**1) If 8TB is the available disk space per node (10 disks with 1 TB, 2 disk for operating system etc. were excluded.). Assuming initial data size is 600 TB. How will you estimate the number of data nodes (n)?**

Estimating the hardware requirement is always challenging in Hadoop environment because we never know when data storage demand can increase for a business. We must understand following factors in detail to come to a conclusion for the current scenario of adding right numbers to the cluster:

1. The actual size of data to store – 600 TB
2. At what pace the data will increase in the future (per day/week/month/quarter/year) – Data trending analysis or business requirement justification (prediction)
3. We are in Hadoop world, so replication factor plays an important role – default 3x replicas
4. Hardware machine overhead (OS, logs etc.) – 2 disks were considered
5. Intermediate mapper and reducer data output on hard disk - 1x
6. Space utilization between 60 % to 70 % - Finally, as a perfect designer we never want our hard drive to be full with their storage capacity.
7. Compression ratio

Let’s do some calculation to find the number of data nodes required to store 600 TB of data:

**Rough calculation:**

* Data Size – 600 TB
* Replication factor – 3
* Intermediate data – 1
* Total Storage requirement – (3+1) \* 600 = 2400 TB
* Available disk size for storage – 8 TB
* Total number of required data nodes (approx.): 2400/8 = 300 machines

**Actual Calculation:** Rough Calculation + Disk space utilization + Compression ratio

* Disk space utilization – 65 % (differ business to business)
* Compression ratio – 2.3
* Total Storage requirement – 2400/2.3 = 1043.5 TB
* Available disk size for storage – 8\*0.65 = 5.2 TB
* Total number of required data nodes (approx.): 1043.5/5.2 = 201 machines
* Actual usable cluster size (100 %): (201\*8\*2.3)/4 = 925 TB

**Case:** Business has predicted 20 % data increase in a quarter and we need to predict the new machines to be added in a year

* Data increase – 20 % over a quarter
* Additional data:
* 1st quarter: 1043.5 \* 0.2 = 208.7 TB
* 2nd quarter: 1043.5 \* 1.2 \* 0.2 = 250.44 TB
* 3rd quarter: 1043.5 \* (1.2)^2 \* 0.2 = 300.5 TB
* 4th quarter: 1043.5 \* (1.2)^3 \* 0.2 = 360.6 TB
* Additional data nodes requirement (approx.):
* 1st quarter: 208.7/5.2 = 41 machines
* 2nd quarter: 250.44/5.2 = 49 machines
* 3rd quarter: 300.5/5.2 = 58 machines
* 4th quarter: 360.6/5.2 = 70 machines

With these numbers you can predict next year additional machines requirement for the cluster (last quarter + 24), (last quarter + 28) and so on.

**2) You have a directory DeZyre that has the following files – HadoopTraining.txt, \_SparkTraining.txt, #DataScienceTraining.txt, .SalesforceTraining.txt. If you pass the DeZyre directory to the Hadoop MapReduce jobs, how many files are likely to be processed?**

Only HadoopTraining.txt and #DataScienceTraining.txt will be processed for Mapreduce jobs because when we process a file (either in a directory or individual) in Hadoop using any FileInputFormat such as TextInputFormat, KeyValueInputFormat or SequenceFileInputFormat, we must confirm that none of files must have a hidden file prefix such as “\_” or “.” because mapreduce FileInputFormat will by default uses hiddenFileFilter class to ignore all those files with these prefix in their name.

  private static final PathFilter hiddenFileFilter = new PathFilter(){

      public boolean accept(Path p){

        String name = p.getName();

        return !name.startsWith("\_") && !name.startsWith(".");

      }

    };

However, we can set our own custom filter such as FileInputFormat.setInputPathFilter to eliminate such criteria but remember, hiddenFileFilter is always active.

**3) Imagine that you are uploading a file of 500MB into HDFS.100MB of data is successfully uploaded into HDFS and another client wants to read the uploaded data while the upload is still in progress. What will happen in such a scenario, will the 100 MB of data that is uploaded will it be displayed?**

Although the default blocks size is 64 MB in Hadoop 1x and 128 MB in Hadoop 2x whereas in such a scenario let us consider block size to be 100 MB which means that we are going to have 5 blocks replicated 3 times (default replication factor). Let’s consider an example of how does a block is written to HDFS:

We have 5 blocks (A/B/C/D/E) for a file, a client, a namenode and a datanode. So, first the client will take Block A and will approach namenode for datanode location to store this block and the replicated copies. Once client is aware about the datanode information, it will directly reach out to datanode and start copying Block A which will be simultaneously replicated to other 2 datanodes. Once the block is copied and replicated to the datanodes, client will get the confirmation about the Block A storage and then, it will initiate the same process for next block “Block B”.

So, during this process if 1st block of 100 MB is written to HDFS and the next block has been started by the client to store then 1st block will be visible to readers. Only the current block being written will not be visible by the readers.

**4) When decommissioning the nodes in a Hadoop Cluster, why should you stop all the task trackers?**

We are aware about a complete process on how to decommission a datanode and there are loads of material available on internet to do so but what about the task tracker running a MapReduce job on a datanode which is likely to be decommissioned. Unlike the datanode, there is no graceful way to decommission a tasktracker. It is always assumed that when we want to move the same task to other node then we need to rely on making the task process to stop for failure and let it be rescheduled elsewhere on the cluster. It is possible that a task on its final attempt is running on the tasktracker and that a final failure may result in the entire job failing. Unfortunately, it’s not always possible to prevent this case from occurring. So, the idea behind decommissioning that it will stop your datanode but to move the current task to another node, we need to manually stop the task tracker running on the decommissioned node.

**5) When does a NameNode enter the safe mode?**

Namenode is responsible for managing the meta storage of the cluster and if something is missing from the cluster then Namenode will be held. This makes Namenode checking all the necessary information during the safe mode before making cluster writable to the users. There are couple of reasons for Namenode to enter the safe mode during startup such as;

i) Namenode loads the filesystem state from fsimage and edits log file, it then waits for datanodes to report their blocks, so it does not start replicating the blocks which already exist in the cluster another.

6) Did you ever run a lopsided job that resulted in out of memory ever? If yes, then how did you handle it?

7) There are 100 map tasks that are running, of which 99 tasks have completed and one task is running very slow. The slow running map task is replicated on a different machine and the output is gathered from the first completed map task. All other map tasks are killed. What is this phenomenon referred to in Hadoop?

8) There is an external jar file of size 1.5 MB having all the required dependencies to run your Hadoop MapReduce jobs. How will copy the jar file to the task tracker and what are the steps to follow?

9) If there are ‘m’ mappers and ‘r’ reducers in a given hadoop mapreduce job, how many copy and write operations will be required for the shuffle and sort algorithm? M\*N

10) When a job is run the properties file is copied to the distributed cache for the map jobs to access. How can you access the properties file?

11) How will you calculate the size of your hadoop cluster?

1. This is a formula to estimate Hadoop storage (H):

H=c\*r\*S/(1-i)

Where :

c = average compression ratio. It depends on the type of compression used (Snappy, LZOP, ...) and size of the data. When no compression is used, c=1.

r = replication factor. It is usually 3 in a production cluster.

S = size of data to be moved to Hadoop. This could be a combination of historical data and incremental data. The incremental data can be daily for example and projected over a period of time (3 years for example).

i = intermediate factor. It is usually 1/3 or 1/4. Hadoop's working space dedicated to storing intermediate results of Map phases.

Example: With no compression i.e. c=1, a replication factor of 3, an intermediate factor of .25=1/4

H= 1\*3\*S/(1-1/4)=3\*S/(3/4)=4\*S

With the assumptions above, the Hadoop storage is estimated to be 4 times the size of the initial data size.

2. This is the formula to estimate the number of data nodes (n):

n= H/d = c\*r\*S/(1-i)\*d

where d= disk space available per node. All other parameters remain the same as in 1.

Example: If 8TB is the available disk space per node (10 disks with 1 TB , 2 disk for operating system etc were excluded.). Assuming initial data size is 600 TB. n= 600/8=75 data nodes needed

12) How will you estimate the Hadoop storage given the size of the data to be moved, average compression ratio, intermediate and replication factors