Hadoop Admin Interview Questions

1. **What are the main components of a Hadoop Application?**

- HDFS: This is the file system in which Hadoop data is stored. It is a distributed file system with very high bandwidth.
- Hadoop Common_: This is common set of libraries and utilities used by Hadoop.
- Hadoop MapReduce: This is based on MapReduce algorithm for providing large-scale data processing.
- Hadoop YARN: This is used for resource management in a Hadoop cluster. It can also schedule tasks for users.

2. **What is the core concept behind Apache Hadoop framework?**
   Apache Hadoop is based on the concept of MapReduce algorithm. In MapReduce algorithm, Map and Reduce operations are used to process very large data set. In this concept, Map method does the filtering and sorting of data. Reduce method performs the summarizing of data. This is a concept from functional programming. The key points in this concept are scalability and fault tolerance. In Apache Hadoop these features are achieved by multi-threading and efficient implementation of MapReduce.
3. **What is Hadoop Streaming?**
   Hadoop distribution provides a Java utility called Hadoop Streaming. It is packaged in a jar file. With Hadoop Streaming, we can create and run Map Reduce jobs with an executable script. We can create executable scripts for Mapper and Reducer functions. These executable scripts are passed to Hadoop Streaming in a command. Hadoop Streaming utility creates Map and Reduce jobs and submits these to a cluster. We can also monitor these jobs with this utility.
4. **What is the difference between Nodes in HDFS?**
   The differences between NameNode, BackupNode and Checkpoint NameNode are as follows:

- NameNode: NameNode is at the heart of the HDFS file system that manages the metadata i.e. the data of the files is not stored on the NameNode but rather it has the directory tree of all the files present in the HDFS file system on a Hadoop cluster. NameNode uses two files for the namespace:
  - fsimage file: This file keeps track of the latest checkpoint of the namespace.
  - edits file: This is a log of changes made to the namespace since checkpoint.
- Checkpoint Node: Checkpoint Node keeps track of the latest checkpoint in a directory that has same structure as that of NameNode’s directory. Checkpoint node creates

checkpoints for the namespace at regular intervals by downloading the edits and fsimage file from the NameNode and merging it locally. The new image is then again updated back to the active NameNode.

- BackupNode: This node also provides check pointing functionality like that of the Checkpoint node but it also maintains its up-to-date in-memory copy of the file system namespace that is in sync with the active NameNode.

5. **What are the default port numbers on which Nodes run in Hadoop?**
   Default port numbers of Name Node, Job Tracker and Task Tracker are as follows: NameNode runs on port 50070 Task Tracker runs on port 50060 Job Tracker runs on port 50030
6. **What do you know about Block and Block scanner in HDFS?**
   A large file in HDFS is broken into multiple parts and each part is stored on a different Block. By default a Block is of 64 MB capacity in HDFS. Block Scanner is a program that every Data node in HDFS runs periodically to verify the checksum of every block stored on the data node. The purpose of a Block Scanner is to detect any data corruption errors on Data node.
7. **How will you disable a Block Scanner on HDFS DataNode?**
   In HDFS, there is a configuration dfs.datanode.scan.period.hours in hdfs-site.xml to set the number of hours interval at which Block Scanner should run. We can set dfs.datanode.scan.period.hours=0 to disable the Block Scanner. It means it will not run on HDFS DataNode.
8. **How does inter cluster data copying works in Hadoop?**
   In Hadoop, there is a utility called DistCP (Distributed Copy) to perform large inter/intra-cluster copying of data. This utility is also based on MapReduce. It creates Map tasks for files given as input. After every copy using DistCP, it is recommended to run crosschecks to confirm that there is no data corruption and copy is complete.
9. **How can we update a file at an arbitrary location in HDFS?**
   This is a trick question. In HDFS, it is not allowed to update a file at an arbitrary location. All the files are written in append only mode. It means all writes are done at the end of a file. So there is no possibility of updating the files at any random location.
10. **What is Replication factor in HDFS?**
    Replication factor in HDFS is the number of copies of a file in file system. A Hadoop application can specify the number of replicas of a file it wants HDFS to maintain. This information is stored in NameNode. We can set the replication factor in following ways: We can use Hadoop fs shell, to specify the replication factor for a file. Command as follows:

- $hadoop fs –setrep –w 5 /file_name In above command, replication factor of file_name file is set as 5. We can also use Hadoop fs shell, to specify the replication factor of all the files in a directory.

- $hadoop fs –setrep –w 2 /dir_name In above command, replication factor of all the files under directory dir_name is set as 2.

11. **What is the difference between NAS and DAS in Hadoop cluster?**
    NAS stands for Network Attached Storage and DAS stands for Direct Attached Storage. In NAS, compute and storage layers are separated. Storage is distributed over different servers on a network. In DAS, storage is attached to the node where computation takes place. Apache Hadoop is based on the principle of moving processing near the location of data. So it needs storage disk to be local to computation. With DAS, we get very good performance on a Hadoop cluster. Also DAS can be implemented on commodity hardware. So it is more cost effective. Only when we have very high bandwidth (around 10 GbE) it is preferable to use NAS storage.
12. **What are the two messages that NameNode receives from DataNode?**
    NameNode receives following two messages from every DataNode:

- Heartbeat: This message signals that DataNode is still alive. Periodic receipt of Heartbeat is vey important for NameNode to decide whether to use a DataNode or not.
- Block Report: This is a list of all the data blocks hosted on a DataNode. This report is also very useful for functioning of NameNode. With this report, NameNode gets information about what data is stored on a specific DataNode.

13. **How does indexing work in Hadoop?**
    Indexing in Hadoop has two different levels. Index based on File URI: In this case data is indexed based on different files. When we search for data, index will return the files that contain the data. Index based on InputSplit: In this case, data is indexed based on locations where input split is located.
14. **What data is stored in a HDFS NameNode?**
    NameNode is the central node of an HDFS system. It does not store any actual data on which MapReduce operations have to be done. But it has all the metadata about the data stored in HDFS DataNodes. NameNode has the directory tree of all the files in HDFS filesystem. Using this meta data it manages all the data stored in different DataNodes.
15. **What are the main functions of Secondary NameNode?**
    Main functions of Secondary NameNode are as follows:

- FsImage: It stores a copy of FsImage file and EditLog.
- NameNode crash: In case NameNode crashes, we can use Secondary NameNode's FsImage to recreate the NameNode.
- Checkpoint: Secondary NameNode runs Checkpoint to confirm that data is not corrupt in HDFS.
  Update: It periodically applies the updates from EditLog to FsImage file. In this way

FsImage file on Secondary NameNode is kept up to date. This helps in saving time during NameNode restart.

16. **What is the meaning of Rack Awareness in Hadoop?**
    In Hadoop, most of the components like NameNode, DataNode etc are rack- aware. It means they have the information about the rack on which they exist. The main use of rack awareness is in implementing fault-tolerance. Any communication between nodes on same rack is much faster than the communication between nodes on two different racks. In Hadoop, NameNode maintains information about rack of each DataNode. While reading/writing data, NameNode tries to choose the DataNodes that are closer to each other. Due to performance reasons, it is recommended to use close data nodes for any operation. So Rack Awareness is an important concept for high performance and fault-tolerance in Hadoop. If we set Replication factor 3 for a file, does it mean any computation will also take place 3 times? No. Replication factor of 3 means that there are 3 copies of a file. But computation takes place only one the one copy of file. If the node on which first copy exists, does not respond then computation will be done on second copy.
17. **Why do we use fsck command in HDFS?**
    fsck command is used for getting the details of files and directories in HDFS. Main uses of fsck command in HDFS are as follows:

- delete: We use this option to delete files in HDFS.
- move: This option is for moving corrupt files to lost/found.
- locations: This option prints all the locations of a block in HDFS.
- racks: This option gives the network topology of data-node locations.
- blocks: This option gives the report of blocks in HDFS.

18. **What are the core methods of a Reducer in Hadoop?**
    The main task of Reducer is to reduce a larger set of data that shares a key to a smaller set of data. In Hadoop, Reducer has following three core methods:

- setup(): At the start of a task, setup() method is called to configure various parameters for Reducer.
- reduce(): This is the main operation of Reducer. In reduce() method we define the task that has to be done for a set of values that share a key.
- cleanup(): Once reduce() task is done, we can use cleanup() to clean any intermediate data or temporary files.

19. **What are the primary phases of a Reducer in Hadoop?**
    In Hadoop, there are three primary phases of a Reducer:

- Shuffle: In this phase, Reducer copies the sorted output from each Mapper.

- Sort: In this phase, Hadoop framework sorts the input to Reducer by same key. It uses merge sort in this phase. Sometimes, shuffle and sort phases occur at the same time.
- Reduce: This is the phase in which output values associated with a key are reduced to give output result. Output from Reducer is not re-sorted.

20. **What is the use of Context object in Hadoop?**
    Hadoop uses Context object with Mapper to interact with rest of the system. Context object gets the configuration of the system and job in its constructor. We use Context object to pass the information in setup(), cleanup() and map() methods. This is an important object that makes the important information available during the map operations.
21. **How does partitioning work in Hadoop?**
    Partitioning is the phase between Map phase and Reduce phase in Hadoop workflow. Since partitioner gives output to Reducer, the number of partitions is same as the number of Reducers. Partitioner will partition the output from Map phase into distinct partitions by using a user-defined condition. Partitions can be like Hash based buckets. E.g. If we have to find the student with the maximum marks in each gender in each subject. We can first use Map function to map the keys with each gender. Once mapping is done, the result is passed to Partitioner. Partitioner will partition each row with gender on the basis of subject. For each subject there will be a different Reducer. Reducer will take input from each partition and find the student with the highest marks.
22. **What is a Combiner in Hadoop?**
    Combiner is an optional step between Map and Reduce. Combiner is also called Semi-Reducer. Combiner takes output from Map, creates Key-value pairs and passes these to Reducer. Combiner's task is to summarize the outputs from Map into summary records with same key. By using Combiner, we can reduce the data transfer between Mapper and Reducer. Combiner does the task similar to reduce but it is done on the Map machine itself.
23. **Why does HDFS store data in Block structure?**
    HDFS stores all the data in terms of Blocks. With Block structure there are some benefits that HDFS gets. Some of these are as follows: Fault Tolerance: With Block structure, HDFS implements replication. By replicating same block in multiple locations, fault tolerance of the system increases. Even if some copy is not accessible, we can get the data from another copy. Large Files: We can store very large files that cannot be even stored one disk, in HDFS by using Block structure. We just divide the data of file in multiple Blocks. Each Block can be stored on same or different machines. Storage management: With Block storage it is easier for Hadoop nodes to calculate the data storage as well as perform optimization in the algorithm to minimize data transfer across the network.
24. **How will you create a custom Partitioner in a Hadoop job?**
    Partition phase runs between Map and Reduce phase. It is an optional phase. We can create a custom partitioner by extending the org.apache.hadoop.mapreduce.Partitio class

in Hadoop. In this class, we have to override getPartition(KEY key, VALUE value, int numPartitions) method. This method takes three inputs. In this method, numPartitions is same as the number of reducers in our job. We pass key and value to get the partition number to which this key,value record will be assigned. There will be a reducer corresponding to that partition. The reducer will further handle to summarizing of the data.Once custom Partitioner class is ready, we have to set it in the Hadoop job. We can use following method to set it: job.setPartitionerClass(CustomPartitioner)

25. **What is a Checkpoint node in HDFS?**
    A Checkpoint node in HDFS periodically fetches fsimage and edits from NameNode, and merges them. This merge result is called a Checkpoint. Once a Checkpoint is created, Checkpoint Node uploads the Checkpoint to NameNode. Secondary node also takes Checkpoint similar to Checkpoint Node. But it does not upload the Checkpoint to NameNode. Main benefit of Checkpoint Node is in case of any failure on NameNode. A NameNode does not merge its edits to fsimage automatically during the runtime. If we have long running task, the edits will become huge. When we restart NameNode, it will take much longer time, because it will first merge the edits. In such a scenario, Checkpoint node helps for a long running task. Checkpoint nodes performs the task of merging the edits with fsimage and then uploads these to NameNode. This saves time during the restart of NameNode.
26. **What is a Backup Node in HDFS?**
    Backup Node in HDFS is similar to Checkpoint Node. It takes the stream of edits from NameNode. It keeps these edits in memory and also writes these to storage to create a new checkpoint. At any point of time, Backup Node is in sync with the Name Node. The difference between Checkpoint Node and Backup Node is that Backup Node does not upload any checkpoints to Name Node. Also Backup node takes a stream instead of periodic reading of edits from Name Node.
27. **What is the meaning of term Data Locality in Hadoop?**
    In a Big Data system, the size of data is huge. So it does not make sense to move data across the network. In such a scenario, Hadoop tries to move computation closer to data. So the Data remains local to the location wherever it was stored. But the computation tasks will be moved to data nodes that hold the data locally. Hadoop follows following rules for Data Locality optimization:

    - Hadoop first tries to schedule the task on node that has an HDFS file on a local disk. If it cannot be done, then Hadoop will try to schedule the task on a node on the same rack as the node that has data. If this also cannot be done, Hadoop will schedule the task on the node with same data on a different rack. The above method works well, when we work with the default replication factor of 3 in Hadoop.

28. **What is a Balancer in HDFS?**
    In HDFS, data is stored in blocks on a DataNode. There can be a situation when data is

not uniformly spread into blocks on a DataNode. When we add a new DataNode to a cluster, we can face such a situation. In such a case, HDFS provides a useful tool Balancer to analyze the placement of blocks on a DataNode. Some people call it as Rebalancer also. This is an administrative tool used by admin staff. We can use this tool to spread the blocks in a uniform manner on a DataNode.

29. **What are the important points a NameNode considers before selecting the DataNode for placing a data block?**
    Some of the important points for selecting a DataNode by NameNode are as follows: NameNode tries to keep at least one replica of a Block on the same node that is writing the block. It tries to spread the different replicas of same block on different racks, so that in case of one rack failure, other rack has the data. One replica will be kept on a node on the same node as the one that it writing it. It is different from point 1. In Point 1, block is written to same node. In this point block is written on a different node on same rack. This is important for minimizing the network I/O. NameNode also tries to spread the blocks uniformly among all the DataNodes in a cluster.
30. **What is Safemode in HDFS?**
    Safemode is considered as the read-only mode of NameNode in a cluster. During the startup of NameNode, it is in SafeMode. It does not allow writing to file-system in Safemode. At this time, it collects data and statistics from all the DataNodes. Once it has all the data on blocks, it leaves Safemode. The main reason for Safemode is to avoid the situation when NameNode starts replicating data in DataNodes before collecting all the information from DataNodes. It may erroneously assume that a block is not replicated well enough, whereas, the issue is that NameNode does not know about whereabouts of all the replicas of a block. Therefore, in Safemode, NameNode first collects the information about how many replicas exist in cluster and then tries to create replicas wherever the number of replicas is less than the policy.
31. **How will you replace HDFS data volume before shutting down a DataNode?**
    In HDFS, DataNode supports hot swappable drives. With a swappable drive we can add or replace HDFS data volumes while the DataNode is still running. The procedure for replacing a hot swappable drive is as follows: First we format and mount the new drive. We update the DataNode configuration dfs.datanode.data.dir to reflect the data volume directories. Run the "dfsadmin -reconfig datanode HOST:PORT start" command to start the reconfiguration process Once the reconfiguration is complete, we just unmount the old data volume After unmount we can physically remove the old disks.
32. **What are the important configuration files in Hadoop?**
    There are two important configuration files in a Hadoop cluster:

- Default Configuration: There are core-default.xml, hdfs-default.xml and mapred-default.xml files in which we specify the default configuration for Hadoop cluster. These are read only files.

- Custom Configuration: We have site-specific custom files like core-site.xml, hdfs-site.xml, mapred-site.xml in which we can specify the site-specific configuration.
  All the Jobs in Hadoop and HDFS implementation uses the parameters defined in the above-mentioned files. With customization we can tune these processes according to our use case. In Hadoop API, there is a Configuration class that loads these files and provides the values at run time to different jobs.

33. **Why do we need Serialization in Hadoop map reduce methods?**
    In Hadoop, there are multiple data nodes that hold data. During the processing of map and reduce methods data may transfer from one node to another node. Hadoop uses serialization to convert the data from Object structure to Binary format. With serialization, data can be converted to binary format and with de-serialization data can be converted back to Object format with reliability.
34. **What is the use of Distributed Cache in Hadoop?**
    Hadoop provides a utility called Distributed Cache to improve the performance of jobs by caching the files used by applications. An application can specify which file it wants to cache by using JobConf configuration. Hadoop framework copies these files to the nodes one which a task has to be executed. This is done before the start of execution of a task. DistributedCache supports distribution of simple read only text files as well as complex files like jars, zips etc.
35. **How will you synchronize the changes made to a file in Distributed Cache in Hadoop?**
    It is a trick question. In Distributed Cache, it is not allowed to make any changes to a file. This is a mechanism to cache read-only data across multiple nodes.Therefore, it is not possible to update a cached file or run any synchronization in Distributed Cache.
36. **Where Mapreduce not recommended?**
    Mapreduce is not recommended for Iterative kind of processing. It means repeat the output in a loop manner.To process Series of Mapreduce jobs, MapReduce not suitable. each job persists data in local disk, then again load to another job. It’s costly operation and not recommended.
37. **What is Namenode and it’s responsibilities?**
    Namenode is a logical daemon name for a particular node. It’s heart of the entire Hadoop system. Which store the metadata in FsImage and get all block information in the form of Heartbeat.
38. **What is Jobtracker’s responsibility?**
    Scheduling the job’s tasks on the slaves. Slaves execute the tasks as directed by the JobTracker. Monitoring the tasks, if failed, reexecute the failed tasks.
39. **What are the Jobtracker and Tasktracker?**
    MapReduce Framework consists of a single Job Tracker per Cluster, one Task Tracker per node. Usually A cluster has multiple nodes, so each cluster has single Job Tracker and

multiple TaskTrackers.JobTracker can schedule the job and monitor the Task Trackers. If Task Tracker failed to execute tasks, try to re-execute the failed tasks. TaskTracker follow the JobTracker's instructions and execute the tasks. As a slave node, it report the job status to Master JobTracker in the form of Heartbeat.

40. **What is Job scheduling importance in Hadoop Mapreduce?**
    Scheduling is a systematic procedure of allocating resources in the best possible way among multiple tasks. Hadoop task tracker performing many procedures, sometime a particular procedure should finish quickly and provide more prioriety, to do it few job schedulers come into the picture. Default Schedule is FIFO. Fair scheduling, FIFO and CapacityScheduler are most popular hadoop scheduling in hadoop.
41. **When used Reducer?**
    To combine multiple mapper's output used reducer. Reducer has 3 primary phases sort, shuffle and reduce. It's possible to process data without reducer, but used when the shuffle and sort is required.
42. **Where the Shuffle and Sort process does?**
    After Mapper generate the output temporary store the intermediate data on the local File System. Usually this temporary file configured at coresite.xml in the Hadoop file. Hadoop Framework aggregate and sort this intermediate data, then update into Hadoop to be processed by the Reduce function. The Framework deletes this temporary data in the local system after Hadoop completes the job.
43. **What methods can controle the Map And Reduce function's output?**
    setOutputKeyClass() and setOutputValueClass() If they are different, then the map output type can be set using the methods. setMapOutputKeyClass() and setMapOutputValueClass()
44. **What is the main difference between Mapper And Reducer?**
    Map method is called separately for each key/value have been processed. It process input key/value pairs and emits intermediate key/value pairs. Reduce method is called separately for each key/values list pair. It process intermediate key/value pairs and emits final key/value pairs. Both are initialize and called before any other method is called. Both don't have any parameters and no output.
45. **What is difference between mapside join and reduce side join?**
    Join multple tables in mapper side, called map side join. Please note mapside join should has strict format and sorted properly. If dataset is smaller tables, goes through reducer phrase. Data should partitioned properly. Join the multiple tables in reducer side called reduce side join. If you have large amount of data tables, planning to join both tables. One table is large amount of rows and columns, another one has few number of tables only, goes through Rreduce side join. It's the best way to join the multiple tables
46. **When we are goes to Combiner?**
    Mappers and reducers are independent they dont talk each other. When the functions that are commutative(a.b = b.a) and associative {a.(b.c) = (a.b).c} we goes to combiner to

optimize the mapreduce process. Many mapreduce jobs are limited by the bandwidth, so by default Hadoop framework minimizes the data bandwidth network wastage. To achieve it's goal, Mapreduce allows user defined "Cominer function" to run on the map output. It's an MapReduce optimization technique, but it's optional.

47. **What is the main difference between Mapreduce Combiner and Reducer?**
Both Combiner and Reducer are optional, but most frequently used in MapReduce. There are three main differences such as: combiner will get only one input from one Mapper. While Reducer will get multiple mappers from different mappers. If aggregation required used reducer, but if the function follows commutative (a.b=b.a) and associative a.(b.c)= (a.b).c law, use combiner. Input and output keys and values types must same in combiner, but reducer can follows any type input, any output format.
48. **What is partition?**
After combiner and intermediate mapoutput the Partitioner controls the keys after sort and shuffle. Partitioner divides the intermediate data according to the number of reducers so that all the data in a single partition gets executed by a single reducer. It means each partition can executed by only a single reducer. If you call reducer, automatically partition called in reducer by automatically.
49. **When we goes to Partition?**
By default Hive reads entire dataset even the application have a slice of data. It's a bottleneck for mapreduce jobs. So Hive allows special option called partitions. When you are creating table, hive partitioning the table based on requirement.
50. **What are the important steps when you are partitioning table?**
Don't over partition the data with too small partitions, it's overhead to the namenode. if dynamic partition, atleast one static partition should exist and set to strict mode by using given commands.

- SET hive.exec.dynamic.partition = true;
- SET hive.exec.dynamic.partition.mode = nonstrict;
first load data into nonpartitioned table, then load such data into partitioned table. It's not possible to load data from local to partitioned table. insert overwrite table table_name partition(year) select * from nonpartitiontable;

51. **Can you elaborate Mapreduce Job architecture?**
First Hadoop programmer submit Mpareduce program to JobClient. Job Client request the JobTracker to get Job id, Job tracker provide JobID, its's in the form of Job_HadoopStartedtime_00001. It's unique ID. Once JobClient receive received Job ID copy the Job resources (job.xml, job.jar) to File System (HDFS) and submit job to JobTracker. JobTracker initiate Job and schedule the job. Based on configuration, job split the input splits and submit to HDFS. TaskTracker retrive the job resources from HDFS and

launch Child JVM. In this Child JVM, run the map and reduce tasks and notify to the Job tracker the job status.

52. **Why Jobclient and Job Tracker submits job resources to file system?**
    Data locality. Move competition is cheaper than moving Data. So logic/ competition in Jar file and splits. So Where the data available, in File System Datanodes. So every resources copy where the data available.
53. **How to configure the split value?**
    By default block size = 64mb, but to process the data, job tracker split the data. Hadoop architect use these formulas to know split size.

- split size = min (max_splitsize, max (block_size, min_split_size));
- split size = max(min_split_size, min (block_size, max_split, size));
  by default split size = block size. Always No of splits = No of mappers. Apply above formula:
- split size = Min (max_splitsize, max (64, 512kB) // max _splitsize = depends on env, may 1gb or 10gb split size = min (10gb (let assume), 64) split size = 64MB.
- split size = max(min_split_size, min (block_size, max_split, size)); split size = max (512kb, min (64, 10GB)); split size = max (512kb, 64);split size = 64 MB;

54. **What is difference between block And split?**

- Block: How much chunk data to stored in the memory called block.
- Split: how much data to process the data called split.

55. **Why Hadoop Framework reads a file parallel why not sequential?**
    To retrieve data faster, Hadoop reads data parallel, the main reason it can access data faster. While, writes in sequence, but not parallel, the main reason it might result one node can be overwritten by other and where the second node. Parallel processing is independent, so there is no relation between two nodes, if writes data in parallel, it's not possible where the next chunk of data has. For example 100 MB data write parallel, 64 MB one block another block 36, if data writes parallel first block doesn't know where the remaining data. So Hadoop reads parallel and write sequentially.
56. **If I am change block size from 64 to 128?**
    Even you have changed block size not effect existent data. After changed the block size, every file chunked after 128 MB of block size. It means old data is in 64 MB chunks, but new data stored in 128 MB blocks.
57. **What is the data locality?**
    Whereever the data is there process the data, computation/process the data where the data available, this process called data locality. "Moving Computation is Cheaper than Moving Data" to achieve this goal follow data locality. It's possible when the data is splittable, by default it's true.

58. **What is speculative execution?**
    Hadoop run the process in commodity hardware, so it's possible to fail the systems also has low memory. So if system failed, process also failed, it's not recommendable.Speculative execution is a process performance optimization technique.Computation/logic distribute to the multiple systems and execute which system execute quickly. By default this value is true. Now even the system crashed, not a problem, framework choose logic from other systems.

- Eg: logic distributed on A, B, C, D systems, completed within a time.
  System A, System B, System C, System D systems executed 10 min, 8 mins, 9 mins 12 mins simultaneously. So consider system B and kill remaining system processes, framework take care to kill the other system process.

59. **Why Tasktracker launch child Jvm to do a task?**
    Sometime child threads currupt parent threads. It means because of programmer mistake entired MapReduce task distruped. So task tracker launch a child JVM to process individual mapper or tasker. If tasktracker use existent JVM, it might damage main JVM. If any bugs occur, tasktracker kill the child process and relaunch another child JVM to do the same task. Usually task tracker relaunch and retry the task 4 times.
60. **Explain how input and output data format of the Hadoop Framework?**
    The MapReduce framework operates exclusively on pairs, that is, the framework views the input to the job as a set of pairs and produces a set of pairs as the output of the job, conceivably of different types. See the flow mentioned below:

- (input) -> map -> -> combine/sorting -> -> reduce -> (output)

61. **How a task is scheduled by a Jobtracker?**
    The TaskTrackers send out heartbeat messages to the JobTracker, usually every few minutes, to reassure the JobTracker that it is still alive. These messages also inform the JobTracker of the number of available slots, so the JobTracker can stay up to date with where in the cluster work can be delegated. When the JobTracker tries to find somewhere to schedule a task within the MapReduce operations, it first looks for an empty slot on the same server that hosts the DataNode containing the data, and if not, it looks for an empty slot on a machine in the same rack.
62. **How does a NameNode handle the failure of the Data Nodes?**
    HDFS has master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on. The NameNode and DataNode are pieces of software designed to run on commodity machines. NameNode periodically receives a Heartbeat and a Block report from each of the DataNodes in the

cluster. Receipt of a Heartbeat implies that the DataNode is functioning properly. A Blockreport contains a list of all blocks on a DataNode. When NameNode notices that it has not received a heartbeat message from a data node after a certain amount of time, the data node is marked as dead. Since blocks will be under replicated the system begins replicating the blocks that were stored on the dead DataNode. The NameNode Orchestrates the replication of data blocks from one DataNode to another. The replication data transfer happens directly between DataNode and the data never passes through the NameNode.

63. **How Hdfs is different from traditional File Systems?**
HDFS, the Hadoop Distributed File System, is responsible for storing huge data on the cluster. This is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS is designed to support very large files. Applications that are compatible with HDFS are those that deal with large data sets. These applications write their data only once but they read it one or more times and require these reads to be satisfied at streaming speeds. HDFS supports write-once-read-many semantics on files.
64. **What is the basic difference between traditional Rdbms and Hadoop?**
Traditional RDBMS is used for transactional systems to report and archive the data, whereas Hadoop is an approach to store huge amount of data in the distributed file system and process it. RDBMS will be useful when you want to seek one record from Big data, whereas, Hadoop will be useful when you want Big data in one shot and perform analysis on that later.
65. **On what basis Data will be stored on a Rack?**
When the client is ready to load a file into the cluster, the content of the file will be divided into blocks. Now the client consults the Namenode and gets 3 datanodes for every block of the file which indicates where the block should be stored. While placing the datanodes, the key rule followed is “for every block of data, two copies will exist in one rack, third copy in a different rack“. This rule is known as “Replica Placement Policy“.
66. **Which are the two types of writes In Hdfs?**
There are two types of writes in HDFS:

- Posted and non-posted write. Posted Write is when we write it and forget about it, without worrying about the acknowledgement. It is similar to our traditional Indian post.
- Non-posted Write, we wait for the acknowledgement. It is similar to the today’s courier services. Naturally, non-posted write is more expensive than the posted write. It is much more expensive, though both writes are asynchronous.

67. **Why reading is done in parallel and writing is not in Hdfs?**
Reading is done in parallel because by doing so we can access the data fast. But we do not perform the write operation in parallel. The reason is that if we perform the write operation in parallel, then it might result in data inconsistency. For example, you have a file and two nodes are trying to write data into the file in parallel, then the first node does not know what the second node has written and vice-versa. So, this makes it confusing which data to be stored and accessed.
68. **Explain in brief the three Modes in which Hadoop can be run?**
The three modes in which Hadoop can be run are:

- Standalone (local) mode - No Hadoop daemons running, everything runs on a single Java Virtual machine only.
- Pseudo-distributed mode - Daemons run on the local machine, thereby simulating a cluster on a smaller scale.
- Fully distributed mode - Runs on a cluster of machines.

69. **Explain what are the features of Standalone local Mode?**
In stand-alone or local mode there are no Hadoop daemons running, and everything runs on a single Java process. Hence, we don't get the benefit of distributing the code across a cluster of machines. Since, it has no DFS, it utilizes the local file system. This mode is suitable only for running MapReduce programs by developers during various stages of development. Its the best environment for learning and good for debugging purposes.
70. **What are the features of fully distributed mode?**
In Fully Distributed mode, the clusters range from a few nodes to 'n' number of nodes. It is used in production environments, where we have thousands of machines in the Hadoop cluster. The daemons of Hadoop run on these clusters. We have to configure separate masters and separate slaves in this distribution, the implementation of which is quite complex. In this configuration, Namenode and Datanode runs on different hosts and there are nodes on which task tracker runs. The root of the distribution is referred as HADOOP_HOME.
71. **Explain what are the main features Of pseudo mode?**
In Pseudo-distributed mode, each Hadoop daemon runs in a separate Java process, as such it simulates a cluster though on a small scale. This mode is used both for development and QA environments. Here, we need to do the configuration changes.
72. **What is Rack Awareness?**
It is basically an algorithm that guides the Name Node on how the blocks are to be stored. Its main aim is to put a limit on the traffic in the network. It also manages and controls the replicas of each block.
73. **How will you decide whether you need to use the Capacity Scheduler or the Fair Scheduler?**

Fair Scheduling is the process in which resources are assigned to jobs such that all jobs get to share equal number of resources over time. Fair Scheduler can be used under the following circumstances -
i) If you wants the jobs to make equal progress instead of following the FIFO order then you must use Fair Scheduling.
ii) If you have slow connectivity and data locality plays a vital role and makes a significant difference to the job runtime then you must use Fair Scheduling.
iii) Use fair scheduling if there is lot of variability in the utilization between pools.
Capacity Scheduler allows runs the hadoop mapreduce cluster as a shared, multi-tenant cluster to maximize the utilization of the hadoop cluster and throughput.Capacity Scheduler can be used under the following circumstances -
i) If the jobs require scheduler detrminism then Capacity Scheduler can be useful.
ii) CS's memory based scheduling method is useful if the jobs have varying memory requirements.
iii) If you want to enforce resource allocation because you know very well about the cluster utilization and workload then use Capacity Scheduler.

74. **Explain about the different schedulers available in Hadoop.**

- FIFO Scheduler – This scheduler does not consider the heterogeneity in the system but orders the jobs based on their arrival times in a queue.
- COSHH- This scheduler considers the workload, cluster and the user heterogeneity for scheduling decisions.
- Fair Sharing-This Hadoop scheduler defines a pool for each user. The pool contains a number of map and reduce slots on a resource. Each user can use their own pool to execute the jobs.

75. **What is jps command used for?**
jps command is used to verify whether the daemons that run the Hadoop cluster are working or not. The output of jps command shows the status of the NameNode, Secondary NameNode, DataNode, TaskTracker and JobTracker.
76. **Explain the role of Hadoop YARN.**
In Hadoop 1.x, MapReduce handles resource management and job scheduling using JobTracker. However, slow processing is a significant drawback.
Hadoop 2.0 introduced YARN (Yet Another Resource Negotiator) to address these issues. YARN decouples resource management and job scheduling into separate components, which improves scalability and resource utilization. The primary components of YARN are:

- ResourceManager (RM): Manages resources across the cluster.
- NodeManager (NM): Manages resources and monitoring of individual nodes.
- ApplicationMaster (AM): Manages the lifecycle of applications, including their resource needs and execution.

How YARN works:

1. Job submission: The client submits an application to the YARN cluster.
2. Resource allocation: The ResourceManager starts allocating resources by coordinating with the ApplicationMaster.
3. ApplicationMaster registration: The ApplicationMaster registers itself with the ResourceManager.
4. Container negotiation: The ApplicationMaster negotiates resources (containers) from the ResourceManager.
5. Container launch: Once resources are allocated, the ApplicationMaster instructs NodeManagers to launch containers.
6. Application execution: The application code is executed within the allocated containers.
7. **Explain the role of checksum in detecting corrupted data. Also, define the standard error-detecting code.**
   Checksums identify corrupted data in HDFS. When data enters the system, it creates a small value known as a checksum. Hadoop recalculates the checksum when a user requests data transfer. If the new checksum matches the original, the data is intact; if not, it is corrupt.
   Error detecting code for Hadoop is CRC-32.
8. **How does HDFS achieve fault tolerance?**
   HDFS achieves fault tolerance through a replication process that ensures reliability and availability. Here's how the replication process works:

- Initial block distribution: When a file is saved, HDFS divides it into blocks and assigns them to different DataNodes. For example, a file divided into blocks A, B, and C might be stored on DataNodes A1, B2, and C3.
- Replication: Each DataNode holding a block replicates that block to other DataNodes in the cluster. For example, DataNode A1, which holds block A, will create additional copies on other DataNodes, such as D1 and D2. This means if A1 crashes, copies of A1 on D1 and D2 will be available.
- Handling failures: If DataNode D1 fails, you can retrieve the required blocks (B and C) from other DataNodes such as B2 and C3.

79. **What is Avro Serialization in Hadoop?**

- The process of translating objects or data structures state into binary or textual form is called Avro Serialization. It is defined as a language-independent schema (written in JSON).
- It provides AvroMapper and AvroReducer for running MapReduce programs.

80. **How to commission (adding) the nodes in the Hadoop cluster?**

- Update the network addresses in the dfs.include and mapred.include
- Update the NameNode: Hadoop dfsadmin -refreshNodes
- Update the Jobtracker: Hadoop mradmin-refreshNodes
- Update the slave file.
- Start the DataNode and NodeManager on the added Node.

81. **What is a rack-aware replica placement policy?**

- Rack Awareness is the algorithm used for improving the network traffic while reading/writing HDFS files to the Hadoop cluster by NameNode.
- NameNode chooses the Datanode which is closer to the same rack or nearby rack for reading/Write request. The concept of choosing closer data nodes based on racks information is called Rack Awareness.
- Consider the replication factor is 3 for data blocks on HDFS it means for every block of data two copies are stored on the same rack, while the third copy is stored on a different rack. This rule is called Replica Placement Policy.

82. **What is the command used for printing the topology?**
hdfs dfsadmin -point topology is used for printing the topology. It displays the tree of racks and DataNodes attached to the tracks.
83. **What is *Hadoop* and what are its *core components*?**
Apache Hadoop is a robust, open-source platform that facilitates distributed storage and processing of vast datasets across clusters of computers. It provides a cost-effective, powerful, and scalable foundation for Big Data analytics.
HDFS

- Purpose: Designed for high-speed access to application data and redundantly storing and managing large volumes of data.
- Key Features: Fault tolerance through data replication, high throughput for data access, data integrity, and coherency.
- HDFS Components: NameNode (manages the file system namespace and regulates access to files), DataNodes (store and manage data within the file system), Secondary NameNode (performs periodic checkpoints of the namespace).
Yet Another Resource Negotiator (YARN)
- Purpose: Serves as a distributed resource management system for allocating computational resources in Hadoop clusters.
- Key Features: Allows multiple data processing engines like MapReduce, Spark, and others to run on Hadoop in a shared manner.
- YARN Components: ResourceManager (manages and monitors cluster resources), NodeManager (manages resources on individual nodes), ApplicationMaster (coordinates

execution of a particular application or job), Containers (virtualized resources where application code runs).
MapReduce

- Purpose: A data processing model that processes large data sets across a Hadoop cluster in a distributed and parallel manner.
- Key Features: Implements data distribution, data processing, and data aggregation phases.
- MapReduce Components: Mapper (processes input data and generates key-value pairs), Reducer (aggregates the key-value pairs generated by the Mappers), Partitioner (distributes the key-value pairs across Reducers), Combiner (performs local aggregation on the Map output before it's shuffled to the Reducers).

Other Hadoop Ecosystem Components
Hadoop's rich ecosystem comprises tools and frameworks that extend its functionality to various Big Data tasks:

- Apache Hive: A data warehouse infrastructure that provides data summarization and ad hoc querying using a SQL-like language called HiveQL. It translates queries to MapReduce jobs.
- Apache HBase: A NoSQL database designed to operate on top of HDFS. It's capable of real-time read/write access to Big Data.
- Apache ZooKeeper: A centralized service for distributed systems that enables synchronization and group services, such as configuration management and distributed locks.
- Apache Oozie: A workflow scheduler to manage Apache Hadoop jobs.
- Apache Mahout: A library of scalable machine learning algorithms that can be run on Hadoop. It allows easy implementation of simple Hadoop workflows.
- Apache Pig: A platform for analyzing large datasets. It provides a high-level language, Pig Latin, which automates common data manipulation operations.

Hadoop is highly flexible and compatible with a wide array of hardware vendors and cloud service providers, making it a favorite choice for efficient Big Data management and analysis.

84. **What is a *Rack Awareness algorithm* in *HDFS*, and why is it *important*?**
n Hadoop Distributed File System (HDFS), the concept of Rack Awareness refers to HDFS's ability to understand the network topology and the physical location of its nodes. The primary goal of Rack Awareness is to optimize data storage reliability and data transfer efficiency (reducing inter-rack data traffic).
Multi-level Topology Knowledge
HDFS leverages a two-level topology consisting of Racks and Nodes:
Racks: Represent physical groups or locations of nodes, often within the same data center. They serve as a defining structure that organizes and groups nodes.

Nodes: Indicate individual machines or servers that store data and comprise Hadoop clusters.

By utilizing this generated network, Hadoop actively places blocks of data across different racks to achieve fault tolerance and performance

Rack Awareness in Hadoop's Transfer Modes

HDFS nodes optimize data transfer during block movement, particularly during operations such as rebalancing, replication, and block recovery. This results in a reduction in unnecessary and potentially costly inter-rack data traffic.

Why Is Rack Awareness Important?

Fault Tolerance:

Minimizes probability of losing data due to rack-wide failures.

Guarantees at least one replica of a block exists on an adjacent rack (except in the case of just one rack).

Network Efficiency:

Reduces bottlenecks and network congestion because data can be read closer to the requesting node, from the local rack where possible.

Lessens the risk of latency and data transfer overhead caused by inter-rack nodes.

Inter-Rack Balancing:

Distributes blocks equitably across racks, ensuring efficient use of storage and resources.

Facilitates uniform data access and load distribution.

Key Algorithms

HDFS uses Rack Awareness Algorithms to control how data placement and data transfer operations work across nodes and racks. Latter versions of HDFS come with dynamic rack discovery techniques to handle more complex and evolving data center configurations.

The default rack allocation strategy employs a simple Round-Robin approach to distribute blocks among the data cluster's racks, offering a fundamental yet efficient methodology for data storage and transfer operations. Different strategies, whether static or dynamic, including manual rack planning, might be more fitting for specific scenarios or requirements.

Rack Awareness Configuration

You can set and modify the Rack Awareness policies and configurations in your Hadoop setup using HDFS-specific configuration files, such as hdfs-site.xml.

For instance, you can make use of the following Rack Awareness configurations to:

Define the number of replicas managed within HDFS.

Outline the number of racks and their locations within your datacenter.

Assign nodes to particular racks and guarantee the right configuration of the clusters.

Allocate nodes fine-tuned to specific racks fitting your rack-based storage or networking needs.

85. **What are the different schedulers available in YARN?**

The different schedulers available in YARN are:

- **FIFO scheduler** - This places applications in a queue and runs them in the order of submission (first in, first out). It is not desirable, as a long-running application might block the small running applications
- **Capacity scheduler** - A separate dedicated queue allows the small job to start as soon as it is submitted. The large job finishes later compared to using the FIFO scheduler
- **Fair scheduler** - There is no need to reserve a set amount of capacity since it will dynamically balance resources between all the running jobs

86. **What is *Apache ZooKeeper* and why is it *important* for *Hadoop*?**
Apache ZooKeeper is a powerful coordination service essential for distributed systems like Hadoop, ensuring that tasks are carried out efficiently and consistently across multiple nodes.
Core Functions

- Configuration Management: Simplifies configuration updates in distributed systems, ensuring each node consistently applies any changes.
- Naming Services: Utilizes the hierarchical namespace to register and locate resources (often data nodes in Hadoop clusters).
- Synchronization: Offers different kinds of locks, allowing for fine-grained control over access to resources in distributed environments.
Hadoop's ZooKeeper Use-Cases
- HDFS Federation: Utilizes ZooKeeper for Active NameNode selection in federated HDFS clusters.
- HBase: The data store uses ZooKeeper to keep track of active and standby Master servers, providing high availability.
- Distributed Coordination Service: Multiple Apache distributed applications use it as a standard coordination mechanism.
ZooKeeper Guarantees
- Ordering and Atomicity: It offers strong consistency, ensuring that operations are applied in the same order across all nodes.
- Reliable Delivery: Every change notification is sent to clients, making sure they are aware of the system's current state.
How ZooKeeper Improves Hadoop Systems
- Fault Tolerance: In the event of node failures, tasks and services are redistributed to ensure continued functioning.
- High Availability: Redundant services and nodes are managed, and active ones are identified to provide consistent services.
- Synchronization: Ensures data consistency and avoids race conditions, making aggregate computations and shared data processing more reliable.

- Reliable Deployment: ZooKeeper takes care of tasks that are otherwise complex in distributed systems, like leader election and distributed locking, making the deployment and management of Hadoop services easier.

87. **How do you configure and optimize HDFS for better read/write performance?**

- Replication factor: Set to 3 for fault tolerance; reduce it for less critical data.
- Block size: Adjust the default block size based on the size of files (e.g., larger blocks for large files).
- Compression: Use compression codecs (like Snappy) to reduce storage and improve I/O.
- Data locality: Ensure computation happens near data using YARN and HDFS rack awareness.
- HDFS cache: Use in-memory caching for frequently accessed files.

88. **How do you ensure data integrity in HDFS, and what steps would you take if corruption is detected?**

- Checksums: HDFS calculates checksums for each block during write operations. Corruption is detected during read operations.
- fsck command: Use `hdfs fsck /path` to identify corrupt files.
- Recovery: Delete the corrupted block and let HDFS replicate from healthy replicas. If all replicas are corrupt, retrieve the file from backups or snapshots.
- Enable periodic block scanner checks on DataNodes for proactive detection.

89. **How do you address NameNode RPC call delays or timeouts?**

- Verify network health and latency between the NameNode and clients.
- Increase `dfs.namenode.handler.count` to handle more concurrent requests.
- Optimize **heap size** and perform GC tuning for the NameNode JVM.

90. **Your Hadoop cluster has been compromised due to a ransomware attack. How would you recover?**

```
- Immediate steps: Isolate the cluster to prevent further spread.
- Restore from snapshots or offsite backups.
- Audit logs to identify how the attack occurred and patch the
vulnerability.
- Implement stronger security practices, such as Kerberos, encryption, and
user access policies.
```

91. **How do you perform maintenance on a DataNode without affecting the cluster's performance?**

- Decommission the DataNode using `hdfs dfsadmin –decommission`.
- Ensure the data is replicated to other nodes before stopping services.
- Perform maintenance tasks and recommission the node using `hdfs dfsadmin –recommission`.

92. **Several DataNodes in the same rack fail simultaneously. How do you recover?**

- Verify and fix hardware or network issues causing the failure.
- Check the HDFS web UI or use `hdfs dfsadmin –report` to identify under-replicated blocks.
- Increase replication factor temporarily and run a **Balancer** after nodes are restored.

93. **Users report slow query performance, but resource usage is low. How would you debug this?**

- Check for I/O bottlenecks at the HDFS layer.
- Analyze YARN application logs for issues like data skew or shuffle problems.
- Inspect Hive query plans to identify unoptimized joins or filters.
- Ensure sufficient DataNode heap size to avoid frequent GC pauses.

94. **What are some best practices for maintaining a healthy Hadoop cluster?**
    - Enable rack awareness for better fault tolerance.
    - Regularly monitor cluster health using tools like Ambari or Cloudera Manager.
    - Automate log rotation and clean-up tasks.
    - Schedule snapshot-based backups for critical data.
    - Stay up to date with security patches and upgrades.