Neha Maddali

**PART 1**

1. No, a NameNode machine should not be on the same machine as a DataNode. This is because they have different performance characteristics and hardware requirements. DataNode requires more disk space and network bandwidth for storage of data blocks. NameNode needs more memory and CPU power to handle metadata. If NameNode and DataNode were to be on the same machine, there may be a chance of performance degradation and resource contention.
2. No. A secondary node is not a substitute/back up node for the NameNode. A secondary node performs checkpoints at intervals of the namespace. If there is a NameNode failure, the checkpoint can be used to restore the metadata to a new NameNode. A secondary node doesn’t have the ability to take the responsibilities of the NameNode, hence not a substitute/back up node.
3. One advantage of YARN over Hadoop is its ability to support multiple applications and processing frameworks concurrently. YARN allows different data processing engines to coexist and share the same cluster resources. This makes it more efficient in resource utilization and gives more flexibility in deploying and managing different workloads.
4. A heartbeat is a signal sent by a DataNode to the NameNode to indicate that the DataNode is still functioning properly and alive. The message has information like the DataNode’s health status, storage space, and new data blocks created since the last heartbeat. Heartbeats are sent regularly to make sure that the NameNode has up to date information about the state of the cluster.
5. Yes, is it possible to change the block size of HDFS files and the immediate consequence would be a change in the size of the data blocks in which the file is divided which can affect the performance of the HDFS cluster. If the block size is increased, each block will contain more data and the number of blocks required to store the file will decrease and vice versa.

**PART 2**

1. Time = 500 GB / 256 MB/sec = 512,000 MB / 256 MB/sec = 2,000 sec
2. In a best case scenario, distribute blocks among nodes having one block per node. This way each node can read its own block without a network transfer.

500 GB / 128 MB = 4,000 nodes.

Time for one block reading = 128 MB / 256MB/sec = 0.5 sec

With respect to part a, the speed up will be = 2,000 sec / 0.5 sec = 4,000 times

1. Since there is only 100 nodes, each node would have to store about 4,000/100 = 40 blocks on average. We know that the time to read one block is 0.5 seconds. The time to read all blocks on one node is going to be approximately 39\*0.5 = 19.5 sec which is also the time to read the whole file in parallel.

**PART 3**

Input: two n x n square matrices (matrix A and matrix B), represented by input files in HDFS.

Mapper:

read the input files A and B in (K,V) key-value pairs. K is the line number. V is the matrix row from the csv format.

for each (K,V) pair from matrix A

emit intermediate pairs (j, (“A”, i, value)) where j is the column index and i is the row index. value is the matrix element

for each (K,V) pair from matrix B

emit intermediate pairs (i, (“B”, j, value)) just as done for matrix A

Reducer:

for each key K received by a reducer

multiple corresponding matrix elements of A and B to get C = A x B

emit final (K,V) pairs (i, (j, value)). i and j are same as specified in mapper. Value is the matrix

element in C

Per-Mapper CPU Cost: the mapper will read one line of input at a time. If the input file has n^2 lines, the total CPU cost is O(n^2 / M) where M is the mappers.

Per-Reducer CPU Cost: the reducer processes one key-value pair at a time. The number of pairs for each key will be the number of elements in matrix A’s row and the number of elements in matrix B’s column which is n^2. So the total CPU cost is O(n^2 / R) where R is reducers.

Communication cost: The number of elements in the A x B matrix is the number of pairs in total. That is n^2 and so the total communication cost is O(n^2).