Hadoop

### What are the differences between regular Filesystem and HDFS?

🡪In regular Filesystem, data is maintained in a single system. If the machine crashes, data recovery is challenging due to low fault tolerance. Seek time is more and hence it takes more time to process the data.

🡪In HDFS, Data is distributed and maintained on multiple systems. If a Data Node crashes, data can still be recovered from other nodes in the cluster. Time taken to read data is comparatively more, as there is local data read to the disc and coordination of data from multiple systems.

### What is HDFS fault-tolerant?

🡪 The primary mechanism through which HDFS achieves fault tolerance is through the replication of data blocks. When a file is stored in HDFS, it is split into blocks (default size is 128 MB), and each block is stored on multiple nodes across the cluster. The default replication factor is three, meaning each block has three copies stored on different nodes. This replication ensures that if one node fails, the data can still be accessed from another node that has a copy of the same block.

### What are the two types of metadata that a NameNode server holds

🡪 In HDFS (Hadoop Distributed File System), the NameNode is critical for managing the filesystem and stores two primary types of metadata. The first is \*\*Filesystem Metadata\*\*, which includes the directory tree, file names, and the permissions and properties of each file and directory, essentially organizing the structure of all files and directories within HDFS. The second type is \*\*Block Metadata\*\*, which contains detailed information about the data blocks that comprise each file, such as block IDs, their locations on specific DataNodes, block sizes, and replication levels. This metadata enables the NameNode to efficiently manage the filesystem, ensuring quick data retrieval and maintaining data reliability across the distributed environment.

### If you have an input file of 350 MB, how many blocks would HDFS create and what would be the size of each block? Assuming default block size is 128MB.

🡪When storing a file in HDFS, the file is split into blocks, and each block is stored independently. If the size of input file is 350 and default block size of HDFS is 128MB, the number of blocks required for a file of 350 MB is 350/128=2.734. there cannot be fraction of a block. Hence, 2 full blocks of 128MB each and 1 block of 94MB will be created.

### How does rack awareness work in HDFS?

🡪 Rack awareness is a strategy used in HDFS to improve data reliability, availability, and network traffic optimization across the cluster. When HDFS stores data, it considers the rack configuration of the cluster, which is the physical grouping of nodes in data centers. By knowing the "rack" where each node resides, HDFS strategically places replicas of data blocks across different racks. Typically, when a block of data is written to HDFS, the first replica is placed on a node in one rack, the second replica on a different node in the same rack, and the third replica on a node in a different rack. This placement policy helps in ensuring that even if an entire rack fails due to power, network, or hardware issues, at least one replica of the data remains accessible on a different rack, thereby enhancing fault tolerance. Additionally, by distributing the data across different racks, HDFS optimizes network traffic during data processing by balancing loads and reducing inter-rack data transfers, which are generally slower and more costly than intra-rack transfers.

### Is it possible to change the number of mappers to be created in a MapReduce job?

🡪 In a MapReduce job, it is not possible to directly change the number of mappers, but it can be changed by adjusting how the data is split before processing. Essentially, mappers are determined by the number of input splits into which data is divided. For example, if the data is stored in a file of 256 MB and you set the block size to 128 MB, then the job would typically use two mappers. Smaller splits increase the number of mappers, which can be useful for quicker processing of small tasks, while larger splits reduce the number of mappers, which may be beneficial for processing large, complex tasks more efficiently.

### What is the distributed cache in MapReduce?

🡪 The distributed cache in MapReduce is a Hadoop framework feature used to cache files needed by applications. Once cached, Hadoop makes these files available on each node across the cluster where map/reduce tasks are running, providing significant performance benefits. Application developers can use this feature to share read-only data/input files (such as dictionaries, configuration files, or data subsets) efficiently with all nodes rather than copying these files into HDFS or making a fresh copy with each task. This helps prevent redundant data transfers across the network, saving bandwidth and decreasing the time required to start processing. When setting up a job, developers specify files to be cached via job configuration. The framework then automatically copies the specified files to the local node before executing any tasks (map/reduce), thus making file access faster and more efficient for the tasks.