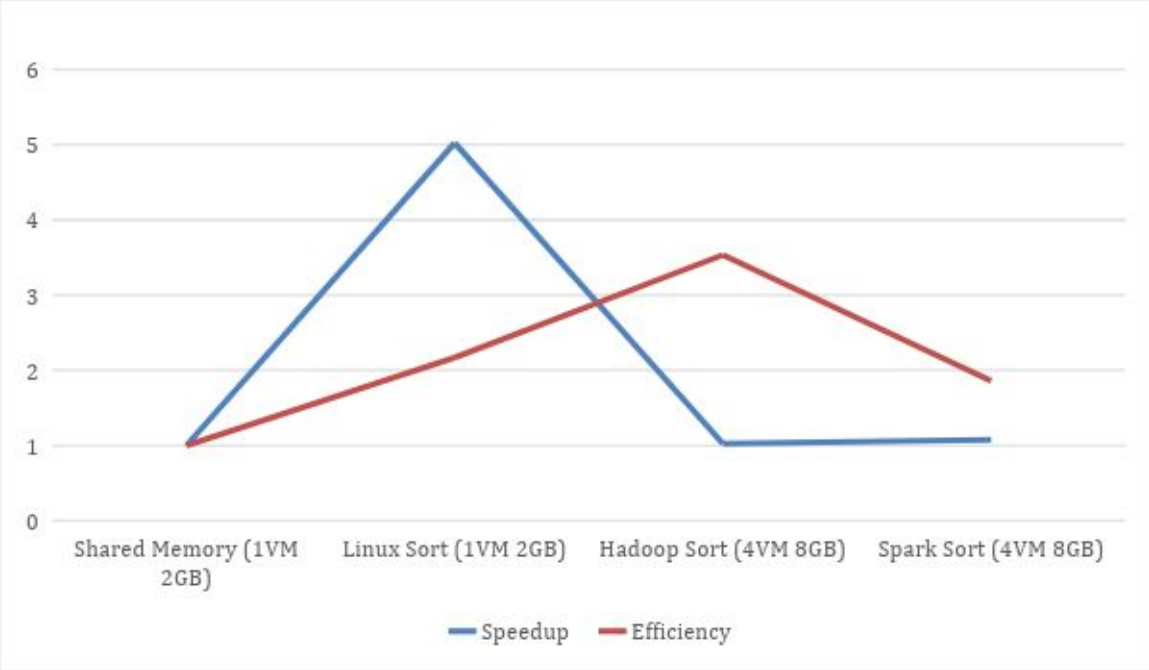
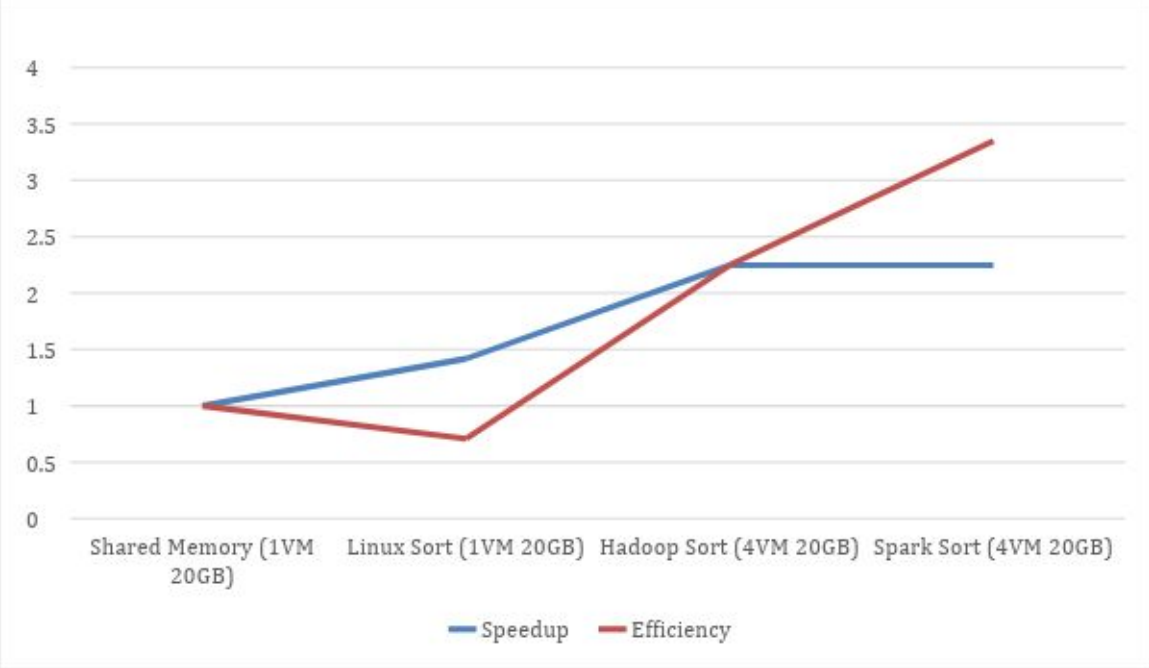
Evaluations:

Table 1 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)	120.56	24	471	448
Data Read (GB)	2	2	16	8
Data Write (GB)	2	2	16	8
I/O throughput (MB/sec)	76.92307692	166.6666667	67.94055202	35.71428571
Speedup	1x	5.023333333	1.023864119	1.076428571
Efficiency	1x	2.166666667	3.532908705	1.857142857



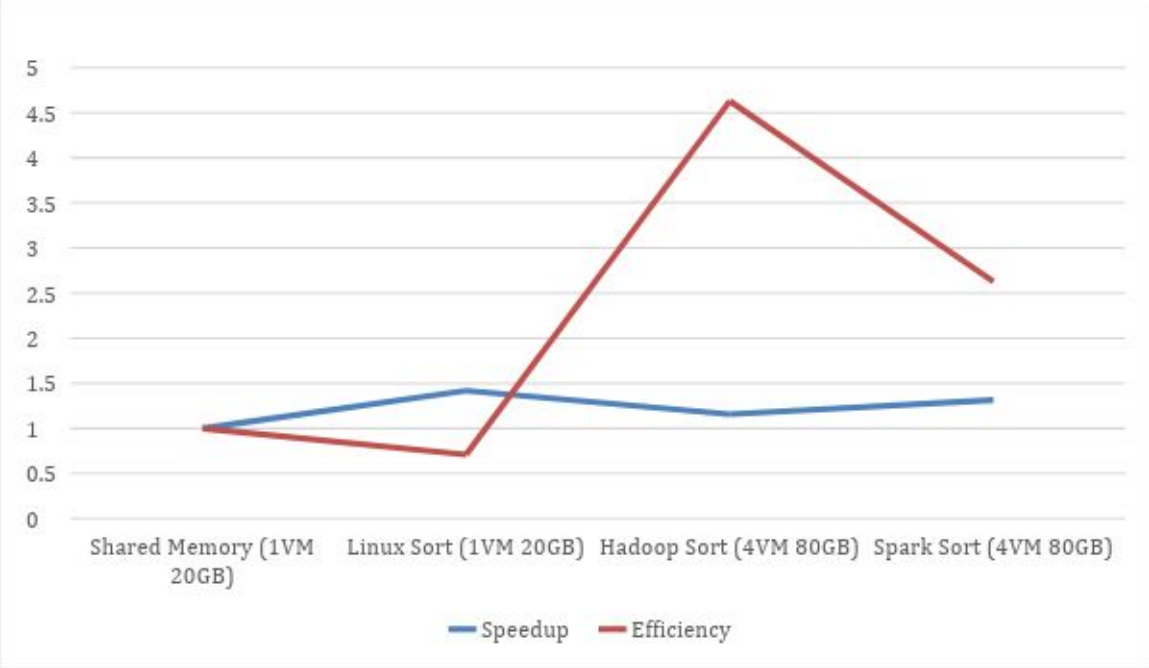
For sorting smaller chunk of data Shared Memory performs better than Hadoop and Spark sorting techniques.

Table 2 Evaluation of Sort (strong scaling - large dataset)				
Experiment	Shared Memory (1VM 20GB)	Linux Sort (1VM 20GB)	Hadoop Sort (4VM 20GB)	Spark Sort (4VM 20GB)
Computation Time (sec)	728.92	514	1297	435.45
Data Read (GB)	40	20	40	20
Data Write (GB)	40	20	40	20
I/O throughput (MB/sec)	109.75	77.82	61.68	91.85899644
Speedup	1x	1.418132296	2.248018504	2.248018504
Efficiency	1x	0.7090660592	2.248018223	3.347936089



For 20GB of data Spark sort is a lot more efficient that Hadoop Sort. SHared memory does not perform very well compared to both of the techniques.

Table 3 Evaluation of Sort (weak scaling - large dataset)				
Experiment	Shared Memory (1VM 20GB)	Linux Sort (1VM 20GB)	Hadoop Sort (4VM 80GB)	Spark Sort (4VM 80GB)
Computation Time (sec)	728.92	514	2520	2220
Data Read (GB)	40	20	160	80
Data Write (GB)	40	20	160	80
I/O throughput (MB/sec)	109.75	77.82	126.984127	72.07207207
Speedup	1x	1.418132296	1.157015873	1.313369369
Efficiency	1x	0.7090660592	4.628123079	2.626772558



For 80GB data Spark performs better than hadoop in terms of speedup.

Questions and Answers:

Question 1: What conclusions can you draw?
Answer: Infrastructure, the kind of application we want to build, data we are going to process and tasks to be performed on that data are the major factors in deciding what software stack will be used. For smaller amount of data we do not need to setup clusters so we can run the SharedMemory implementation which can be run as a simple batch job. For large dataset if processing of intermediate data is required less frequently Hadoop can be used, where processing of intermediate data is required more frequently Spark can be used.

Question 2: Which seems to be best at 1 node scale?
Answer: For 1 node writing own implementation of external sort(SharedMemory) seems to be a better option as we do not need to setup clusters or store files in HDFS. The code implementation required for 1 node is also shorter in case of writing own implementation.

Question 3: Can you predict which would be best at 100 node scale?
Answer: Spark. Because it has heterogeneous architecture and it is scalable. Also, spark provides easy to use API for python, scala and java. As our application is 100 node scale all these features of spark would be advantageous.

Question 4: How about 1000 node scale?
Answer: Hadoop and Spark. As the number of nodes is quite high we can use combination of clusters of both spark and hadoop which can be expanded as needed. Both hadoop and Spark have various advantages which can be combined to make the application perform better.

Question 5: Compare your results with those from the Sort Benchmark , specifically the winners in 2013 and 2014 who used Hadoop and Spark. Also, what can you learn from the CloudSort benchmark.
Answer: Benchmark application: 330 EC2 instances and 100TB data, so data sorted on every node =0.33 TB. SharedMemory application is sorting 20GB data a single node. CloudSort benchmark is a lot more efficient as it also is running on a network of nodes and providing much better results. For SharedMemory the performance could be enhanced by following sorting and threading techniques used in CloudSort benchmark.