349][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 504/512 flushes, 3.3 MB/sec
[350][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 512/512 flushes, 6.3 MB/sec
[399][SORT]: 0.00% completed, ETA inf seconds, 0/128 flushes, 0.0 MB/sec
[431][SORT]: 12.50% completed, ETA 550.0 seconds, 16/128 flushes, 261.6 MB/sec
[478][SORT]: 25.00% completed, ETA 376.5 seconds, 32/128 flushes, 174.6 MB/sec
[511][SORT]: 37.50% completed, ETA 264.7 seconds, 48/128 flushes, 246.0 MB/sec
[543][SORT]: 50.00% completed, ETA 191.3 seconds, 64/128 flushes, 252.4 MB/sec
[576][SORT]: 62.50% completed, ETA 134.5 seconds, 80/128 flushes, 249.1 MB/sec
[610][SORT]: 75.00% completed, ETA 85.8 seconds, 96/128 flushes, 246.2 MB/sec
[643][SORT]: 87.50% completed, ETA 41.5 seconds, 112/128 flushes, 247.9 MB/sec
Completed 64 GB vault data-64GB.bin in 642.98 seconds : 6.68 MH/s 101.93 MB/s
```

1 large instance, 32GB dataset, 16GB RAM



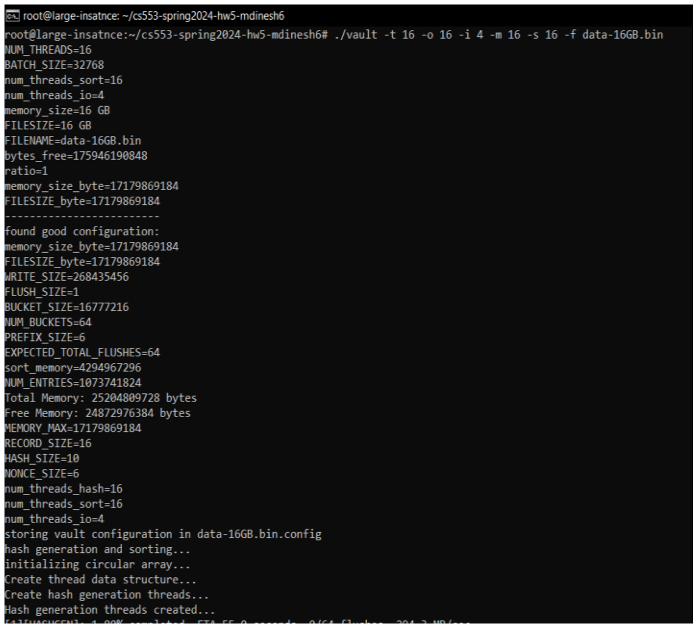
root@large-insatnce: ~/cs553-spring2024-hw5-mdinesh6

```
root@large-insatnce:~/cs553-spring2024-hw5-mdinesh6# ./vault -t 16 -o 16 -i 4 -m 16 -s 32 -f data-16GB.bin
NUM_THREADS=16
BATCH_SIZE=32768
num_threads_sort=16
num_threads_io=4
memory_size=16 GB
FILESIZE=32 GB
FILENAME=data-16GB.bin
bytes_free=141586440192
ratio=2
memory_size_byte=17179869184
FILESIZE_byte=34359738368
------------------------
found good configuration:
memory_size_byte=17179869184
FILESIZE_byte=34359738368
WRITE_SIZE=268435456
FLUSH_SIZE=2
BUCKET_SIZE=33554432
```

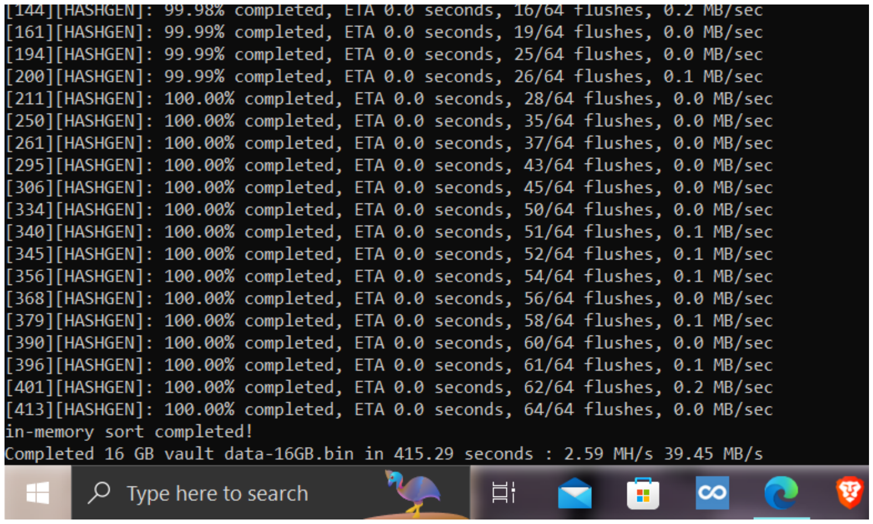


```
[152][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 96/128 flushes, 0.7 MB/sec
[155][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 99/128 flushes, 0.4 MB/sec
[156][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 103/128 flushes, 0.5 MB/sec
[158][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 109/128 flushes, 1.2 MB/sec
[167][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 117/128 flushes, 0.1 MB/sec
[169][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 123/128 flushes, 1.2 MB/sec
[170][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 127/128 flushes, 4.4 MB/sec
[222][SORT]: 0.00% completed, ETA inf seconds, 0/64 flushes, 0.0 MB/sec
[252][SORT]: 25.00% completed, ETA 237.9 seconds, 16/64 flushes, 275.3 MB/sec
[296][SORT]: 50.00% completed, ETA 123.6 seconds, 32/64 flushes, 184.7 MB/sec
[341][SORT]: 75.00% completed, ETA 56.3 seconds, 48/64 flushes, 181.5 MB/sec
Completed 32 GB vault data-16GB.bin in 341.67 seconds : 6.29 MH/s 95.91 MB/s
```

1 large instance, 16GB dataset, 16GB RAM



```
root@large-insatnce: ~/cs553-spring2024-hw5-mdinesh6
root@large-insatnce:~/cs553-spring2024-hw5-mdinesh6# ./vault -t 16 -o 16 -i 4 -m 16 -s 16 -f data-16GB.bin
NUM_THREADS=16
BATCH_SIZE=32768
num_threads_sort=16
num_threads_io=4
memory_size=16 GB
FILESIZE=16 GB
FILENAME=data-16GB.bin
bytes_free=175946190848
ratio=1
memory_size_byte=17179869184
FILESIZE_byte=17179869184
-------------------------
found good configuration:
memory_size_byte=17179869184
FILESIZE_byte=17179869184
WRITE_SIZE=268435456
FLUSH_SIZE=1
BUCKET_SIZE=16777216
NUM_BUCKETS=64
PREFIX_SIZE=6
EXPECTED_TOTAL_FLUSHES=64
sort_memory=4294967296
NUM_ENTRIES=1073741824
Total Memory: 25204809728 bytes
Free Memory: 24872976384 bytes
MEMORY_MAX=17179869184
RECORD_SIZE=16
HASH_SIZE=10
NONCE_SIZE=6
num_threads_hash=16
num_threads_sort=16
num_threads_io=4
storing vault configuration in data-16GB.bin.config
hash generation and sorting...
initializing circular array...
Create thread data structure...
Create hash generation threads...
Hash generation threads created...
[1][HASHGEN]: 1.80% completed, ETA 55.0 seconds, 0/64 flushes, 284.3 MB/sec
```



```
[144][HASHGEN]: 99.98% completed, ETA 0.0 seconds, 16/64 flushes, 0.2 MB/sec
[161][HASHGEN]: 99.99% completed, ETA 0.0 seconds, 19/64 flushes, 0.0 MB/sec
[194][HASHGEN]: 99.99% completed, ETA 0.0 seconds, 25/64 flushes, 0.0 MB/sec
[200][HASHGEN]: 99.99% completed, ETA 0.0 seconds, 26/64 flushes, 0.1 MB/sec
[211][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 28/64 flushes, 0.0 MB/sec
[250][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 35/64 flushes, 0.0 MB/sec
[261][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 37/64 flushes, 0.0 MB/sec
[295][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 43/64 flushes, 0.0 MB/sec
[306][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 45/64 flushes, 0.0 MB/sec
[334][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 50/64 flushes, 0.0 MB/sec
[340][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 51/64 flushes, 0.1 MB/sec
[345][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 52/64 flushes, 0.1 MB/sec
[356][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 54/64 flushes, 0.1 MB/sec
[368][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 56/64 flushes, 0.0 MB/sec
[379][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 58/64 flushes, 0.1 MB/sec
[390][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 60/64 flushes, 0.0 MB/sec
[396][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 61/64 flushes, 0.1 MB/sec
[401][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 62/64 flushes, 0.2 MB/sec
[413][HASHGEN]: 100.00% completed, ETA 0.0 seconds, 64/64 flushes, 0.0 MB/sec
in-memory sort completed!
Completed 16 GB vault data-16GB.bin in 415.29 seconds : 2.59 MH/s 39.45 MB/s
```

1. How many threads, mappers, reducers, you used in each experiment?

   Ans : Spark and Hadoop employed the maptopair function, which accepts a key and maps it from 0 to 9 and a to z. To aggregate the values for each key, the reducers applied an associative and commutative reduction function. In addition, each experiment required 48 threads.

2. How many times did you have to read and write the dataset for each experiment?
   Ans : When the dataset could be accommodated in RAM, MySort and Linuxsort employed internal sorting, which required only one read and write. However,
   when the dataset exceeded the RAM limit, external sorting was applied, resulting in two reads and two writes. Hadoop requires input to be read twice and written twice: the first read occurs at the start of a map job, the second read occurs during the shuffle phase, the first write comes at the completion of a map work,
   and the final write occurs after the reducer process. Spark stores interim results in memory rather than on disc, as Hadoop does. However, when the file size surpasses the RAM capacity of Spark

3. What speedup and efficiency did you achieve?
   Ans: Spark instances function at a faster pace than Hadoop instances, despite the fact that their workers are similar. This is because to Spark's strategy of keeping data in memory before writing, whereas Hadoop publishes the data directly to sorting.

4. Conclusions? Which seems to be best at 1 node scale (1 large.instance)?Is there a difference between 1 small.instance and 1 large.instance? How about 6 nodes (6 small.instance)?

Ans: When running on a single node, a larger instance outperforms a smaller one due to increased memory and processing capability. This advantage becomes increasingly apparent as the data volume increases. When running on four nodes, Hadoop sort and Spark sort both provide distinct benefits for the same dataset, which can be ascribed to the increased number of workers available for the master node to distribute tasks to concurrently. These six nodes cannot use Linux or shared memory sort since they run on a single node. While Linux and MySort appear to be faster than Hadoop and Spark, this is primarily due to lower overhead costs and the ability to handle smaller dataset sizes.

5. What speedup do you achieve with strong scaling between 1 to 6 nodes? What speedup do you achieve with weak scaling between 1 to 6 nodes?
   Ans: Hadoop behaved differently when it was heavily scaled from 16GB on a small instance to 16GB on a large instance, with the larger instance showing to be faster. Furthermore, after analysing diverse datasets, Hadoop's efficiency appeared to rise by a factor of at least two. The Hadoop sort's speed improved as a result of its poor scalability, as seen by the 16GB instance on a small instance. With 64GB on six tiny instances, we saw poor mobility.

6. How many small instances do you need with Hadoop to achieve the same level of performance as your shared memory sort?

Ans: we require approximately 12 tiny Hadoop instances.

7. How about how many small instances do you need with Spark to achieve the same level of performance as you did with your shared memory sort?

Ans: we need 1.28 times in spark to achieve the same performance as memory sort

8. Can you predict which would be best if you had 100 small instances? How about 1000?

Ans: The use of 100 tiny instances allows for the affordable and effective sorting of huge datasets. While raising the number of instances to 100 may improve speed, it also creates issues such as scheduling and synchronization. Without a strong management structure, this could lead to unexpected issues and outcomes. Furthermore, one of the benefits of having a large number of instances is the reliability it gives. Despite the fact that the likelihood of failure grows with the number of nodes, one node's failure has little impact on overall system performance.