Pooja Patel
A20396099

REPORT PA2B

Sort on Hadoop/Spark

PROGRAMMING ASSIGNMENT 02B
CS 553 CLOUD COMPUTING
SPRING 2018

Contents

PROBLEM STATEMENT:

- The main objective of this assignment 2B is to perform sorting using Hadoop and Spark frameworks
- The size of the input files to be sorted is 8 GB , 20 GB and 80 GB
- The main criteria or the points to be noted while experimental on the various input file size with Hadoop and spark are :
    - Computation Time in seconds
    - Data Read (GB)
    - Data Write (GB)
    - I/O Throughput in MB/sec
    - Speedup
    - Efficiency
- Weak scaling (small dataset)
- Strong scaling (large dataset)
- Weak scaling (large dataset)

APPROACH USED FOR HADOOP SORT:

- The data set size to be sorted on using Hadoop is 8 GB , 20 GB , 80 GB input files
- I have this program of map-reduce using java
- My code first takes input file and reads it
- It reads one line and then splits into 2 parts
- The first 10 bytes are key so it takes 1st part as key
- The remaining bytes are value and hence it takes the other part as the value
- The counting of the keys is done by mapper
- Then it's the shuffler who shuffles
- The sorting of the keys and giving the sorted output of the file is done by reducer

APPROACH USED FOR SPARK SORT:

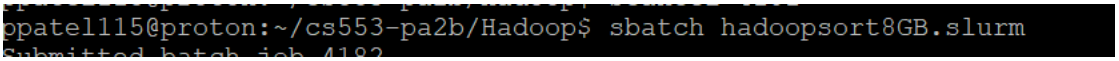
- The sorting of the spark is also done is similar manner

- From the input file the keys are taken and then they are sorted

PERFORMANCE EVALUATION FOR HADOOP:

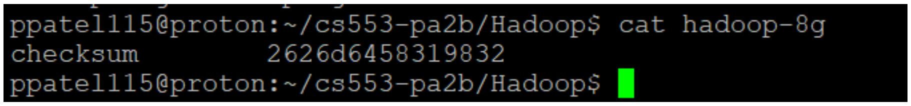
Below are the few screenshots that I have taken while running the experiment for sorting various files sizes using hadoop:

- Below shows how I have submitted job for sorting 8 GB file using Hadoop



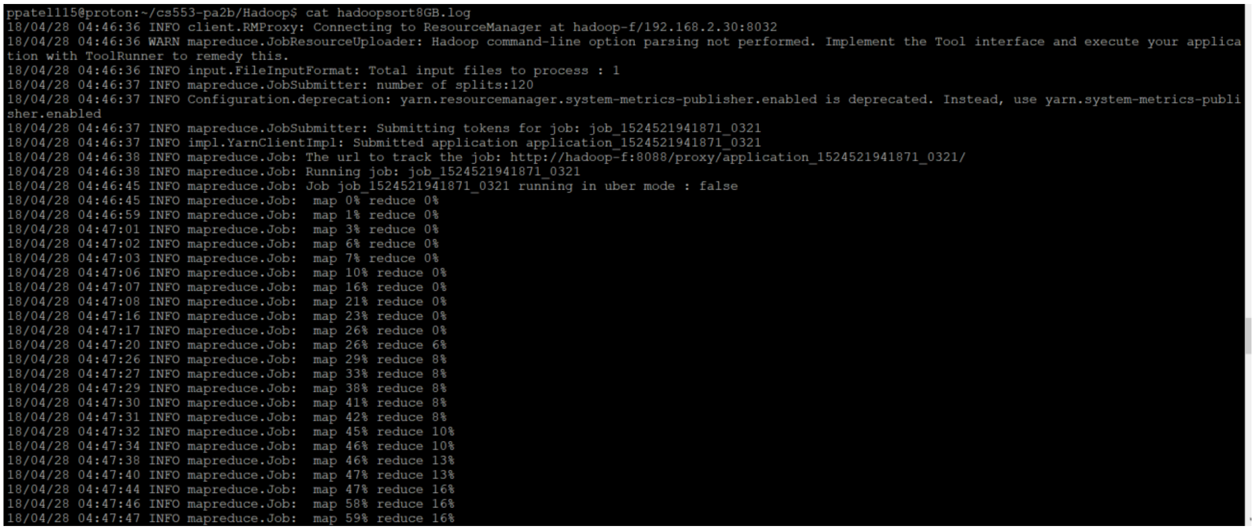
```
ppatel115@proton:~/cs553-pa2b/Hadoop$ sbatch hadoopsort8GB.slurm
```

- Below screenshot shows the Checksum for 8 GB



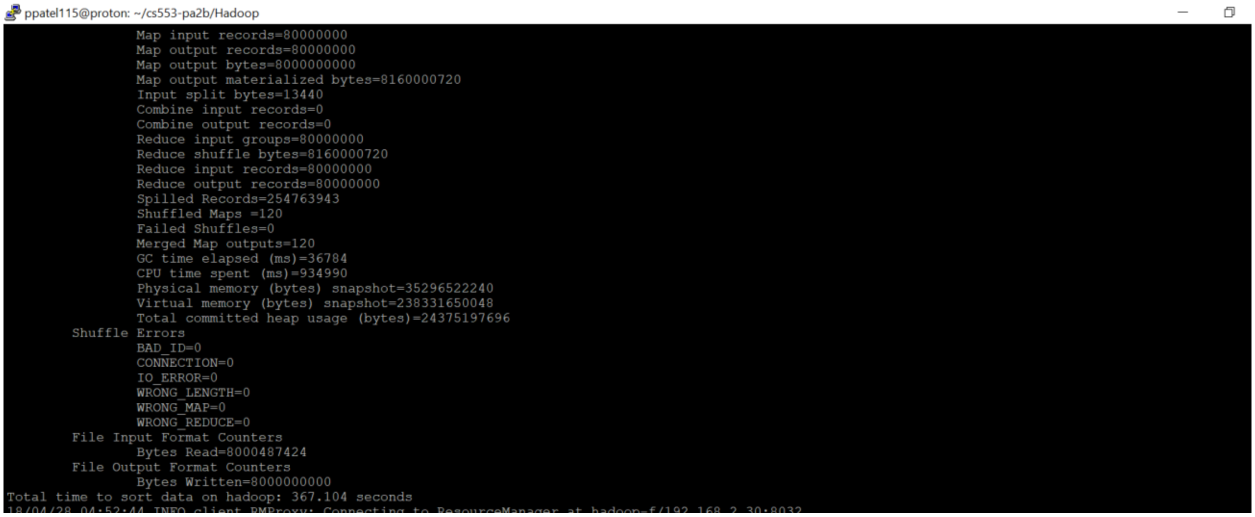
```
ppatel115@proton:~/cs553-pa2b/Hadoop$ cat hadoop-8g
checksum        2626d6458319832
ppatel115@proton:~/cs553-pa2b/Hadoop$
```

- Following picture shows the Log file generated while performing sorting on 8 GB file using hadoop



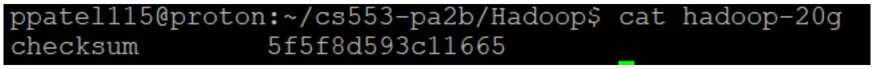
```
ppatel115@proton:~/cs553-pa2b/Hadoop$ cat hadoopsort8GB.log
18/04/28 04:46:36 INFO client.RMProxy: Connecting to ResourceManager at hadoop-f/192.168.2.30:8032
18/04/28 04:46:36 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your applica
tion with ToolRunner to remedy this.
18/04/28 04:46:36 INFO input.FileInputFormat: Total input files to process : 1
18/04/28 04:46:37 INFO mapreduce.JobSubmitter: number of splits:120
18/04/28 04:46:37 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publi
sher.enabled
18/04/28 04:46:37 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1524521941871_0321
18/04/28 04:46:37 INFO impl.YarnClientImpl: Submitted application application_1524521941871_0321
18/04/28 04:46:38 INFO mapreduce.Job: The url to track the job: http://hadoop-f:8088/proxy/application_1524521941871_0321/
18/04/28 04:46:38 INFO mapreduce.Job: Running job: job_1524521941871_0321
18/04/28 04:46:45 INFO mapreduce.Job: Job job_1524521941871_0321 running in uber mode : false
18/04/28 04:46:45 INFO mapreduce.Job:  map 0% reduce 0%
18/04/28 04:46:59 INFO mapreduce.Job:  map 1% reduce 0%
18/04/28 04:47:01 INFO mapreduce.Job:  map 3% reduce 0%
18/04/28 04:47:02 INFO mapreduce.Job:  map 6% reduce 0%
18/04/28 04:47:03 INFO mapreduce.Job:  map 7% reduce 0%
18/04/28 04:47:06 INFO mapreduce.Job:  map 10% reduce 0%
18/04/28 04:47:07 INFO mapreduce.Job:  map 16% reduce 0%
18/04/28 04:47:08 INFO mapreduce.Job:  map 21% reduce 0%
18/04/28 04:47:16 INFO mapreduce.Job:  map 23% reduce 0%
18/04/28 04:47:17 INFO mapreduce.Job:  map 26% reduce 0%
18/04/28 04:47:20 INFO mapreduce.Job:  map 26% reduce 6%
18/04/28 04:47:26 INFO mapreduce.Job:  map 29% reduce 8%
18/04/28 04:47:27 INFO mapreduce.Job:  map 33% reduce 8%
18/04/28 04:47:29 INFO mapreduce.Job:  map 38% reduce 8%
18/04/28 04:47:30 INFO mapreduce.Job:  map 41% reduce 8%
18/04/28 04:47:31 INFO mapreduce.Job:  map 42% reduce 8%
18/04/28 04:47:32 INFO mapreduce.Job:  map 45% reduce 10%
18/04/28 04:47:34 INFO mapreduce.Job:  map 46% reduce 10%
18/04/28 04:47:38 INFO mapreduce.Job:  map 46% reduce 13%
18/04/28 04:47:40 INFO mapreduce.Job:  map 47% reduce 13%
18/04/28 04:47:44 INFO mapreduce.Job:  map 47% reduce 16%
18/04/28 04:47:46 INFO mapreduce.Job:  map 58% reduce 16%
18/04/28 04:47:47 INFO mapreduce.Job:  map 59% reduce 16%
```

The below screenshot captured shows the computation time for 8GB file to be sorted using Hadoop (in log file name “hadoopsort8GB.log”):



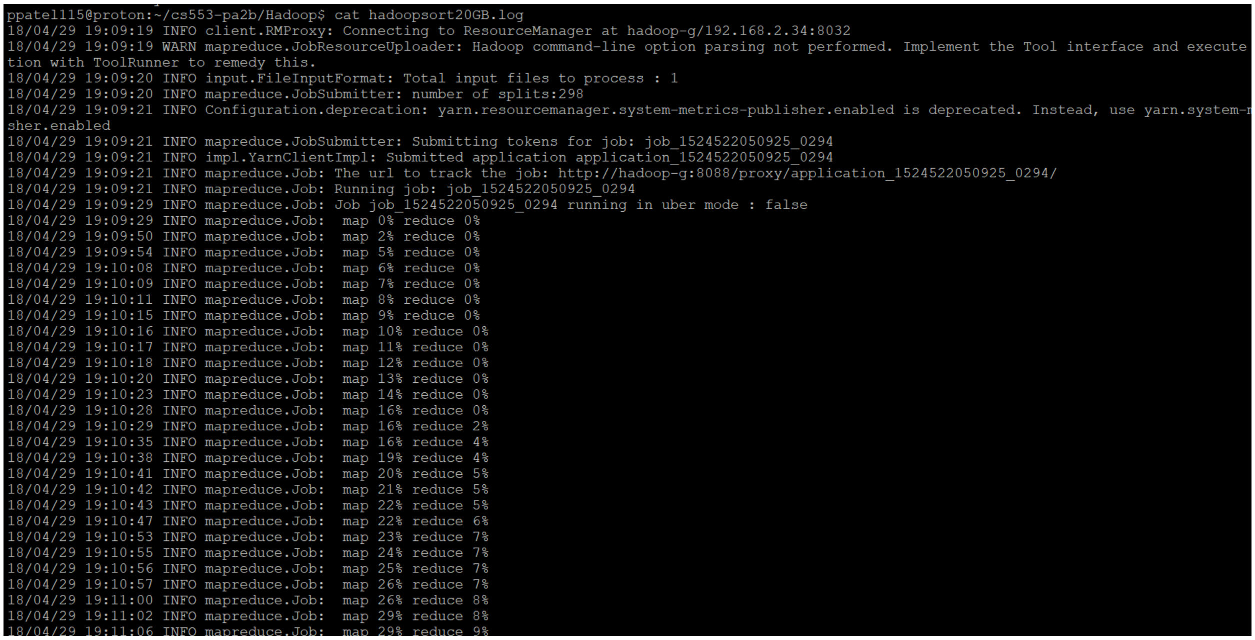
```
                Map input records=80000000
                Map output records=80000000
                Map output bytes=8000000000
                Map output materialized bytes=8160000720
                Input split bytes=13440
                Combine input records=0
                Combine output records=0
                Reduce input groups=80000000
                Reduce shuffle bytes=8160000720
                Reduce input records=80000000
                Reduce output records=80000000
                Spilled Records=254763943
                Shuffled Maps =120
                Failed Shuffles=0
                Merged Map outputs=120
                GC time elapsed (ms)=36784
                CPU time spent (ms)=934990
                Physical memory (bytes) snapshot=35296522240
                Virtual memory (bytes) snapshot=238331650048
                Total committed heap usage (bytes)=24375197696
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=8000487424
        File Output Format Counters
                Bytes Written=8000000000
Total time to sort data on hadoop: 367.104 seconds
18/04/28 04:52:44 INFO client.RMProxy: Connecting to ResourceManager at hadoop-f/192.168.2.30:8032
```

- Below screenshot shows the Checksum for 20GB sorted using hadoop



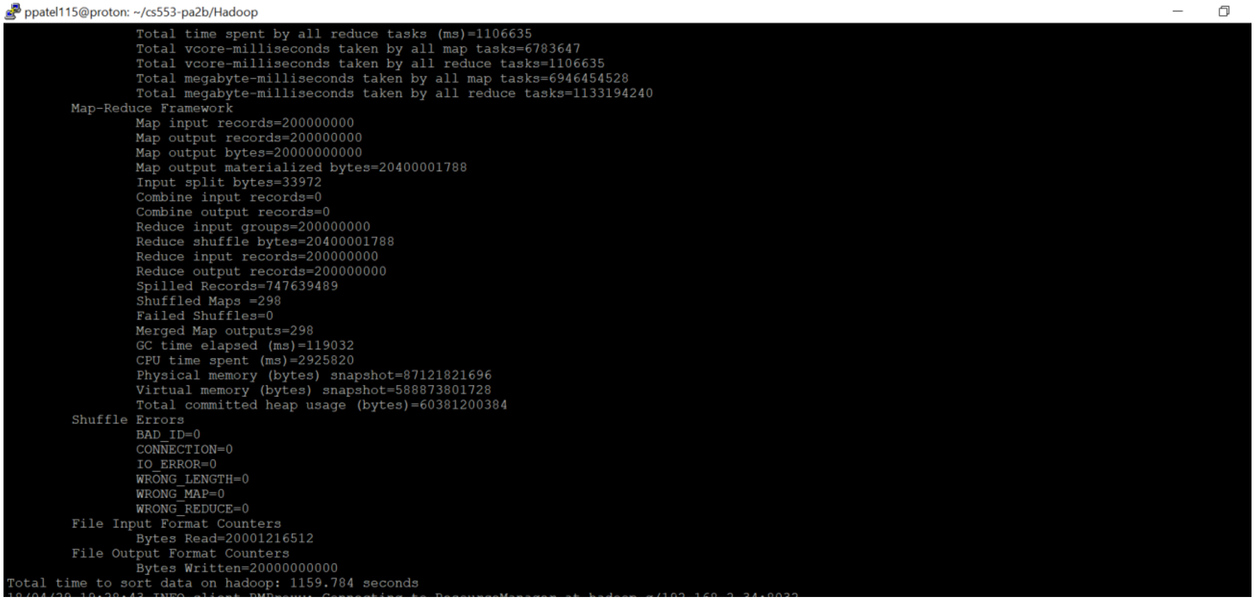
```
ppatel115@proton:~/cs553-pa2b/Hadoop$ cat hadoop-20g
checksum        5f5f8d593c11665
```

- Following figure shows the Log file screenshot ("hadoopsort20GB.log") for 20 GB Hadoop sort



```
ppatel115@proton:~/cs553-pa2b/Hadoop$ cat hadoopsort20GB.log
18/04/29 19:09:19 INFO client.RMProxy: Connecting to ResourceManager at hadoop-g/192.168.2.34:8032
18/04/29 19:09:19 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute
tion with ToolRunner to remedy this.
18/04/29 19:09:20 INFO input.FileInputFormat: Total input files to process : 1
18/04/29 19:09:20 INFO mapreduce.JobSubmitter: number of splits:298
18/04/29 19:09:21 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-m
sher.enabled
18/04/29 19:09:21 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1524522050925_0294
18/04/29 19:09:21 INFO impl.YarnClientImpl: Submitted application application_1524522050925_0294
18/04/29 19:09:21 INFO mapreduce.Job: The url to track the job: http://hadoop-g:8088/proxy/application_1524522050925_0294/
18/04/29 19:09:21 INFO mapreduce.Job: Running job: job_1524522050925_0294
18/04/29 19:09:29 INFO mapreduce.Job: Job job_1524522050925_0294 running in uber mode : false
18/04/29 19:09:29 INFO mapreduce.Job:  map 0% reduce 0%
18/04/29 19:09:50 INFO mapreduce.Job:  map 2% reduce 0%
18/04/29 19:09:54 INFO mapreduce.Job:  map 5% reduce 0%
18/04/29 19:10:08 INFO mapreduce.Job:  map 6% reduce 0%
18/04/29 19:10:09 INFO mapreduce.Job:  map 7% reduce 0%
18/04/29 19:10:11 INFO mapreduce.Job:  map 8% reduce 0%
18/04/29 19:10:15 INFO mapreduce.Job:  map 9% reduce 0%
18/04/29 19:10:16 INFO mapreduce.Job:  map 10% reduce 0%
18/04/29 19:10:17 INFO mapreduce.Job:  map 11% reduce 0%
18/04/29 19:10:18 INFO mapreduce.Job:  map 12% reduce 0%
18/04/29 19:10:20 INFO mapreduce.Job:  map 13% reduce 0%
18/04/29 19:10:23 INFO mapreduce.Job:  map 14% reduce 0%
18/04/29 19:10:28 INFO mapreduce.Job:  map 16% reduce 0%
18/04/29 19:10:29 INFO mapreduce.Job:  map 16% reduce 2%
18/04/29 19:10:35 INFO mapreduce.Job:  map 16% reduce 4%
18/04/29 19:10:38 INFO mapreduce.Job:  map 19% reduce 4%
18/04/29 19:10:41 INFO mapreduce.Job:  map 20% reduce 5%
18/04/29 19:10:42 INFO mapreduce.Job:  map 21% reduce 5%
18/04/29 19:10:43 INFO mapreduce.Job:  map 22% reduce 5%
18/04/29 19:10:47 INFO mapreduce.Job:  map 22% reduce 6%
18/04/29 19:10:53 INFO mapreduce.Job:  map 23% reduce 7%
18/04/29 19:10:55 INFO mapreduce.Job:  map 24% reduce 7%
18/04/29 19:10:56 INFO mapreduce.Job:  map 25% reduce 7%
18/04/29 19:10:57 INFO mapreduce.Job:  map 26% reduce 7%
18/04/29 19:11:00 INFO mapreduce.Job:  map 26% reduce 8%
18/04/29 19:11:02 INFO mapreduce.Job:  map 29% reduce 8%
18/04/29 19:11:06 INFO mapreduce.Job:  map 29% reduce 9%
```

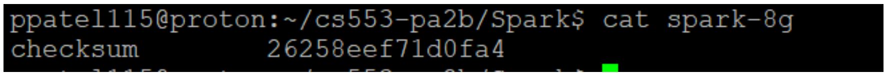
The below shows the computation time for 20 GB file using Hadoop :



```
                Total time spent by all reduce tasks (ms)=1106635
                Total vcore-milliseconds taken by all map tasks=6783647
                Total vcore-milliseconds taken by all reduce tasks=1106635
                Total megabyte-milliseconds taken by all map tasks=6946454528
                Total megabyte-milliseconds taken by all reduce tasks=1133194240
        Map-Reduce Framework
                Map input records=200000000
                Map output records=200000000
                Map output bytes=20000000000
                Map output materialized bytes=20400001788
                Input split bytes=33972
                Combine input records=0
                Combine output records=0
                Reduce input groups=200000000
                Reduce shuffle bytes=20400001788
                Reduce input records=200000000
                Reduce output records=200000000
                Spilled Records=747639489
                Shuffled Maps =298
                Failed Shuffles=0
                Merged Map outputs=298
                GC time elapsed (ms)=119032
                CPU time spent (ms)=2925820
                Physical memory (bytes) snapshot=87121821696
                Virtual memory (bytes) snapshot=588873801728
                Total committed heap usage (bytes)=60381200384
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=20001216512
        File Output Format Counters
                Bytes Written=20000000000
Total time to sort data on hadoop: 1159.784 seconds
```

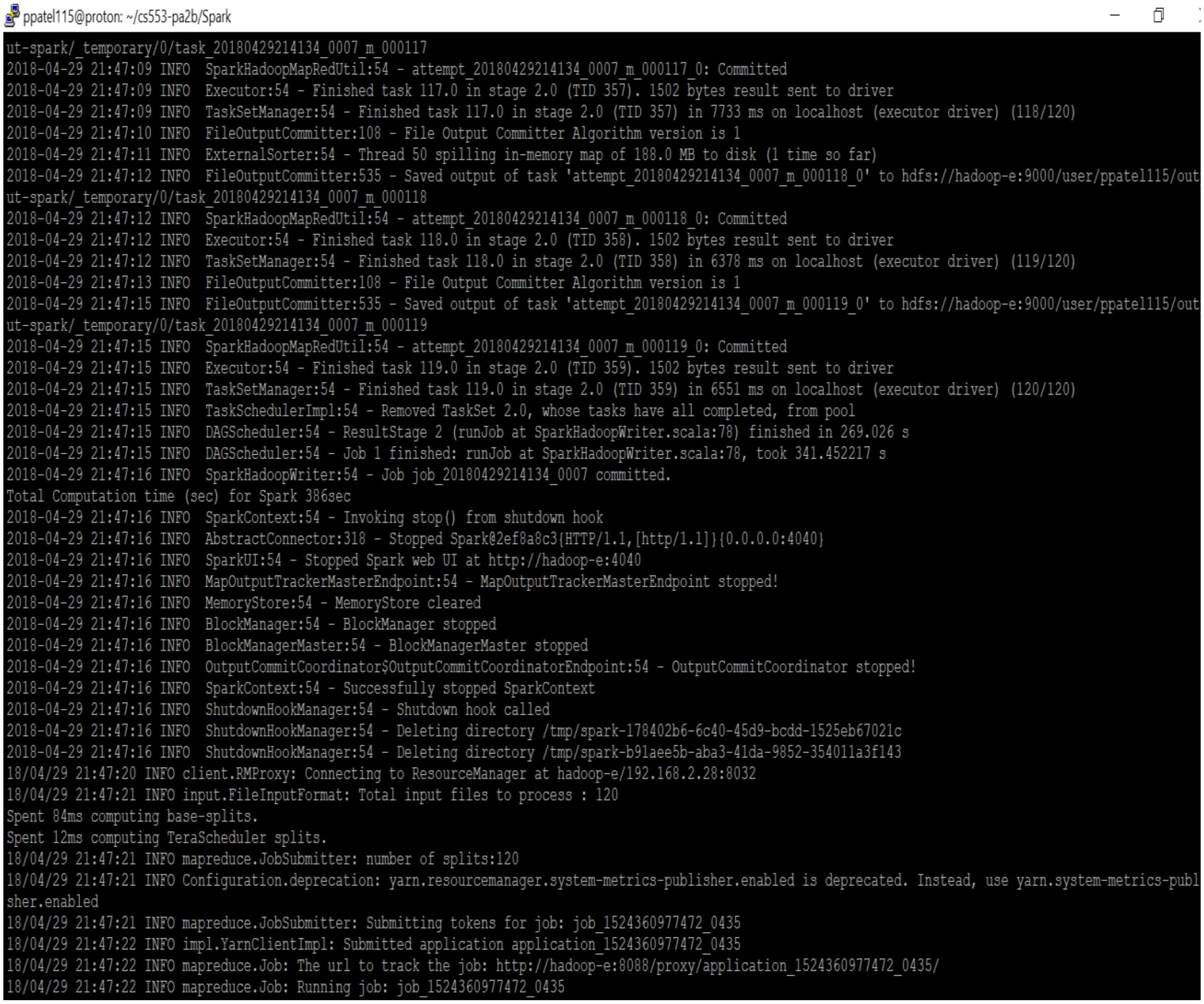
PERFORMANCE EVALUATION FOR SPARK:

- Below picture shows Checksum generated for 8GB file using Spark



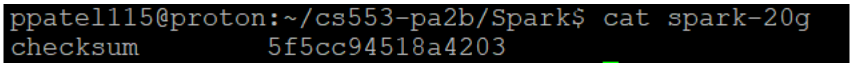
```
ppatel115@proton:~/cs553-pa2b/Spark$ cat spark-8g
checksum          26258eef71d0fa4
```

- Log file for 8 GB file and shows the computation time for sorting 8GB file using spark (Name of the log file is “sparksort8GB.log”):



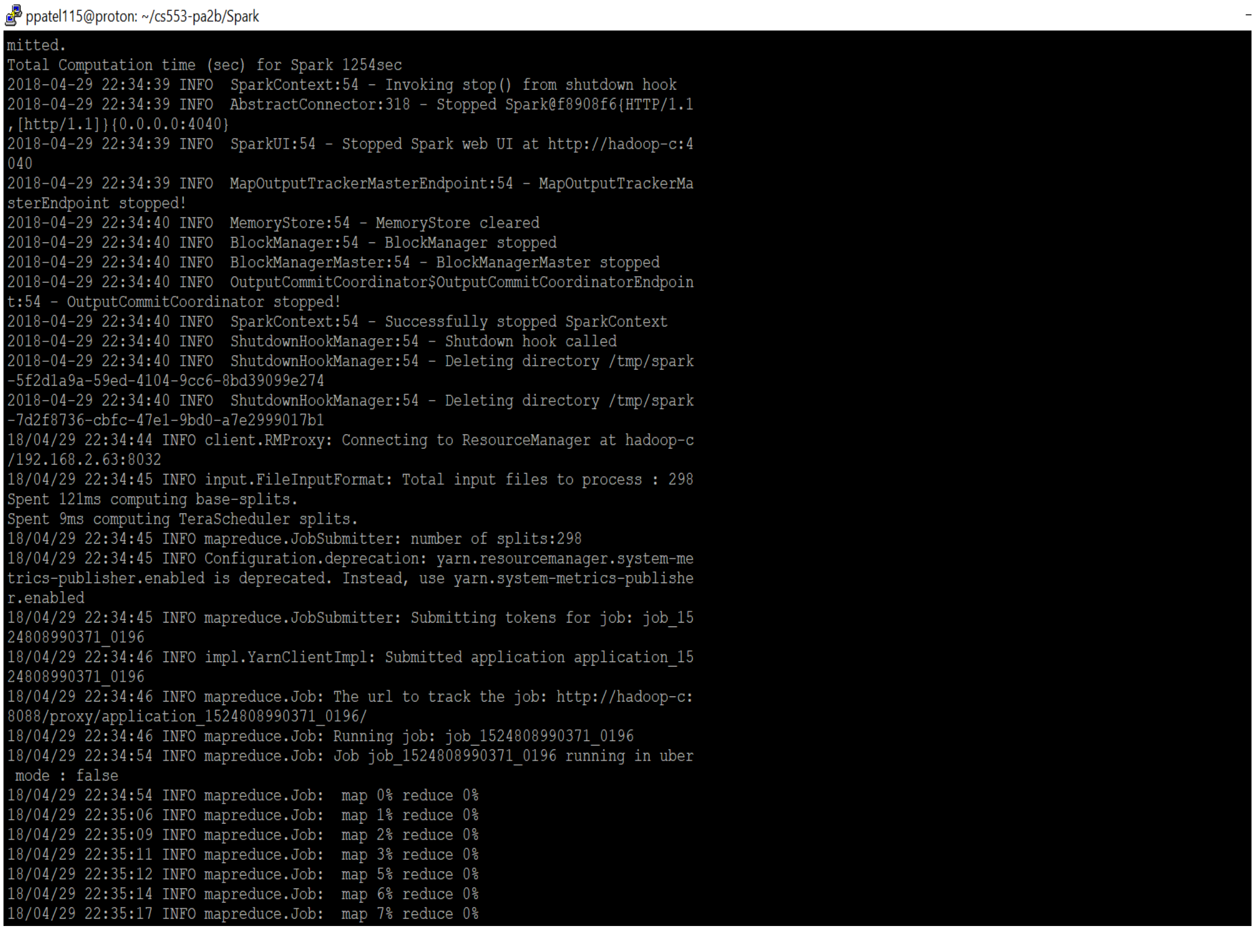
```
ut-spark/_temporary/0/task_20180429214134_0007_m_000117
2018-04-29 21:47:09 INFO  SparkHadoopMapRedUtil:54 - attempt_20180429214134_0007_m_000117_0: Committed
2018-04-29 21:47:09 INFO  Executor:54 - Finished task 117.0 in stage 2.0 (TID 357). 1502 bytes result sent to driver
2018-04-29 21:47:09 INFO  TaskSetManager:54 - Finished task 117.0 in stage 2.0 (TID 357) in 7733 ms on localhost (executor driver) (118/120)
2018-04-29 21:47:10 INFO  FileOutputCommitter:108 - File Output Committer Algorithm version is 1
2018-04-29 21:47:11 INFO  ExternalSorter:54 - Thread 50 spilling in-memory map of 188.0 MB to disk (1 time so far)
2018-04-29 21:47:12 INFO  FileOutputCommitter:535 - Saved output of task 'attempt_20180429214134_0007_m_000118_0' to hdfs://hadoop-e:9000/user/ppatel115/out
ut-spark/_temporary/0/task_20180429214134_0007_m_000118
2018-04-29 21:47:12 INFO  SparkHadoopMapRedUtil:54 - attempt_20180429214134_0007_m_000118_0: Committed
2018-04-29 21:47:12 INFO  Executor:54 - Finished task 118.0 in stage 2.0 (TID 358). 1502 bytes result sent to driver
2018-04-29 21:47:12 INFO  TaskSetManager:54 - Finished task 118.0 in stage 2.0 (TID 358) in 6378 ms on localhost (executor driver) (119/120)
2018-04-29 21:47:13 INFO  FileOutputCommitter:108 - File Output Committer Algorithm version is 1
2018-04-29 21:47:15 INFO  FileOutputCommitter:535 - Saved output of task 'attempt_20180429214134_0007_m_000119_0' to hdfs://hadoop-e:9000/user/ppatel115/out
ut-spark/_temporary/0/task_20180429214134_0007_m_000119
2018-04-29 21:47:15 INFO  SparkHadoopMapRedUtil:54 - attempt_20180429214134_0007_m_000119_0: Committed
2018-04-29 21:47:15 INFO  Executor:54 - Finished task 119.0 in stage 2.0 (TID 359). 1502 bytes result sent to driver
2018-04-29 21:47:15 INFO  TaskSetManager:54 - Finished task 119.0 in stage 2.0 (TID 359) in 6551 ms on localhost (executor driver) (120/120)
2018-04-29 21:47:15 INFO  TaskSchedulerImpl:54 - Removed TaskSet 2.0, whose tasks have all completed, from pool
2018-04-29 21:47:15 INFO  DAGScheduler:54 - ResultStage 2 (runJob at SparkHadoopWriter.scala:78) finished in 269.026 s
2018-04-29 21:47:15 INFO  DAGScheduler:54 - Job 1 finished: runJob at SparkHadoopWriter.scala:78, took 341.452217 s
2018-04-29 21:47:16 INFO  SparkHadoopWriter:54 - Job job_20180429214134_0007 committed.
Total Computation time (sec) for Spark 386sec
2018-04-29 21:47:16 INFO  SparkContext:54 - Invoking stop() from shutdown hook
2018-04-29 21:47:16 INFO  AbstractConnector:318 - Stopped Spark@2ef8a8c3{HTTP/1.1,[http/1.1]}{0.0.0.0:4040}
2018-04-29 21:47:16 INFO  SparkUI:54 - Stopped Spark web UI at http://hadoop-e:4040
2018-04-29 21:47:16 INFO  MapOutputTrackerMasterEndpoint:54 - MapOutputTrackerMasterEndpoint stopped!
2018-04-29 21:47:16 INFO  MemoryStore:54 - MemoryStore cleared
2018-04-29 21:47:16 INFO  BlockManager:54 - BlockManager stopped
2018-04-29 21:47:16 INFO  BlockManagerMaster:54 - BlockManagerMaster stopped
2018-04-29 21:47:16 INFO  OutputCommitCoordinator$OutputCommitCoordinatorEndpoint:54 - OutputCommitCoordinator stopped!
2018-04-29 21:47:16 INFO  SparkContext:54 - Successfully stopped SparkContext
2018-04-29 21:47:16 INFO  ShutdownHookManager:54 - Shutdown hook called
2018-04-29 21:47:16 INFO  ShutdownHookManager:54 - Deleting directory /tmp/spark-178402b6-6c40-45d9-bcdd-1525eb67021c
2018-04-29 21:47:16 INFO  ShutdownHookManager:54 - Deleting directory /tmp/spark-b91aee5b-aba3-41da-9852-354011a3f143
18/04/29 21:47:20 INFO client.RMProxy: Connecting to ResourceManager at hadoop-e/192.168.2.28:8032
18/04/29 21:47:21 INFO input.FileInputFormat: Total input files to process : 120
Spent 84ms computing base-splits.
Spent 12ms computing TeraScheduler splits.
18/04/29 21:47:21 INFO mapreduce.JobSubmitter: number of splits:120
18/04/29 21:47:21 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publ
sher.enabled
18/04/29 21:47:21 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1524360977472_0435
18/04/29 21:47:22 INFO impl.YarnClientImpl: Submitted application application_1524360977472_0435
18/04/29 21:47:22 INFO mapreduce.Job: The url to track the job: http://hadoop-e:8088/proxy/application_1524360977472_0435/
18/04/29 21:47:22 INFO mapreduce.Job: Running job: job_1524360977472_0435
```

- Below picture shows Checksum generated for 20 GB file using Spark



```
ppatel115@proton:~/cs553-pa2b/Spark$ cat spark-20g
checksum        5f5cc94518a4203
```

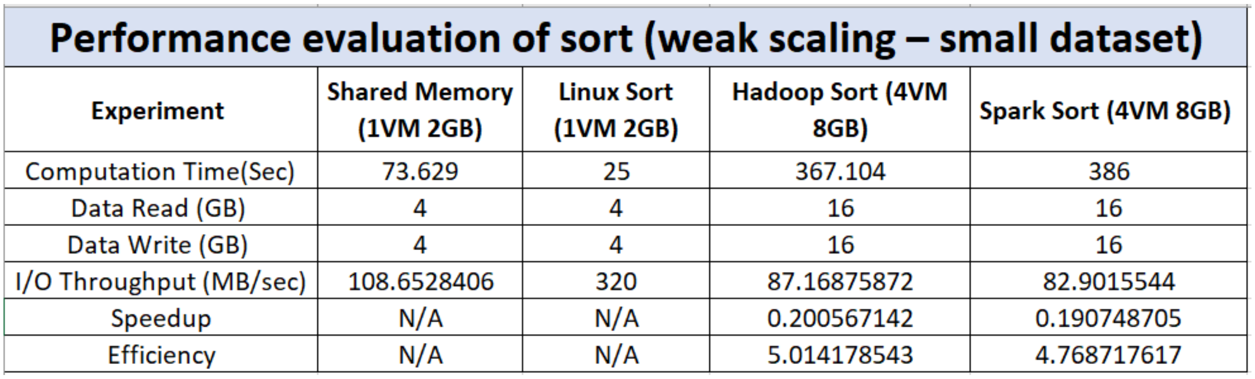
- Log file for 20 GB file and shows the computation time for sorting 20 GB file using spark (Name of the log file is “sparksort20GB.log”):



```
mitted.
Total Computation time (sec) for Spark 1254sec
2018-04-29 22:34:39 INFO  SparkContext:54 - Invoking stop() from shutdown hook
2018-04-29 22:34:39 INFO  AbstractConnector:318 - Stopped Spark@f8908f6{HTTP/1.1
,[http/1.1]}{0.0.0.0:4040}
2018-04-29 22:34:39 INFO  SparkUI:54 - Stopped Spark web UI at http://hadoop-c:4
040
2018-04-29 22:34:39 INFO  MapOutputTrackerMasterEndpoint:54 - MapOutputTrackerMa
sterEndpoint stopped!
2018-04-29 22:34:40 INFO  MemoryStore:54 - MemoryStore cleared
2018-04-29 22:34:40 INFO  BlockManager:54 - BlockManager stopped
2018-04-29 22:34:40 INFO  BlockManagerMaster:54 - BlockManagerMaster stopped
2018-04-29 22:34:40 INFO  OutputCommitCoordinator$OutputCommitCoordinatorEndpoin
t:54 - OutputCommitCoordinator stopped!
2018-04-29 22:34:40 INFO  SparkContext:54 - Successfully stopped SparkContext
2018-04-29 22:34:40 INFO  ShutdownHookManager:54 - Shutdown hook called
2018-04-29 22:34:40 INFO  ShutdownHookManager:54 - Deleting directory /tmp/spark
-5f2d1a9a-59ed-4104-9cc6-8bd39099e274
2018-04-29 22:34:40 INFO  ShutdownHookManager:54 - Deleting directory /tmp/spark
-7d2f8736-cbfc-47e1-9bd0-a7e2999017b1
18/04/29 22:34:44 INFO client.RMProxy: Connecting to ResourceManager at hadoop-c
/192.168.2.63:8032
18/04/29 22:34:45 INFO input.FileInputFormat: Total input files to process : 298
Spent 121ms computing base-splits.
Spent 9ms computing TeraScheduler splits.
18/04/29 22:34:45 INFO mapreduce.JobSubmitter: number of splits:298
18/04/29 22:34:45 INFO Configuration.deprecation: yarn.resourcemanager.system-me
trics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publishe
r.enabled
18/04/29 22:34:45 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_15
24808990371_0196
18/04/29 22:34:46 INFO impl.YarnClientImpl: Submitted application application_15
24808990371_0196
18/04/29 22:34:46 INFO mapreduce.Job: The url to track the job: http://hadoop-c:
8088/proxy/application_1524808990371_0196/
18/04/29 22:34:46 INFO mapreduce.Job: Running job: job_1524808990371_0196
18/04/29 22:34:54 INFO mapreduce.Job: Job job_1524808990371_0196 running in uber
 mode : false
18/04/29 22:34:54 INFO mapreduce.Job:  map 0% reduce 0%
18/04/29 22:35:06 INFO mapreduce.Job:  map 1% reduce 0%
18/04/29 22:35:09 INFO mapreduce.Job:  map 2% reduce 0%
18/04/29 22:35:11 INFO mapreduce.Job:  map 3% reduce 0%
18/04/29 22:35:12 INFO mapreduce.Job:  map 5% reduce 0%
18/04/29 22:35:14 INFO mapreduce.Job:  map 6% reduce 0%
18/04/29 22:35:17 INFO mapreduce.Job:  map 7% reduce 0%
```

PERFORMANCE EVALUATION TABLE:

WEAK SCALING SMALL DATASET



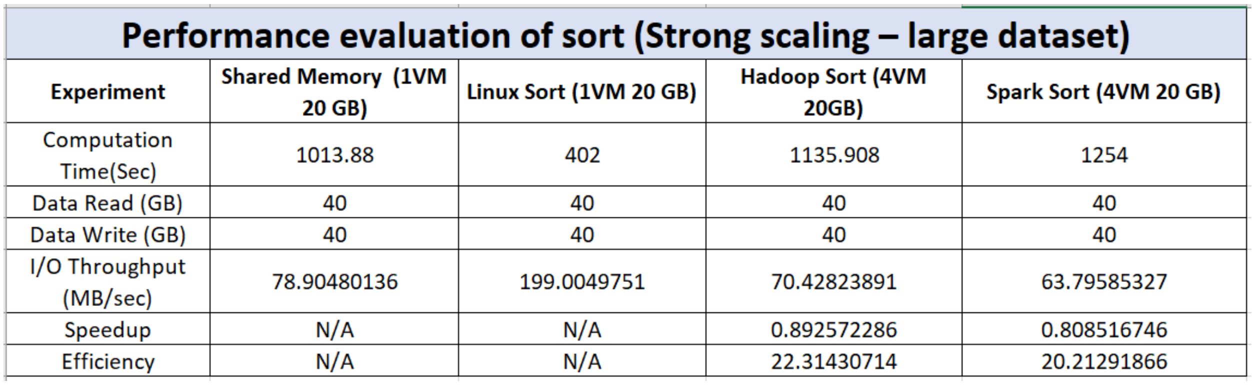
Performance evaluation of sort (weak scaling – small dataset)

Experiment	Shared Memory (1VM 2GB)	Linux Sort (1VM 2GB)	Hadoop Sort (4VM 8GB)	Spark Sort (4VM 8GB)
Computation Time(Sec)	73.629	25	367.104	386
Data Read (GB)	4	4	16	16
Data Write (GB)	4	4	16	16
I/O Throughput (MB/sec)	108.6528406	320	87.16875872	82.9015544
Speedup	N/A	N/A	0.200567142	0.190748705
Efficiency	N/A	N/A	5.014178543	4.768717617

Table 1 : Performance evaluation of sort (Weak Scaling – Small Dataset)

- Table 1 shows the weak scaling (Small dataset) results
- Here the workload is fixed per VM and we increase the cores and hence it is weak scaling and this is with 2 GB dataset per VM so it is “small dataset” weak scaling
- Table 1 shows the results for computation time, data read , data write , I/O Throughput , speedup and efficiency for different sorting techniques like Hadoop , spark , linux sort and shared memory

STRONG SCALING LARGE DATASET



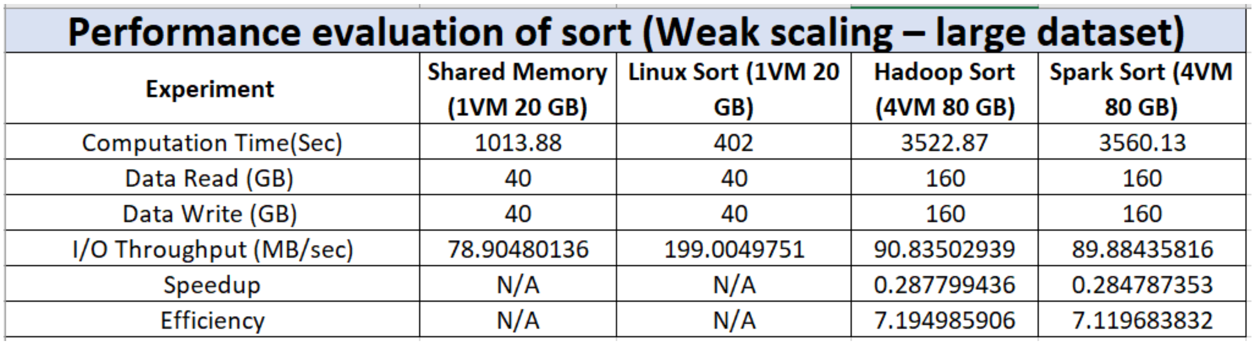
Performance evaluation of sort (Strong scaling – large dataset)

Experiment	Shared Memory (1VM 20 GB)	Linux Sort (1VM 20 GB)	Hadoop Sort (4VM 20GB)	Spark Sort (4VM 20 GB)
Computation Time(Sec)	1013.88	402	1135.908	1254
Data Read (GB)	40	40	40	40
Data Write (GB)	40	40	40	40
I/O Throughput (MB/sec)	78.90480136	199.0049751	70.42823891	63.79585327
Speedup	N/A	N/A	0.892572286	0.808516746
Efficiency	N/A	N/A	22.31430714	20.21291866

Table 2 : Performance evaluation of sort (Strong Scaling – Large Dataset)

- Table 2 shows the strong scaling (large dataset) results
- Here the workload is fixed overall and we increase the cores by keeping the fixed workload and hence it is strong scaling and this is with fixed workload of 20 GB so it is “large dataset” strong scaling
- Table 2 shows the results for computation time, data read , data write , I/O Throughput , speedup and efficiency for different sorting techniques like Hadoop , spark , linux sort and shared memory

WEAK SCALING LARGE DATASET



Performance evaluation of sort (Weak scaling – large dataset)				
Experiment	**Shared Memory (1VM 20 GB)**	**Linux Sort (1VM 20 GB)**	**Hadoop Sort (4VM 80 GB)**	**Spark Sort (4VM 80 GB)**
Computation Time(Sec)	1013.88	402	3522.87	3560.13
Data Read (GB)	40	40	160	160
Data Write (GB)	40	40	160	160
I/O Throughput (MB/sec)	78.90480136	199.0049751	90.83502939	89.88435816
Speedup	N/A	N/A	0.287799436	0.284787353
Efficiency	N/A	N/A	7.194985906	7.119683832

Table 3 : Performance evaluation of sort (Weak Scaling – Large Dataset)

- Table 3 shows the weak scaling (Large dataset) results
- Here the workload is fixed per VM and we increase the cores and hence it is weak scaling and this is with 20 GB dataset per VM so it is “large dataset” weak scaling
- Table 3 shows the results for computation time, data read , data write , I/O Throughput , speedup and efficiency for different sorting techniques like Hadoop , spark , linux sort and shared memory

ANALYSIS ON RESULTS OBTAINED:

WEAK SCALING SMALL DATASET

Analysis 1 : Comparison : Computation Time (sec)

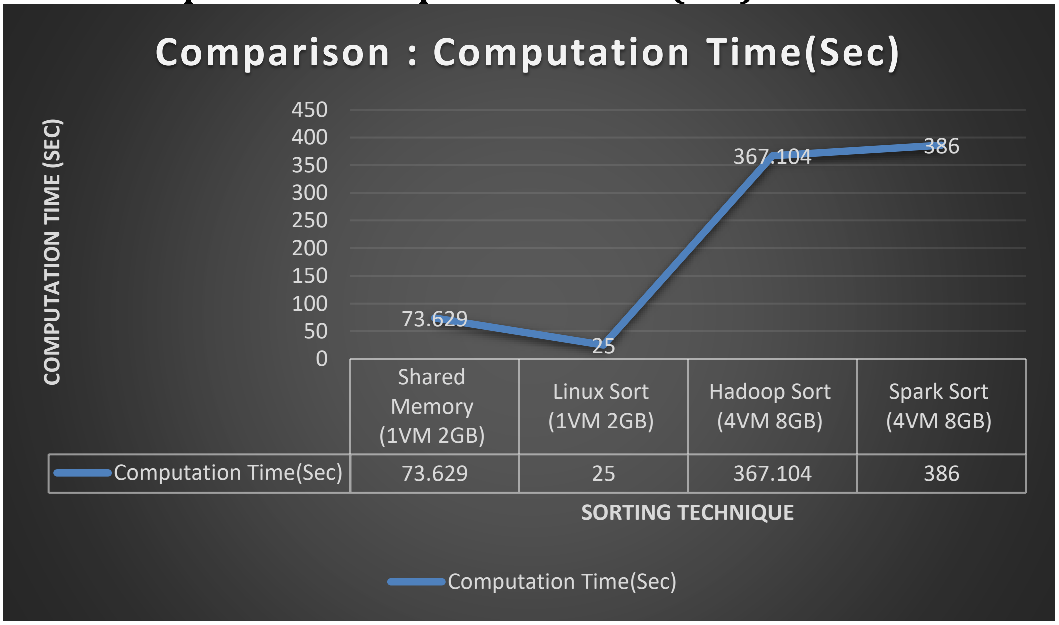


Chart 1 : Comparison : Computation Time (sec)

- The above chart compares the computation time taken in seconds using various sorting techniques and with different no. of VMs. I observed that Linux sort took the least time to sort 2GB data on 1 VM and then shared memory followed by Hadoop sort and lastly spark sort

Analysis 2 : I/O Throughput (MB/sec) Comparison

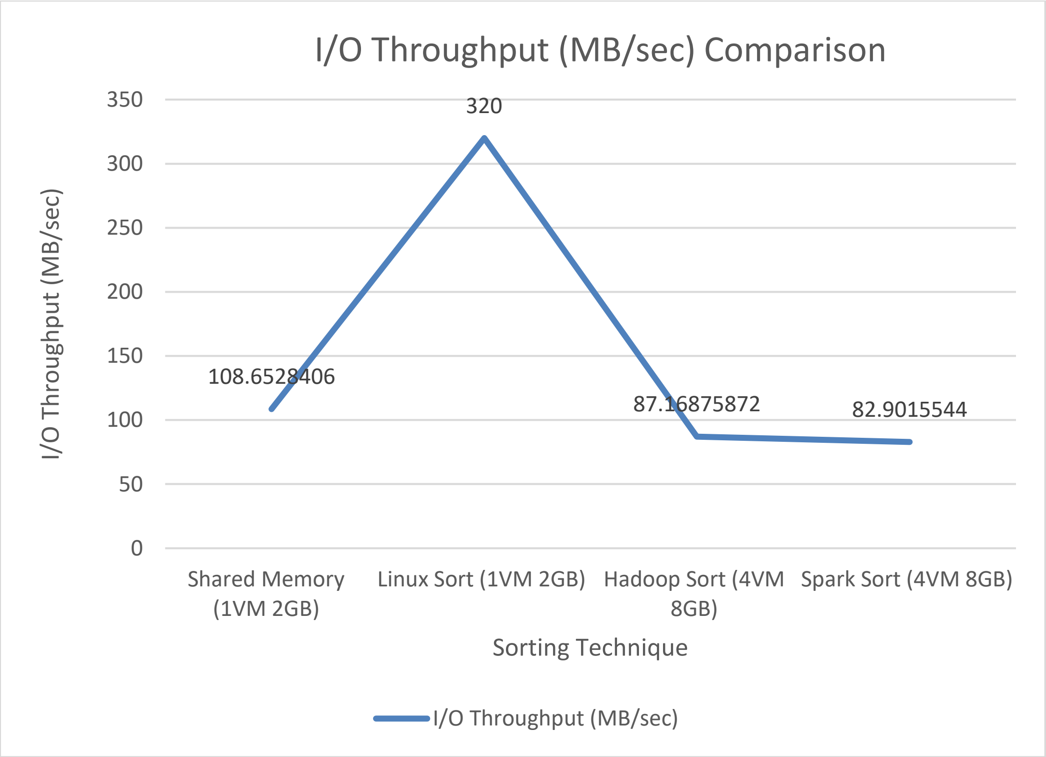


Chart 2 : : I/O Throughput (MB/sec) Comparison

- The above chart compares the I/O throughput (MB/sec) using various sorting techniques and with different no. of VMs. I observed that Linux sort had the highest throughput as it implements efficient sorting algorithm and then followed by shared memory then Hadoop sort and then spark sort

Analysis 3 : Comparing Hadoop and Spark Sort Speedup

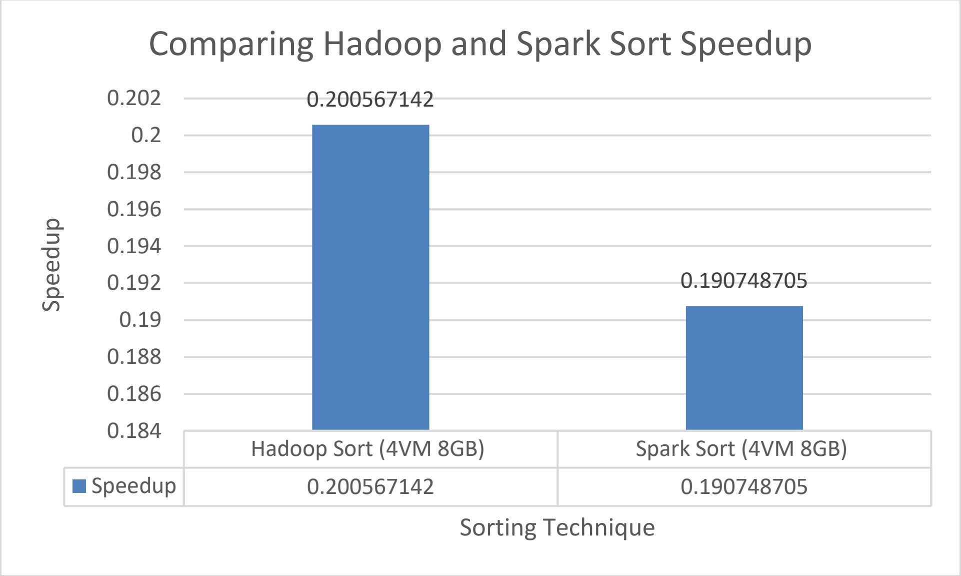


Chart 3 : Comparing Hadoop and Spark Sort Speedup

- The above chart compares speedup for Hadoop and Spark sort(4VM 8GB). From the graph shown Hadoop sort has higher speedup than the spark sort

Analysis 4 : Efficiency Comparison : Hadoop vs Spark Sort

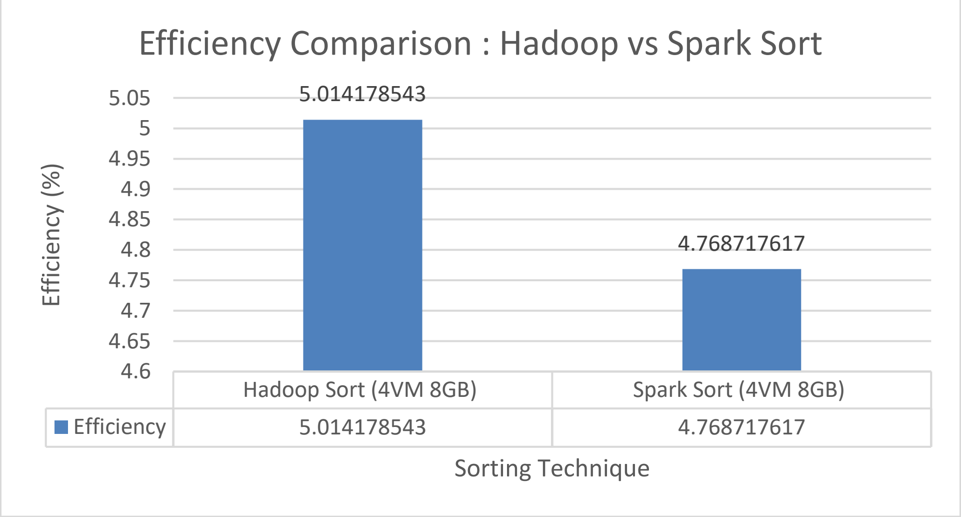


Chart 4 : Efficiency Comparison : Hadoop vs Spark Sort

- The above chart compares the efficiency (%) for Hadoop and Spark sort(4VM 8GB). From the graph shown Hadoop sort has higher efficiency than the spark sort

STRONG SCALING LARGE DATASET

Analysis 1 : Comparison : Computation Time (sec)

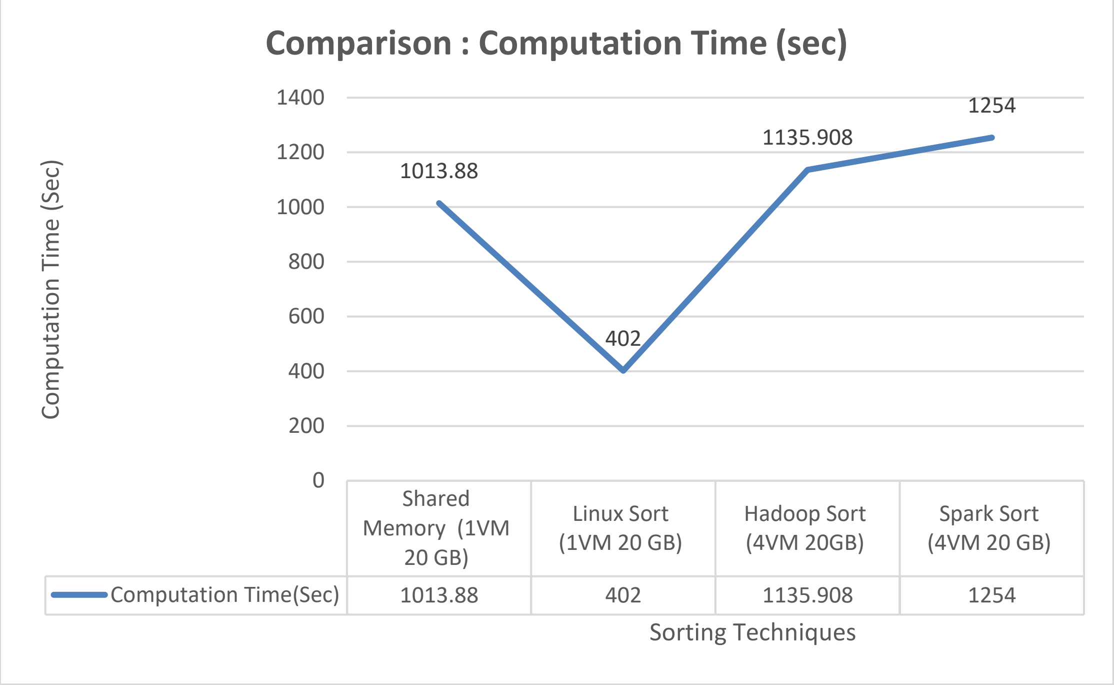


Chart 5 : Comparison : Computation Time (sec)

- The above chart compares the computation time taken in seconds using various sorting techniques and with different no. of VMs. I observed that Linux sort took the least time to sort 20 GB data on 1 VM and then shared memory followed by Hadoop sort and lastly spark sort

Analysis 2 : I/O Throughput (MB/sec) Comparison

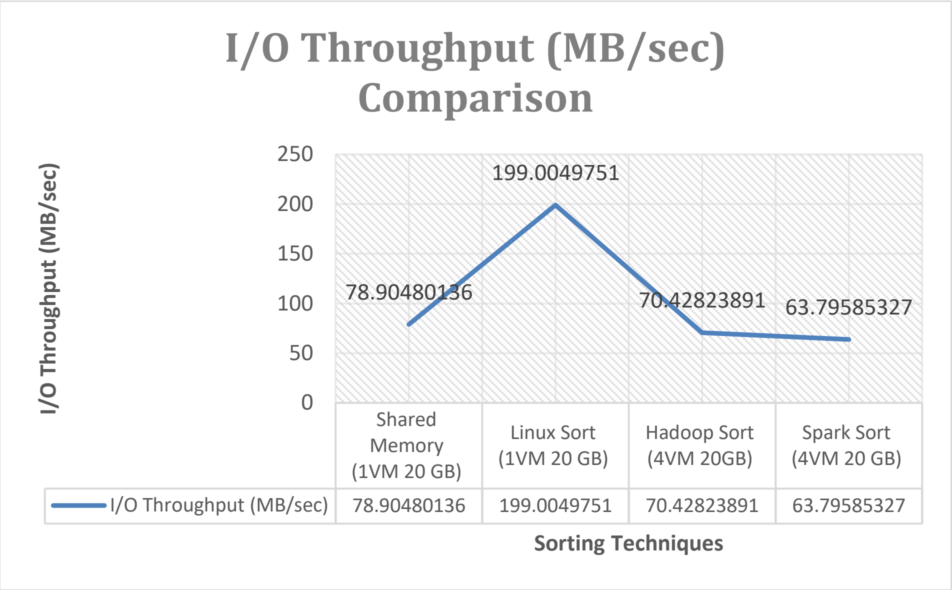


Chart 6 : I/O Throughput (MB/sec) Comparison

- The above chart compares the I/O throughput (MB/sec) using various sorting techniques and with different no. of VMs. I observed that Linux sort had the highest throughput as it implements efficient sorting algorithm and then followed by shared memory then Hadoop sort and then spark sort

Analysis 3 : Comparing Hadoop and Spark Sort Speedup

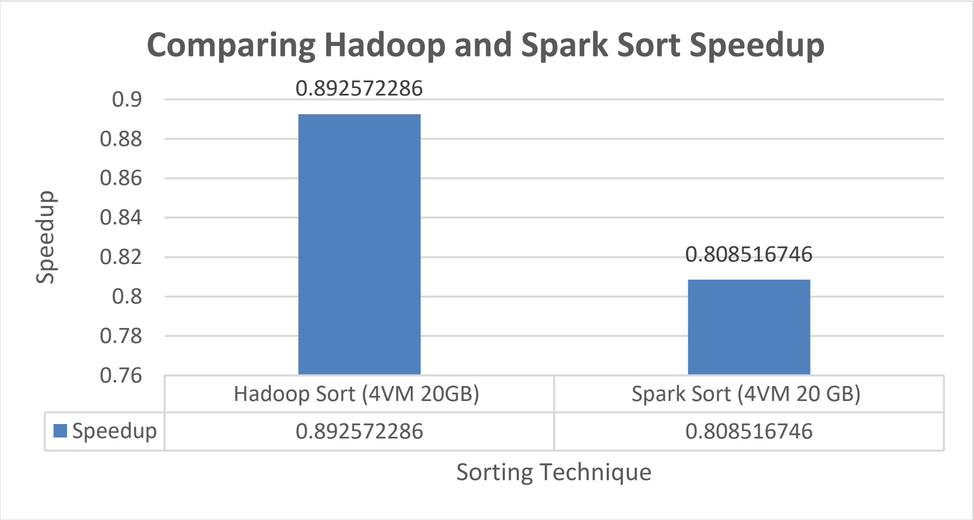


Chart 7 : Comparing Hadoop and Spark Sort Speedup

- The above chart compares speedup for Hadoop and Spark sort(4VM 20 GB). From the graph shown Hadoop sort has higher speedup than the spark sort

Analysis 4 : Efficiency Comparison : Hadoop vs Spark Sort

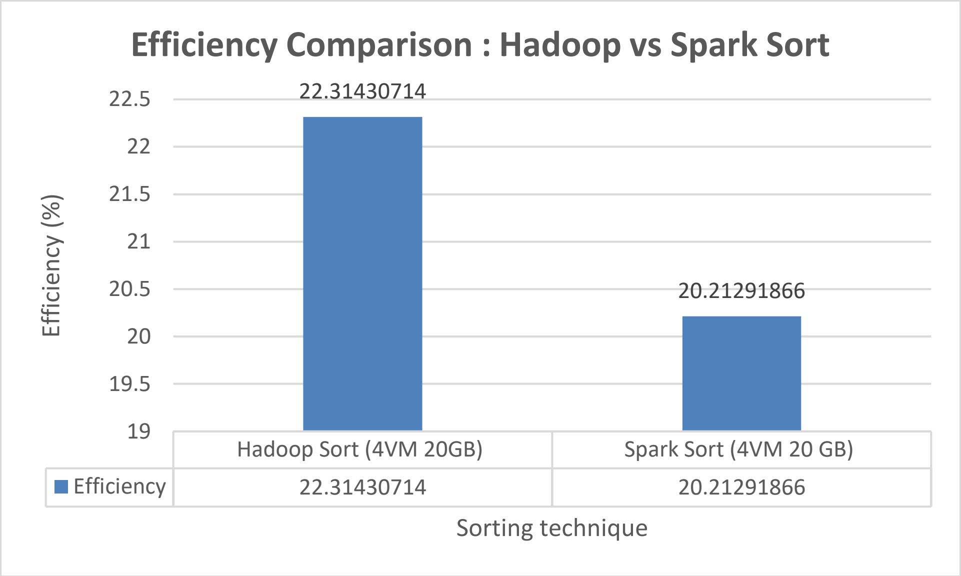


Chart 8 : Efficiency Comparison : Hadoop vs Spark Sort

- The above chart compares the efficiency (%) for Hadoop and Spark sort(4VM 20 GB). From the graph shown Hadoop sort has higher efficiency than the spark sort

WEAK SCALING LARGE DATASET

Analysis 1 : Comparison : Computation Time (sec)

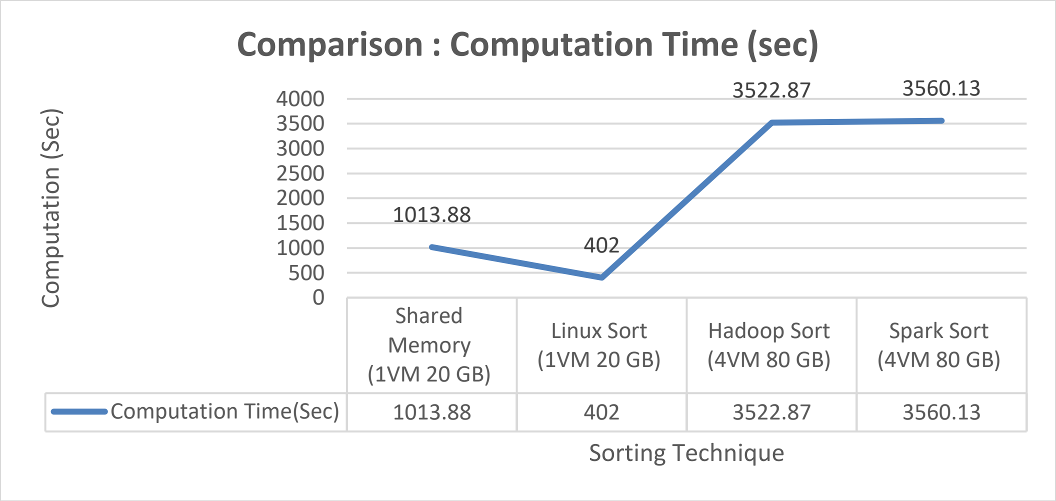


Chart 9 : Comparison : Computation Time (sec)

- The above chart compares the computation time taken in seconds using various sorting techniques and with different no. of VMs. I observed that Linux sort took the least time to sort 20 GB data on 1 VM and then shared memory followed by Hadoop sort and lastly spark sort

Analysis 2 : I/O Throughput (MB/sec) Comparison

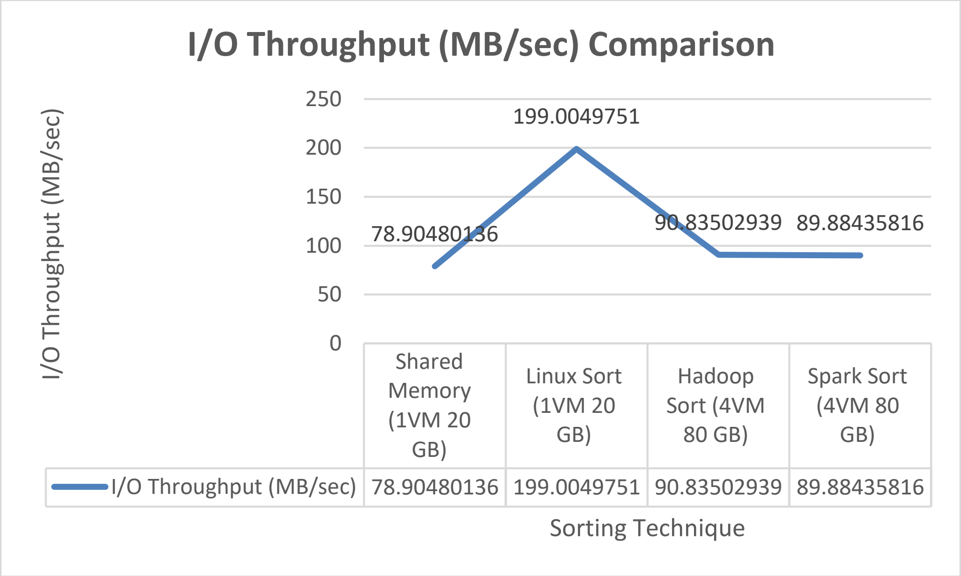


Chart 10 : I/O Throughput (MB/sec) Comparison

- The above chart compares the I/O throughput (MB/sec) using various sorting techniques and with different no. of VMs. I observed that Linux sort had the highest throughput as it implements efficient sorting algorithm and then followed by shared memory then Hadoop sort and then spark sort

Analysis 3 : Comparing Hadoop and Spark Sort Speedup

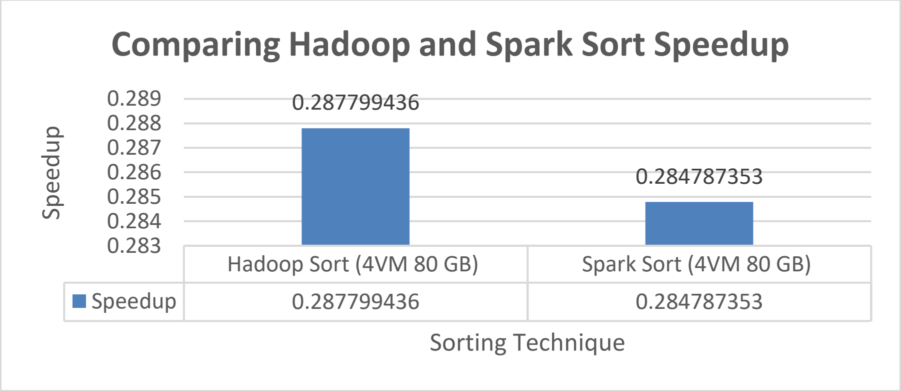


Chart 11 : Comparing Hadoop and Spark Sort Speedup

- The above chart compares speedup for Hadoop and Spark sort(4VM 80 GB). From the graph shown Hadoop sort has higher speedup than the spark sort

Analysis 4 : Efficiency Comparison : Hadoop vs Spark Sort

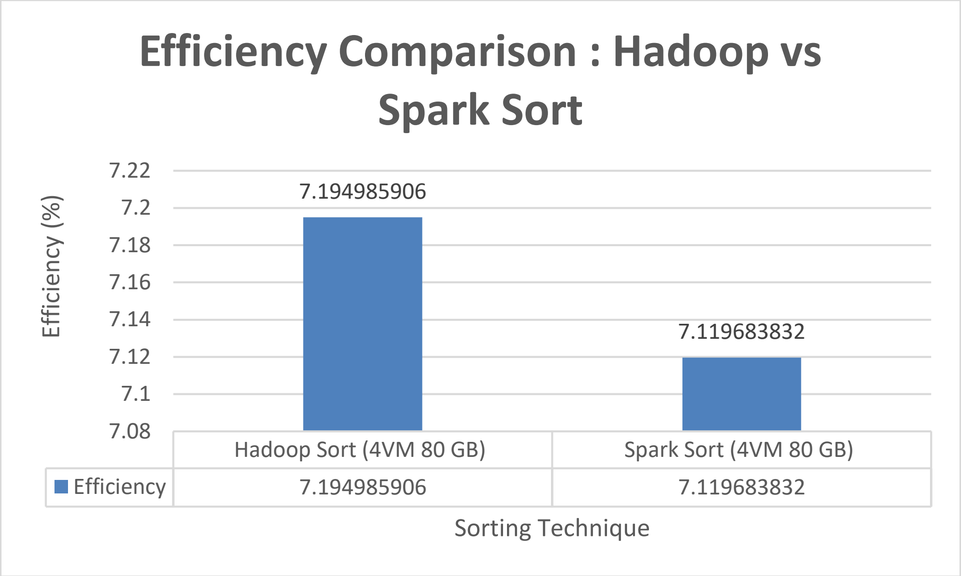


Chart 12 : Efficiency Comparison : Hadoop vs Spark Sort

- The above chart compares the efficiency (%) for Hadoop and Spark sort(4VM 80 GB). From the graph shown Hadoop sort has higher efficiency than the spark sort

CONCLUSION :

- Performed experiments using Hadoop and spark
- Analyzed the concept of weak and strong scaling
- Calculated the speedup and efficiency and also the I/O throughput for all the experiments asked to perform
- From all the experiments performed above multiple times and according to the results I have obtained for one and 4 nodes hadoop is better
- For 100 and thousand nodes spark will give better performance since spark is build on Hadoop
- The sort benchmark given in the assignment can sort 80 GB of data in very less amount of time