What you did and how?

1. What is your last project and how did you use Hadoop?

Have a well constructed answer for this question since most likely this will be your very first question in an interview.

- Mention about the company in 2 sentences.
- Explain the use case of the project.
- What are the input data points for Hadoop cluster – Database, flat files from other applications etc.
- Hadoop ecosystem tools you are using for specific tasks. For e.g. Sqoop for data ingestion, Pig for transformation, Hive for analysis etc.

Let's assume you are working for a big online retail shopping company (Retail Inc.). The answer to the question could be something like below –

Retail Inc. is a fast growing online retail website specializing in Men's and Women's high fashion apparel. One of the key challenges we face at Retail Inc. is maintaining customer loyalty and to provide them with a customized online shopping experience.

Retail Inc. collects and analyzes large amounts of data from our customers 24×7 from several data points – websites, mobile apps, credit card, social media and coupons redemption. Data from these data points could be structured and unstructured in few cases.

All this data is collected, aggregated and analyzed in the Hadoop cluster to find customer's shopping patterns and to provide them with a unique personalized shopping experience. The analysis that is done in our Hadoop cluster help our strategic teams make cross sell, up sell decisions and devise targeted marketing strategies for our customers.

2. What are your day to day responsibilities?

- Explain the key business areas you support.
- What you support and how – data ingestion, data transformation, administration, optimization etc.
- Tools you work with – Pig, Hive, Sqoop etc.

Sample Response:

I am responsible for the data transformation tasks which are written using PIG scripts and the scripts run every night. I also work with the business to gather business requirements and write, optimize Hive queries for data analysis. I also work with the business to gather new requirements to write transformation tasks using Pig for which I design the data ingestion process using Sqoop.

3. Tell us about your Hadoop cluster?

Here are few items to keep in mind when you construct an answer.

- Distribution and version used – For e.g. CDH4 or HDP2
- No of nodes – For e.g. 60 node cluster
- Data volume of the cluster – For e.g. 60 TB cluster (180 TB with replication factor of 3)
- Mention whether you are running Hadoop 1 or Hadoop 2 – For e.g. YARN or JobTracker setup
- Special setup (if any) – HA, Federation etc.

Sample Response

We are running a CDH4 cluster with 120 Nodes. Our total data volume is 200 TB with a replication factor of 3. Our NameNode setup is configured in an Active-Standby High Availability setup. Our applications run on YARN.

4. How do you deploy MapReduce job or Pig or Hive scripts into your cluster?

Each company is different in how code or scripts are deployed in to production. A sample response can look something like below if you use Oozie to coordinate your jobs.

Sample Response

All our production jobs are coordinated using Oozie. Once the development process is complete we bundle the MapReduce job, Pig or Hive scripts into an Oozie job along with coordinator, workflow xml and properties files needed to run the job. The bundle is placed on one of our Edge nodes (look for the question on Edge node below) outside the cluster along with instructions to your admin or deployment team on how to start the Oozie job.

During deployment, the administrator will follow our instructions to pick up the files and deploy them to HDFS and start the job.

5. How did you decide on the number of nodes when you designed your cluster?

This question is aimed at someone who also involved in the initial design of the cluster. There is no straight forward answer to this question. Here are some of the practices or approaches that would go in when designing a cluster.

The primary reason for building a Hadoop cluster is to distribute the execution of your jobs to reduce the amount of execution time.

The most practical way to do decide on the number of nodes is to start with a few node cluster in a development environment. Use the Benchmarking tools that come out of the box like *TestDFSIO* and *MRBench* to benchmark the current cluster. This will give you a very high level idea of whether your initial memory and CPU requirements in your initial test cluster will meet your needs.

Next step is to run an optimized MapReduce job or Pig/Hive script which you would except to run on a production environment. Make sure the job runs on the same amount of data as it would run in prod.

Now adjust the memory settings for your Map or Reduce and any other performance optimizations needed to make the job execute in the agreed SLA time. With this baseline at hand, think about the maximum number of scheduled jobs and adhoc requests your cluster will be expected to handle at peak usage time. Extrapolate your baseline in terms of memory, CPU and number of nodes to meet the SLA for your peak usage time. Always leave 20% room for unexpected spikes in usage.

Also remember the key benefit of Hadoop cluster is horizontal scalability so you add nodes to your cluster anytime.

6. How do you monitor jobs (MapReduce, Pig or Hive) in production?

Each company is different in how the jobs are monitored in production. A sample response can look like below. The answer is structured to match with our answer to the deployment question.

Sample Response:

All our jobs are coordinated and executed using Oozie. Our support team uses the Oozie web interface to look up the status of the jobs. All jobs are configured to send out emails to the support team in case of failure and success. Upon receiving a failure email, the support team will try to investigate the problem and call the appropriate individuals as necessary.

Note: Administration tools like Cloudera Manager (for e.g.) also provide ways to monitor jobs in your cluster.

7. How do you debug a failed job in production?

A job could fail for several reasons in a distributed environment like Hadoop. The key to fix a failed job is to answer the below 3 questions in the same order as listed.

- What failed?
- Where it failed?
- Why it failed?

There could you several reasons for failure. Some of the most common failure reasons are below.

- Data related runtime exceptions
- Resource related (for e.g. Out Of Memory)
- Application or programming errors

Sample response:

I will get the application id of the failed job and will go to the Application Master's URL to look up the failed application. I will look for the overall progress made by the job. That is, I will look for how many mappers or reducers failed over all to get an idea about the extent of the failure. (This will answer what failed) If the failure is with one or two mappers, I will drill down to the failed task note down the node it ran on and will inspect the logs from the Application Master UI itself (which will show only few lines from the error) to see the reason for the failure. (This will answer where it failed) If you want to see the entire log you will ask the admin (if you don't have necessary permissions) to get the entire log from the node or Cloudera Manager for the failed. Once the reason for failure is identified I will take appropriate action to fix the job. (This will answer why it failed)

8. How do you debug a performance issue or a long running job?

This is an open ended question and the interviewer is trying to see the level of hands-on experience you have in solving production issues. Use your day to day work experience to answer this question. Here are some of the scenarios and responses to help you construct your answer. On a very high level you will follow the below steps.

- Understand the symptom
- Analyze the situation
- Identify the problem areas
- Propose solution

Scenario 1 - Job with 100 mappers and 1 reducer takes a long time for the reducer to start after all the mappers are complete. One of the reasons could be that reduce is spending a lot of time copying the map outputs. So in this case we can try couple of things.

1. If possible add a combiner to reduce the amount of output from the mapper to be sent to the reducer 2. Enable map output compression - this will further reduce the size of the outputs to be transferred to the reducer.

Scenario 2 - A particular task is using a lot of memory which is causing the slowness or failure, I will look for ways to reduce the memory usage.

1. Make sure the joins are made in an optimal way with memory usage in mind. For e.g. in Pig joins, the LEFT hand side tables are sent to the reducer first and held in memory and the RIGHT most table is streamed to the reducer. So make sure the RIGHT most table is largest of the datasets in the join. 2. We can also increase the memory requirements needed by the map and reduce tasks by setting - *mapreduce.map.memory.mb* and *mapreduce.reduce.memory.mb*

Scenario 3 - Understanding the data helps a lot in optimizing the way we use the datasets in PIG and HIVE scripts.

1. If you have smaller tables in join, they can be sent to distributed cache and loaded in memory on the Map side and the entire join can be done on the Map side thereby avoiding the shuffle and reduce phase altogether. This will tremendously improve performance. Look up *USING REPLICATED* in Pig and *MAPJOIN* or *hive.auto.convert.join* in Hive 2. If the data is already sorted you can use *USING MERGE* which will do a Map Only join 3. If the data is bucketted in hive, you may use *hive.optimize.bucketmapjoin* or *hive.optimize.bucketmapjoin.sortedmerge* depending on the characteristics of the data

Scenario 4 - The Shuffle process is the heart of a MapReduce program and it can be tweaked for performance improvement.

1. If you see lots of records are being spilled to the disk (check for Spilled Records in the counters in your MapReduce output) you can increase the memory available for Map to perform the Shuffle by increasing the value in *io.sort.mb*. This will reduce the amount of Map Outputs written to the disk so the sorting of the keys can be performed in memory. 2. On the reduce side the merge operation (merging the output from several mappers) can be done in disk by setting the *mapred.inmem.merge.threshold* to 0

9. How do you install, configure a Hadoop cluster?

There are several ways to install and configure a Hadoop cluster. Here are some.

- Hadoop management tools like Cloudera Manager, Apache Ambari etc.
- Tools like Puppet, Apache Whirr
- Manual setup with shell scripts (or other relevant scripting languages)

Sample response

In my client location, Cloudera Manager is used to configure, install, monitor, add or remove services, add or remove hosts to our Hadoop cluster

Note: In most companies developers will not have access to Cloudera Manager for the production environment. Only Administrators will have access to the tool.

10. How do you monitor your Hadoop cluster?

Hadoop related services can be monitored by tools like Cloudera Manager or Apache Ambari. These tools will provide a unified view of the cluster and will be primarily used by Hadoop Administrators. In many companies OS level monitoring will be handled by sophisticated tools like Ganglia, ITRS and will be handled by separate infrastructure teams.

Sample response

In our company we use Cloudera Manager to monitor Hadoop related services or get a unified status of the cluster. Our infrastructure team use Ganglia to monitor the nodes at the OS level.

Note: In most companies developers will not have access to Cloudera Manager for the production environment. Only Administrators will have access to the tool.

11. How is security handled in your cluster?

We have seen Hadoop clusters where security is not configured at all but in places where security is configured below configuration is usually the norm.

Kerberos for authentication Apache Sentry for authorization

12. Where is your cluster hosted - in house or cloud?

A Hadoop cluster can be hosted in house, that is managed and maintained in one of the company's datacenter and it can be hosted on the cloud like Amazon Web Services (AWS) or Google cloud. The answer to the question depends on your setup.

13. Have you used Amazon AWS?

This question is an important question to an interviewer especially if their cluster is hosted on AWS. Even if your cluster is hosted in house and even if you are not using AWS make sure you know the services offered by AWS and have a basic understanding of how to use them and how they work. Check out http://aws.amazon.com/ for offers on new AWS accounts

14. What are the services you have used in AWS?

If you are not familiar with AWS make sure you get yourself familiarized at least with the following services. Sometimes no experience with AWS could be a deal breaker for some interviewers so make sure you at least have a basic understanding of different services offered by AWS.

- Elastic Cloud Compute (EC2)
- Elastic Map Reduce (EMR)
- Simple Storage Service (S3)
- Identity and Access Management (IAM)

HDFS

15. How does the DataNode know about name node?

When the DataNode comes up it reads the value in fs.defaultFS - formerly fs.default.name (which holds the location of the NameNode) and caches the location.

16. If the block size of the cluster is 64 MB and you have a file with 100 KB in size, how much space does the file really occupy when it is stored on disk?

File size on disk is determined by the cluster size (or block size) of the underlying local file system. If the cluster size of the local file system is 4 KB then the file of size 100 KB will need 25 clusters (25 * 4 KB) to store the file on disk. So the answer is, the file size on disk would be 100 KB and not 64 MB

If the file size is 5 KB then 2 clusters are needed to store the file and the file size on disk would be 8 KB. HDFS does not use any more disk space than it is required by the local file system to store the file on disk.

17. Why Hadoop uses huge block size of 128 MB or 256 MB while storing files in HDFS?

Hadoop is designed to handle big files. If the block size is smaller, the NameNode will have to manage a lot of blocks and this would put an enormous stress on the NameNode and will create performance issues during HDFS reads and writes

The other important reason is to minimize disk seeks. A single HDFS block of 128MB will be written to disk sequentially. Therefore there is a fair chance that the data will be written into contiguous space on disk (consisting of multiple blocks next to each other). Since there is a good chance that the data will be stored in contiguous space the reads will be much faster since it avoids random seeks.

18. How corruption is handled by HDFS?

There are several reasons for data corruption - faults in a storage device, network faults, or buggy software.

- When a client creates an HDFS file, it computes a checksum of each block of the file and stores these checksums in a separate hidden file in the same HDFS namespace.
- When a client retrieves file contents it verifies that the data it received from each DataNode matches the checksum stored in the associated checksum file.
- If the checksum did not match the client will retrieve that block from another DataNode that has a replica of that block.
- NameNode will be notified of the corrupted block and it will arrange for replicating the block to maintain the replication factor in the cluster.

19. Have you encrypted files in HDFS?

This question is asked to understand the candidate's knowledge of the encryption functionality in HDFS. HDFS implements transparent, end-to-end encryption.

- Transparent - data read from and written to special HDFS directories is *transparently* encrypted and decrypted without requiring changes to user application code.
- End-to-end - means the data can only be encrypted and decrypted by the client. Encryption is supported on both meaning data on persistent media, such as a disk (at-rest encryption) as well as when data is travelling over the network (in-transit encryption).

20. During a HDFS Read operation how does a DataNode with a replicated block gets picked up by NameNode for a read operation?

- Client contacts NameNode to determine the locations of the blocks for the first few blocks in the file.
- For each block, the NameNode returns the addresses of the DataNodes (sorted by proximity to the client) that have a copy of that block.
- Client then connects to the first (closest) DataNode for the first block in the file. Data is streamed from the DataNode back to the client.
- When the end of the block is reached the connection to the DataNode is closed and then client will start reading the next block from the best (closest) DataNode for the block.

21. How do you set limits on HDFS usage for a particular directory?

The Hadoop Distributed File System (HDFS) allows the administrator to set quotas for the number of names (Name quota) used and the amount of space (Space quota) used for individual directories. File, directory creations and block allocations will fail if the quota is exceeded.

- Name quota

Sets hard limit on the number of file and directory names under a specific directory.

- Space quota

Sets hard limit on the number of bytes used by files under a specific directory. Each replica of a block counts against the quota.

Illustration

$ dfsadmin -setQuota <N> <directory>

Set the name quota to be N for directory.

$ dfsadmin -setSpaceQuota <N> <directory>

Set the space quota to be N bytes for directory.

22. How do you copy the file from one Hadoop cluster to another?

DistCp (distributed copy) is a tool used for large inter/intra-cluster copying.

Illustration

$ hadoop distcp hdfs://nn1:8020/foo/bar hdfs://nn2:8020/bar/foo

23. How do you change the block size of files already in the cluster?

Distcp can be used to change the block size of the original dataset. After this command completes, you can remove the original data.

Illustration

$ distcp -Ddfs.block.size=256 /path/to/inputdata /path/to/inputdata-with-largeblocks

24. What is the need for a rebalance operation in HDFS?

Over a period of time when new DataNodes to an existing cluster the data across the cluster will be unbalanced and data might not be uniformly placed across the DataNodes.

HDFS provides a tool called balancer that analyzes block placement and rebalances data across the DataNode.

Illustration

$ hadoop balancer

25. What is a Safe mode?

During startup the NameNode waits for DataNodes to report their blocks and the NameNode hold the block locations in memory. During this time, the NameNode will be in Safe mode so that it does not prematurely start replicating the blocks though enough replicas already exist in the cluster. In Safe mode the NameNode is essentially in read-only mode. NameNode will leave Safe mode automatically after the DataNodes have reported that most file system blocks are available.

26. Explain the checkpointing process in HDFS.

NameNode persist its HDFS namespace using two files: fsimage and edits

- fsimage - stores the state (metadata) of HDFS
- edits - stores modifications to the file system as a log appended to a native file system file

Any modifications to the file system are not directly written in to fsimage file because the fsimage will be larger in size and a write operation to a big file will create a performance bottleneck. So all modifications are logged in to edits file. Checkpointing process helps in keeping the edits file to a manageable size.

During a Namenode restart, Namenode will apply all the changes from the edits file to fsimage. A bigger edits file will slow the restart operation. Checkpointing process is done by either Secondary Namenode (deprecated) or Checkpoint node.

During the checkpointing process the Secondary Namenode (deprecated) or Checkpoint node will

- Download fsimage and edits from the active NameNode
- Apply the edits log to fsimage file locally
- Upload the new image back to the active NameNode.

The Secondary or the checkpoint node is usually run on a different machine than the primary NameNode since its memory requirements are on the same order as the primary NameNode.

27. What properties control the checkpointing process?

The start of the checkpoint process is controlled by two configuration parameters.

- dfs.namenode.checkpoint.period

Defines the maximum interval between two consecutive checkpoints. Default is 1 hour.

- dfs.namenode.checkpoint.txns

Defines the number of uncheckpointed transactions on the NameNode which will force an urgent checkpoint, even if the checkpoint period has not been reached. Default is 1 million.

28. What is the purpose of secondary or checkpoint nodes?

Secondary and checkpoint nodes help in the checkpointing process. Secondary Namenode is deprecated and replaced by Checkpoint nodes in the latest version of Hadoop. Look above question for details on checkpointing process.

29. What is the difference between checkpoint and back up NameNode?

The Backup node provides the same checkpointing functionality as the Checkpoint node and in addition to that maintains an in-memory, up-to-date copy of the file system namespace that is always synchronized with the active NameNode state.

Checkpoint node or Secondary NameNode will need to download fsimage and edits files from the active NameNode in order to create a checkpoint.

Backup NameNode accepts a journal stream of file system edits from the NameNode and persists to disk so Backup node does not need to download fsimage and edits files from the active NameNode.

Backup node also applies edits into its own copy of the namespace in memory, thus creating a backup of the namespace.

Currently NameNode supports one Backup node at a time. No Checkpoint nodes may be registered if a Backup node is in use.

30. What is a HA cluster or configuration?

In a typical HA cluster; two separate machines are configured as NameNodes. At any point in time, exactly one of the NameNodes is in an Active state, and the other is in a Standby state. The Active NameNode is responsible for all client operations in the cluster, while the Standby is simply acting as a slave, maintaining enough state to provide a fast failover if necessary.

31. What is the need for HA configuration when we have a Secondary Namenode already in the cluster?

The term "secondary name-node" is somewhat misleading. It is not a name-node in the sense that data-nodes cannot connect to the secondary name-node, and it can't replace the primary name-node in case of its failure without some serious changes.

The purpose of the secondary name-node is to perform periodic checkpoints. NameNode records the metadata information of the Namespace in an edits log and this information from the edits log should be periodically applied to FSIMAGE file. The secondary name-node periodically downloads current name-node image and edits log files, joins them into new image and uploads the new image back to the NameNode.

32. How is namespace or metadata information shared between the Active and Standby Namenode in a HA cluster?

Both Active and Standby NameNodes have access to a directory on a shared storage device on an NFS mount.

When any namespace modification is performed by the Active node, it records the modification to an edit log file stored in the shared directory. The Standby node is constantly watching this directory for edits, and as it sees the edits, it applies them to its own namespace. In the event of a failover, the Standby will ensure that it has read all of the edits from the shared storage before promoting itself to the Active state. This ensures that the namespace state is fully synchronized before a failover occurs.

33. What is fencing?

In a HA configuration, during an unexpected failover the previously active NameNode could still be running and could still think and act as an active NameNode even though the cluster would be now controlled by the Standby NameNode (now active). Fencing is a method to prevent the previously active NameNode from doing any damage or corruption to the HDFS namespace during an unexpected failover. Here are couple of widely used fencing methods.

- Killing the NameNode process
- Revoking access to NameNode's shared storage

dfs.ha.fencing.methods property controls fencing method.

34. Is there another way to share information between the NameNodes (Active and Standby) other than using a shared NFS mount?

Other alternative to NFS mount is to use Quorum Journal Manager (QJM) to share edit logs between the Active and Standby NameNodes.

In order for the Standby node to keep its state synchronized with the Active node, both nodes communicate with a group of separate daemons called "JournalNodes" (JNs). When any namespace modification is performed by the Active node, it durably logs a record of the modification to a majority of these JNs. The Standby node is capable of reading the edits from the JNs, and is constantly watching them for changes to the edit log. As the Standby Node sees the edits, it applies them to its own namespace. In the event of a failover, the Standby will ensure that it has read all of the edits from the JounalNodes before promoting itself to the Active state. This ensures that the namespace state is fully synchronized before a failover occurs.

35. Since block locations are stored in memory of the Namenode and not in the edits log file, how does the Standby node know about the block locations in a HA environment?

DataNodes are configured with the location of both NameNodes, and send block location information and heartbeats to both Active and Standby nodes.

36. How do you deal with Namenode failure?

If the NameNode fails and you can restart it on the same physical node then there is no need to shutdown data-nodes, just the name-node need to be restarted. If you cannot use the old NameNode anymore you will need to copy the latest image somewhere else. The latest image can be found either on the node that used to be the primary before failure if available; or on the secondary name-node. The latter will be the latest checkpoint without subsequent edits logs that is the most recent name space modifications may be missing there. You will also need to restart the whole cluster in this case.

In case of HA configuration, the Standby node will quickly become the primary and Active NameNode of the cluster.

37. What is HDFS federation?

The NameNode stores the entire file system metadata in memory. This limits the number of blocks, files, and directories supported on the file system to what can be accommodated in the memory of a single NameNode. Federation solves this problem by introducing multiple independent NameNodes. The multiple NameNodes are independent and don't require coordination with each other. Each NameNode is responsible for the Namespace it manages. The DataNodes are used as common storage for blocks by all the NameNodes. Each DataNode registers with all the NameNodes in the cluster. DataNodes send periodic heartbeats and block reports and handles commands from the NameNodes.

38. How do you benchmark your Hadoop cluster with tools that come with Hadoop?

- TestDFSIO

TestDFSIO gives you an understanding of the I/O performance of your cluster. It is a read and write test for HDFS and helpful in identifying performance bottlenecks in your network, hardware and set up of your NameNode and DataNodes.

- NNBench

NNBench simulate requests for creating, reading, renaming and deleting files on HDFS and is useful for load testing NameNode hardware configuration

- MRBench

MRBench is a test for the MapReduce layer. It loops a small MapReduce job for a specific number of times and checks the responsiveness and efficiency of the cluster.

Illustration

TestDFSIO write test with 100 files and file size of 100 MB each.

$ hadoop jar /dirlocation/hadoop-test.jar TestDFSIO -write -nrFiles 100 -fileSize 100

TestDFSIO read test with 100 files and file size of 100 MB each.

$ hadoop jar /dirlocation/hadoop-test.jar TestDFSIO -read -nrFiles 100 -fileSize 100

MRBench test to run a lob of 50 small test jobs

$ hadoop jar /dirlocation/hadoop-test.jar mrbench -numRuns 50

NNBench test that creates 1000 files using 12 maps and 6 reducers.

*$ hadoop jar /dirlocation/hadoop-test.jar nnbench -operation create_write *
*-maps 12 -reduces 6 -blockSize 1 -bytesToWrite 0 -numberOfFiles 1000 *
-replicationFactorPerFile 3

MapReduce

39. What is the problem with running Hadoop on standalone or Pseudo-Distributed mode?

Standalone or Pseudo-Distributed is not distributed configuration and we will not get any distributed benefits from Hadoop.
In Standalone or Pseudo-Distributed the entire environment is prone to Single Point Of Failure (SPOF)

40. Can you change the number of mappers to be created for a job in Hadoop?

No. The number of mappers is determined by the no of input splits.

41. How do you unit test a MapReduce program?

MRUnit is a testing framework based on JUnit and it can be used to unit test MapReduce programs.
Illustration

```
@Test
public void testMapper() {
mapDriver.withInput(new LongWritable(), new Text("655209;1;796764372490213;804422938115889;6"));
mapDriver.withOutput(new Text("6"), new IntWritable(1));
mapDriver.runTest();
}
@Test
public void testReducer() {
List<IntWritable> values = new ArrayList<IntWritable>();
values.add(new IntWritable(1));
values.add(new IntWritable(1));
reduceDriver.withInput(new Text("6"), values);
reduceDriver.withOutput(new Text("6"), new IntWritable(2));
reduceDriver.runTest();
}
```

42. What is usage of ChainMapper and ChainReducer?

Let's say you have a big data file which is semi-structured and you would like to process the file. You could clean or preprocess the data all in one Mapper and in many cases this would work perfectly OK. But in some cases performing many operations in a single mapper could be problematic due to memory restrictions. You don't want the single Mapper to be the bottleneck and slow down your entire job. Also, having performing all types of functionalities in a single Mapper lacks code isolation. In this scenario creating specialized Mappers is ideal.
The best way to approach the problem is to split the Mapper in to multiple mappers each one doing a very specific task and chain them together along with a Reducer The Mapper classes are invoked in a chained (or piped) fashion, the output of the first becomes the input of the second, and so on until the last Mapper, the output of the last Mapper will be written to the task's output and will flow through in to Reducer.
The structure of your program will look like below and as you can see we are chaining Mappers and Reducer and hence they are called ChainMapper and ChainReducer.
Mapper1 -> Mapper2 -> Reducer -> Zero or More Mappers
Usage: https://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/lib/ChainMapper.html

43. How to create a map only job?

Use the setNumReduceTasks method in the Job class to set the number of reduce tasks to 0 or set mapred.reduce.tasks property to 0 in your Job configuration.
Illustration

```
Job job = Job.getInstance(getConf(), "jobname");
job.setNumReduceTasks(0);
```

44. What are the functions of InputFormat?

- Validate input data is present and check input configuration
- Create InputSplit(s) from blocks
- Create RecordReader implementation to create key/value pairs from the raw InputSplit. These pairs will be sent one by one to their mapper.

45. What is a Record Reader?

A RecordReader uses the data within the boundaries created by the input split to generate key/value pairs. Each of the generated Key/value pair will be sent one by one to their mapper.

46. What is a sequence file in Hadoop?

Sequence file is used to store binary key/value pairs. Sequence files support splitting even when the data inside the file is compressed which is not possible with a regular compressed file. You can either choose to perform a record level compression in which the value in the key/value pair will be compressed. Or you can also choose to choose at the block level where multiple records will be compressed together.

47. List some key benefits in using a Sequence file as opposed to a regular text file?

There are 3 key important benefits we can get from a Sequence File
1. In Sequence files key-value pairs are stored in binary format. Which is compact and very efficient for processing and transfer 2. Sequence file can be compressed at both record and block level. Block level compression is preferred as multiple records will be compressed offering an efficient compression. 3. A compressed text file does not allow Splitting. Where as a compressed (record or block level) Sequence file still allows splitting which is a key benefit.

48. Can you use Combiner in your MapReduce program that involves calculating an average of X elements?

Assume you have a stocks dataset, in which you have stock information like the symbol, close price, opening price, volume and trade date. Now you are asked to calculate the average volume of each stock symbol using MapReduce. Can you use Combiner in your MapReduce program?
Even though Hadoop framework allows using a combiner in any MapReduce job, the usage of combiners may not make sense for all kinds of problems.
So the answer to this question is – No. Combiners cannot be used in all instances as it may distort the output.
Let's see this with an illustration. In your Mapper you will emit symbol as key and volume as value for each record you process. In the reducer side all the key value pairs from all Mappers will be aggregated by the symbol (key) and then you can calculate the average on the list of values in the reducer.

- Without Combiner

{KEY, VALUE} = {STOCK SYMBOL, VOLUME}
Mapper 1 Output - {APPL, 100} ; { APPL, 90} ; { APPL, 101}
Mapper 2 Output - { APPL, 60} ; { APPL, 105}
Reduce Input - { APPL, (100, 90, 101, 60, 105)}
Reduce Output - 91.2

- With Combiner

{KEY, VALUE} = {STOCK SYMBOL, VOLUME}
Mapper 1 Output - { APPL, 100} ; { APPL, 90} ; { APPL, 101}
Combiner 1 Input - { APPL, (100, 90, 101)}
Combiner 1 Output- { APPL, 97}
Mapper 2 Output - { APPL, 60} ; { APPL, 105}
Combiner 2 Input - { APPL, (60, 105)}
Combiner 2 Output- { APPL, 82.5}
Reduce Input - { APPL, (97, 82.5)}
Reduce Output - 89.75

As you can see, the average volume when you don't use combiner is 91.2 and when you use a combiner it is 89.75. Hence using a combiner in this clearly distorted the output and may not be used in this instance.

49. In what instances you may choose to use a Combiner?

This is a slight variation of the previous question and it needs a more generalized answer.
Combiners can only be used on the functions that are commutative (a.b = b.a) and associative {a.(b.c) = (a.b).c}.
This means that combiners may operate only on a subset of your keys and values or may not execute at all, still you want the output of the program to remain same.

50. What is a Partitioner?

Partitioner is responsible for assigning keys from the Map output to Reducers. The key used to derive the partition is typically by a *hash function*. The total number of partitions is the same as the number of reduce tasks for the job.
HashPartitioner is the default Partitioner.

51. If you have multiple datasets each with a different type how would you handle it?

MultipleInputs can be used to set the input path and the corresponding InputFormat to be used for the datasets in the input path.
Illustration
MultipleInputs.addInputPath(merge, new Path("inputPath1"), SequenceFileInputFormat.class, Mapper1.class);
MultipleInputs.addInputPath(merge, new Path("inputPath2"), TextInputFormat.class, Mapper2.class);

52. How do you sort the values from map's output in descending order before it reaches the reduce function?

SortComparator is used to define how map output keys are sorted. The default SortCompartor can be changed as shown below. You have to write a custom comparator to sort the key values in reverse. For e.g. you can write a custom comparator named *ReverseComparator* extending *WritableComparator* and override the compare method to give your implementation for sorting the keys in descending order.
Once the *ReverseComparator* class is created set the class using *setSortComparator* method.
Illustration
job.setSortComparator(ReverseComparator.class);

53. How does a Map function handle a record which does not logically end in the Input Split that is being processed?

The *RecordReader* implementation in the *InputFormat* will take care of the logical record boundaries. For example the *LineRecordReader* in *TextInputFormat* will take care of the logical record boundary.
If the last line in the first input split starts in the first split and end in the second split, then the *RecordReader* will reach out to the second split and read the remaining of the last line and will give it to the first mapper.
The *RecordReader* of the second mapper will skip the record that is spread between the first and the second input split as it is already processed by the first mapper.

54. How can you force the split size always greater than the size of the block?

mapred.min.split.size controls the minimum input split size
Assume the block size is 64 MB and mapred.min.split.size is set to 128 MB then according to the below formula the split size will be 128 MB
Input Split Size = max(minimumSize, min(maximumSize, blockSize))
However, there is not real benefit in forcing the split size to be greater than the block size. Doing so will decrease the number of mappers but at the expense of sacrificing data locality.

55. What is the use case of map only jobs?

For e.g., if you want to convert many text files into PDFs, each file can work on a Mapper and the Mapper can be designed to do PDF conversion. In this case it makes sense to have a map only join.
For instance, Sqoop import jobs are mostly map only since the data will be extracted from RDBMS and written to HDFS all in the Map function since there is no need for data aggregation that would warrant a Reduce function.
In Map only jobs the need for shuffle phase and reduce phase is eliminated which will give performance improvements. For e.g. Map only joins in Pig and Hive

56. When the intermediate map output is spilled to disk, does it get spilled to local file system or HDFS?

Local file system

57. How did you orchestrate, coordinate or link your MapReduce jobs?

There are few tools that can do the job. Oozie is one such tool.
Oozie is integrated with the rest of the Hadoop stack supporting several types of Hadoop jobs out of the box (such as Java map-reduce, Streaming map-reduce, Pig, Hive, Sqoop and Distcp) as well as system specific jobs (such as Java programs and shell scripts).

58. What is Side Data Distribution?

Side data can be defined as extra read-only data needed by a job to process the main dataset. Side Data Distribution is to make side data available to all the map or reduce tasks (which are spread across the cluster) in a convenient and efficient fashion. Look at distributed cache later.

59. What is the need for Distributed Cache when the files are accessible by all the DataNodes through HDFS?

Say we have 1000 mappers for a Job and all the mapper will need to look up a small file in HDFS when they execute.

- When the job gets submitted mappers would start running (as per the available slots) and all mappers will query the NameNode for the small file from HDFS at the same time and soon the small file will become the bottleneck. By adding the file to the Distributed Cache, Hadoop's MapReduce framework will copy the necessary files on to the slave node before any tasks for the job are executed on that node so when the Map starts to run the file is already available in the local file system.
- Files are only copied once per job so if a data node is picked up to run a Mapper 5 times the small file is copied only once and is available for all 5 executions

60. Are there any ways available to distribute Side Data other than using distributed cache?

- It is possible to cache side-data in memory in a static field, so that tasks of the same job that run in succession on the same *TaskTracker* can share the data. Task JVM Reuse needs to be enabled to make several tasks from the job run in succession on the same JVM.
- Other way is to set arbitrary key-value pairs in the job configuration using the various setter methods on *JobConf* (inherited from *Configuration*). This is very useful if you need to pass a small piece of metadata to your tasks. To retrieve the values in the task, override the *configure()* in the Mapper or Reducer and use a getter method on the *JobConf* object passed in.

When using the above techniques make sure not more than a few kilobytes of data because it can put pressure on the memory usage in the Hadoop daemons, particularly in a system running hundreds of jobs.

61. Assume you have Research, Marketing and Finance teams funding 60%, 30% and 10% respectively of your Hadoop Cluster. How will you assign only 60% of cluster resources to Research, 30% to Marketing and 10% to Finance during peak load?

Capacity scheduler in Hadoop is designed to support this use case. Capacity scheduler supports hierarchical queues and capacity can be defined for each queue.

For this use case, you would have to define 3 queues under the root queue and give appropriate capacity in % for each queue.

Illustration

Below properties will be defined in *capacity-scheduler.xml*

```
<property>
<name>yarn.scheduler.capacity.root.queues</name>
<value>research,marketing,finance</value>
</property>
<property>
<name>yarn.scheduler.capacity.research.capacity</name>
<value>60</value>
</property>
<property>
<name>yarn.scheduler.capacity.research.capacity</name>
<value>30</value>
</property>
<property>
<name>yarn.scheduler.capacity.research.capacity</name>
<value>10</value>
</property>
```

62. What are the policies with Fair scheduler?

Fair scheduling is a method of assigning resources to applications such that all applications get an equal share of resources on average over time.

Fair scheduler allows configuration of queues. Fair scheduler allows setting a different policy for each queue to allow sharing the queue's resources in any which way the user wants.

- Fair Share
- FIFO
- Dominant Resource Fairness

63. How do you handle multiple inputs for a MapReduce action in Oozie?

Look at the Illustration below that will go in to the MapReduce action in Oozie. *Tip:* You don't have to memorize the properties. If you have a MapReduce job already running in cluster just look up its job.xml to find the properties you are looking for.

Illustration

```
<property>
<name>mapreduce.input.multipleinputs.dir.mappers</name>
<value>
${nameNode}/user/${wf:user()}/input1; com.Illustration.Mapper1,
${nameNode}/user/${wf:user()}/input2;com.Illustration.Mapper2
</value>
</property>
<property>
<name>mapreduce.input.multipleinputs.dir.formats</name>
<value>
${nameNode}/user/${wf:user()}/input1;org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat,
${nameNode}/user/${wf:user()}/input2;org.apache.hadoop.mapreduce.lib.input.TextInputFormat
</value>
</property>
```

64. How did you handle errors with oozie?

Oozie has an email action and you can set up to end out an email with the error to alert appropriate groups.

```
<action name="sendErrorNotifications">
<email xmlns="uri:oozie:email-action:0.1">
<to>some-dl@company.com</to>
<subject>Job execution failed ${wf:id()}</subject>
<body>Job execution failed, error message: [${wf:errorMessage(wf:lastErrorNode())}]</body>
</email>
<ok to="fail" /> <error to="fail" />
</action>
```

Pig

65. How pig scripts are converted in to MapReduce jobs?

Pig creates 3 plans - Logical, Physical and MapReduce plan to transform a Pig Latin script to a set of MapReduce jobs. In each of these plans Pig tries to optimize the script and at the end produce a MapReduce job.

Logical plan

- Describes the logical operators that Pig will use to execute the script.
- Some optimizations are done on this plan. For e.g., filters are pushed up as possible in the logical plan.

Physical plan

- Pig produces a physical plan after optimizing the logical plan.
- This plan describes the physical operators Pig will use to execute the script, without reference to how they will be executed in MapReduce.
- The load, store functions and the actual paths that will be used have been resolved
- Pig identifies 3 important operators. For instance, *COGROUP* operator will be replaced by Local Rearrange, Global Rearrange, and Package.
- Local Rearrange is the operator Pig uses to prepare data for the shuffle by setting up the key.
- Global Rearrange is for the shuffle.
- Package sits in the reduce phase and directs records to the proper bag

MapReduce Plan

- Pig takes the physical plan and decides how it will place its operators into one or more MapReduce jobs.
- Pig will go through the physical plan looking for all operators that require a new reduce. This occurs anywhere there is a Local Rearrange, Global Rearrange, and Package.
- Pig will look for places to do physical optimizations. For Illustration, it looks for places the combiner can be used, and whether sorts can be avoided by including them as part of the sorting Hadoop does in the shuffle.
- When all the above steps are complete, Pig will produce a MapReduce plan.

66. When do you use pig and when do you use Hive?

Pig is well suited for data transformation jobs as Pig by nature is a Data Transformation Language.

Hive is well suited to be used by Data Analysts and Scientists for data analysis. Hive QL is more aligned with SQL and will be easy to use for anyone with database background.

67. What operations you can't do in pig that you can do in MapReduce?

We need MapReduce when we need very deep level and fine grained control on the way we want to process our data. Sometimes, it is not very convenient to express what we need exactly in terms of Pig and Hive queries.

When your data is hierarchical rather than row-based or if your data is highly unstructured standard MapReduce jobs can be very efficient.

With standard MapReduce you get full control to minimize the number of MapReduce jobs that your data processing flow requires, which translates into performance. But it requires more time to code and introduce changes.

68. What pig operator or clause violate the MapReduce convention that all the key value pairs for the same key must go to same Reducer?

Below 2 operators or clauses violates the MapReduce convention that all the key value pairs for the same key must go the same Reducer.

1. USING SKEWED
2. ORDER BY

69. How do you compare 2 datasets?

Using JOIN operation on the datasets and using the comparison columns as the key columns

COGROUP can also be used giving the columns we are trying to compare

Illustration

```
grunt> joinbysymboldate = JOIN price_table by (symbol, date) , dividend_table by (symbol, date);
grunt> cgrp = COGROUP price_table BY (symbol, date), dividend_table by (symbol, date);
```

Below sample output show 3 rows. Only 2nd row had matching columns on both datasets.

```
((CSL,2009-05-14),{},{(NYSE,CSL,2009-05-14,0.155)}) ((CSL,2009-08-12),{(NYSE,CSL,2009-08-12,32.65,32.73,32.17,32.54,528900,32.39)},{(NYSE,CSL,2009-08-12,0.16)}) ((CSL,2009-08-13),{(NYSE,CSL,2009-08-13,32.58,33.19,32.49,33.15,447600,32.99)},{})
```

70. What happens behind the scenes in terms of MapReduce operation when you do a JOIN operation in Pig?

- In the Map phase, the input files are read separately by individual Map function and records from each input are tagged to indicate the input it came from.
- Join key(s) is used in the shuffle so that all records with the same join key goes to the same Reducer
- All records from the left tables are sent to the reducer first and cached in memory
- All records from the rightmost table are streamed to the reducer.
- As each of records from right most records arrives, it is crossed with each record from the left side to produce an output record.
- Reducer execute the join criteria on records as records from the rightmost table come through

71. How do you efficiently join 2 datasets in Pig?

In a regular join, records from the left hand side tables are cached in the reducer. Records from the right most table are streamed to the reducer. So make sure that the table on the right hand side is the biggest table.

72. How do you efficiently join two datasets which are already sorted?

The column which is used to sort the datasets is very important in this case and if your input datasets are already sorted on the join key the join can be done in the map phase. In Pig this is referred to as the merge join and can be implemented using the hint *using 'merge'* in your join instruction. This join is very efficient because entire shuffle and reduce phase is avoided because the data is already sorted.

Illustration

jnd = join table1 by cola, table2 by colb using 'merge';

73. How do you efficiently join two datasets if once dataset is big and one dataset is small?

If you have tables which are really smaller in size and can fit in memory, the entire reduce phase can be avoided using the *USING REPLICATED* hint -

- Entire Join is done at Map side and Reduce phase is ignored
- All right most tables are loaded in to memory on the nodes running the Mapper using DistributedCache
- Left most table is streamed to the mapper
- *USING REPLICATED* hint supports INNER & LEFT join
- RIGHT JOIN is not supported

74. Assume you are doing a join and you notice that all but one reducer is running for a long time how do you address the problem in Pig?

Pig collects all of the records for a given key together on a single reducer. In many data sets, there are a few keys that have three or more orders of magnitude more records than other keys. This results in one or two reducers that will take much longer than the rest. To deal with this, Pig provides skew join.

- In the first MapReduce job pig scans the second input and identifies keys that have so many records.
- In the second MapReduce job, it does the actual join.
- For all except the records with the key(s) identified from the first job, pig would do a standard join.
- For the records with keys identified by the second job, bases on how many records were seen for a given key, those records will be split across appropriate number of reducers.
- The other input to the join that is not split, only the keys in question are then split and then replicated to each reducer that contains that key

Illustration

jnd = join cinfo by city, users by city using 'skewed';

75. How do you achieve a Join without using JOIN operator in Pig?

COGROUP does first half of a join. The keys are collected together, but the cross product is not done. COGROUP combined with a FOREACH and flattening the resulting bag is equivalent to a join.

COGROUP + FOREACH = Join

Illustration

```
A = load 'input1' as (id:int, val:float);
B = load 'input2' as (id:int, val2:int);
C = cogroup A by id, B by id;
describe C;
C: {group: int,A: {id: int,val: float},B: {id: int,val2: int}}
```

76. How do you achieve a SEMI join in Pig?

Pig Latin does not have a SEMI JOIN operator. COGROUP can be used to perform a SEMI join. Look for questions under Hive to understand what is a SEMI JOIN.

Illustration

```
daily = load 'NYSE_daily' as (exchange:chararray, symbol:chararray, date:chararray, open:float, high:float, low:float, close:float, volume:int, adj_close:float);
divs = load 'NYSE_dividends' as (exchange:chararray, symbol:chararray, date:chararray, dividends:float);
grpd = cogroup daily by (exchange, symbol), divs by (exchange, symbol);
sjnd = filter grpd by not IsEmpty(divs);
final = foreach sjnd generate flatten(daily);
```

77. How do you perform non equi join in PIG?

Pig's join operator supports only equi-joins, that is, joins on an equality condition. Because general join implementations in MapReduce depend on collecting records with the same join key values onto the same reducer. Where as a Non-equi-join would require collecting records that does not have the same join key values and it is difficult to do.

In Pig you can achieve a non equi join using cross followed by filter.

Illustration

```
crossed = cross table1, table2 nonequi = filter crossed by table1::date < table2::date;
```

78. How do you calculate the number of lines in a file in PIG?

Illustration

```
LOGS= LOAD 'log';
LOGS_GROUP= GROUP LOGS ALL;
LOG_COUNT = FOREACH LOGS_GROUP GENERATE COUNT(LOGS);
```

79. How do you LOAD a sequence file in PIG?

Pig comes with a LOAD function to load Sequence Files – *SequenceFileLoader*. Illustration

```
REGISTER /home/ubuntu/pig-0.12.0/contrib/piggybank/java/piggybank.jar; DEFINE SequenceFileLoader
org.apache.pig.piggybank.storage.SequenceFileLoader(); seqdataset = LOAD 'input/ sequencefile' USING SequenceFileLoader AS
(key:long, data:chararray);
```

80. How do you register a UDF in Pig?

Use the REGITER instruction.

To locate the jar file, Pig first checks the classpath. If the jar file can't be found in the classpath, Pig assumes that the location is either an absolute path or a path relative to the location from which Pig was invoked. If the jar file can't be found, an error will be printed: java.io.IOException: Can't read jar file: myudfs.jar.

Illustration

```
REGISTER myudfs.jar;
```

81. How do you store the output into a Sequence File in Pig?

There is no STORE function natively available in Pig to do so. Elephand Bird (Open Source) initiative from Twitter has functions to store a Sequence File.

Hive

82. What is a SerDe?

SerDe is a short name for Serializer and Deserializer. The SerDe interface allows users to instruct Hive as to how a record should be processed. The Deserializer takes a string or binary representation of a record, and translates it into a Java object that Hive can manipulate. The Serializer, however, will take a Java object that Hive has been working with, and turn it into something that Hive can write to HDFS or another supported system. SerDe is commonly used to map data with complex structures to a row in Hive tables.

Below illustration gives the position of Serializer and Deserializer during write and read operation.

Row object --> **Serializer** --> <key, value> --> OutputFileFormat --> HDFS files

HDFS files --> InputFileFormat --> <key, value> --> **Deserializer** --> Row object

83. What is the functionality of ObjectInspector in Hive?

ObjectInspector is used by SerDe during Input and Output processing of the dataset. When the data is read from the dataset, the object inspector is used by the deserialize() method in SerDe to construct individual fields out of a deserialized record. When the data is written back to HDFS, the object inspector is used by the serialize() method in SerDe to get the individual fields in the record in order to convert the record to the appropriate type.

84. How will you use Hive to process datasets which are unstructured and semi-structured for e.g. dataset with email or JSON messages?

- First step is to use appropriate InputFormat to read the data. For e.g. if the file is a Sequence File use SequenceFileInputFormat
- Next step is to create a SerDe to instruct Hive as to how a record should be processed.
- Once the SerDe is created, create a table with the desired structure and use the SerDe in the table.

Below article from Cloudera has step by step instruction on how to create a SerDe to read a JSON dataset.

http://blog.cloudera.com/blog/2012/12/how-to-use-a-serde-in-apache-hive/

85. Explain how do you use case statement in hive select?

Illustration

```
SELECT exchange, price_open, price_close,
CASE
WHEN volume < 200000 THEN 'low'
WHEN volume >= 200000 AND volume < 400000 THEN 'middle'
WHEN volume >= 400000 AND volume < 600000 THEN 'high'
ELSE 'very high'
END AS volume_level FROM stocks;
```

86. What is the difference between SORT BY and ORDER BY in Hive?

ORDER BY performs a total ordering of the query result set. This means that all the data is passed through a single reducer, which may take an unacceptably long time to execute for larger data sets.

SORT BY orders the data only within each reducer, thereby performing a local ordering, where each reducer's output will be sorted. You will not achieve a total ordering on the dataset. Better performance is traded for total ordering.

87. What is DISTRIBUTE BY in Hive?

DISTRIBUTE BY controls how map output is divided among reducers.
DISTRIBUTE BY is used along with *SORT BY* and it is used to ensure that the records with the same value for the specified column will go to the same reducer, and then use *SORT BY* to order the data the way we want.
Illustration
With the below query all records with symbol AMZN (for e.g.) will be sent to the same reducer and the records will then be sorted by the ymd column.
hive> SELECT s.ymd, s.symbol, s.price_close FROM stock_quotes s DISTRIBUTE BY s.symbol SORT BY s.symbol ASC, s.ymd ASC;

88. How do you perform non equi join in Hive?

Just like Pig, Hive does not support non equi join for the same reason that it is difficult to implement non equi joins in MapReduce.
Follow https://issues.apache.org/jira/browse/HIVE-3133 for any developments on the non equi join implementation in Hive

89. What is LEFT SEMI JOIN?

A **is LEFT SEMI JOIN** returns records from the left hand table if records are found in the right hand side table that satisfy the ON predicates. It's a special, optimized case of the more general inner join. Most SQL dialects support an IN ... EXISTS construct to do the same thing.

90. Why LEFT SEMI JOIN is more efficient than INNER JOIN?

In a LEFT SEMI JOIN, for a given record in the left hand table, Hive can stop looking for matching records in the right hand table as soon as any match is found. At that point, the selected columns from the left hand table record can be projected.

91. How do you perform a LEFT SEMI JOIN in PIG and HIVE?

Illustration - Hive
SELECT s.ymd, s.symbol, s.price_close FROM stocks s LEFT SEMI JOIN dividends d ON s.ymd = d.ymd AND s.symbol = d.symbol;
Illustration - Pig
Pig does not have a SEMI join like Hive. But SEMI join can be achieved by using COGROUP
daily = load 'NYSE_daily' as (exchange:chararray, symbol:chararray, date:chararray, open:float, high:float, low:float, close:float, volume:int, adj_close:float); divs = load 'NYSE_dividends' as (exchange:chararray, symbol:chararray, date:chararray, dividends:float); grpd = cogroup daily by (exchange, symbol), divs by (exchange, symbol); sjnd = filter grpd by not IsEmpty(divs); final = foreach sjnd generate flatten(daily);

92. What is an alternative for using IN clause in Hive?

Below query will not work in Hive.
SELECT s.ymd, s.symbol, s.price_close FROM stocks s WHERE s.ymd, s.symbol IN (SELECT d.ymd, d.symbol FROM dividends d);
To achieve the above query use LEFT SEMI JOIN
SELECT s.ymd, s.symbol, s.price_close FROM stocks s LEFT SEMI JOIN dividends d ON s.ymd = d.ymd AND s.symbol = d.symbol;

93. How do you do minus operation between 2 datasets?

The Oracle MINUS operator is used to return all rows in the first SELECT statement that are not returned in the second SELECT statement.
There is no direct MINUS operator in Pig or Hive, instead you can achieve this by LEFT JOIN and do a filter out records from the second table which are null.
Illustration
SELECT t1.* FROM table1 t1 LEFT OUTER JOIN table2 t2 ON t1.col = t2.col WHERE t2.col IS NULL

94. What is a Bucket Map join?

When all the below conditions are met we can perform a Bucket Map join and it will be performed at the map side.
- All join tables are bucketized, and each small table's number of buckets can be divided by big table's number of buckets.
- Bucket columns and Join columns are same

Below property should be set to true to enable this feature.
set hive.optimize.bucketmapjoin = true;

95. What is a Sort Merge Bucket Map Join?

When all the below conditions are met we can perform a Sort Merge Bucket Map Join and it will be performed at the map side.
- All join tables are bucketized, and each small table's number of buckets can be divided by big table's number of buckets.
- 2.Bucket columns and Join columns are same ➢ Join columns are sorted

Below properties should be set to enable this feature.
set hive.optimize.bucketmapjoin = true; set hive.optimize.bucketmapjoin.sortedmerge = true; set hive.input.format=org.apache.hadoop.hive.ql.io.BucketizedHiveInputFormat;

96. How do you calculate the number of lines in a file in Hive?

hive> select count(*) from stocks;

97. How does partitioning a table in Hive improve performance?

Partitioning a Hive table with the right partition column has performance benefits. Partitions are used for distributing data horizontally and help in organizing data in a logical fashion. For example if you have a very big USERS table and your users in the table are spread across different states in the country then you can partition the table by country and state.

Partitioning tables changes how Hive structures the data storage and Hive will now create subdirectories reflecting the partitioning structure like .../users/country=ABC/state=XYZ

So next time when a query is issued against the table with country=ABC and state=XYZ the MapReduce job will target only that specific subdirectory.

98. What is the need for Buckets when there is already a way to distribute the data in Hive using Partitions?

Partitioning a Hive table with the right partition column has performance benefits however a table design that creates too many partitions may result in unbalanced partitions that would be result in unbalanced MapReduce execution. Other drawback is many small partitions create an overhead to NameNode since it has to manage the metadata for many files and blocks now.

You can design a table to use country for the top-level partition and state as the bucketing column, the value of this column will be hashed by a user-defined number into buckets. Users from the same state will always be stored in the same bucket. Assuming the number of states is much greater than the number of buckets, each bucket will have users from many states. With bucketing the number of buckets is fixed so it does not fluctuate with data. Bucketing is also used in doing efficient map-side joins.

Illustration

```
CREATE TABLE usersuser_id BIGINT, firstname STRING, lastname STRING)
PARTITIONED BY(country STRING)
CLUSTERED BY(state) INTO 25 BUCKETS;
```

99. If your table is partitioned with 2 partition keys and the no of buckets is set to 5 how many buckets you would see in each partition?

Dataset or records will be bucketed only at the end of the partition tree. First the table will be partitioned by the 2 partition keys and then the data will be divided and stored in 5 buckets.

/partition1 /partition2 /bucket1 /bucket2 /bucket3 /bucket4 /bucket5

100. What is an RCFile?

RCFile (Record Columnar File) is a file format in Hadoop. A table stored in RCFile is first horizontally partitioned into multiple row groups. Then, each row group is vertically partitioned so that each column is stored independently.

Partitioning table horizontally first makes sure that all the columns for a given row are in the same HDFS block. Vertically partition groups all column values of columns together and it gives the advantages of any column oriented store like faster query performance on column level aggregation, efficient compression at column level.

101. Are random writes possible with RCFile?

RCFile does not allow arbitrary data writing operations. RCFile creates and maintains an in-memory column holder for each column. When a record is appended, all its fields will be scattered, and each field will be appended into its corresponding column holder. In addition, RCFile will record corresponding metadata of each field in the metadata header.

RCFile provides two parameters to control how many records can be buffered in memory before they are flushed into the disk. One parameter is the limit of the number of records, and the other parameter is the limit of the size of the memory buffer.

102. How do you register UDF in Hive?

Once your UDF is ready, package it in a jar and add it to the Hive classpath. Once Hive is started up with your jars in the classpath, the final step is to register your function as described in Create Function.

Illustration

```
CREATE TEMPORARY FUNCTION my_func AS 'com.example.TempFunc';
```

To add jar to the classpath for all jobs initiated from that session.

Illustration

```
hive> add jar /tmp/my_jar.jar;
Added /tmp/my_jar.jar to class path
hive> list jars;
my_jar.jar
```

103. Assume you have a sales table in a company and it has sales entries from salesman around the globe. How do you rank each salesperson by country based on their sales volume in Hive?

Hive support several analytic functions and one of the functions is *RANK*() and it is designed to do this operation.

Lookup details on other window and analytic functions -
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+WindowingAndAnalytics

Illustration

```
Hive>SELECT rep_name, rep_country, sales_volume, rank() over (PARTITION BY rep_country ORDER BY sales_volume DESC)
as rank FROM salesrep;
```

104. MySQL GROUP_CONCAT() function returns a string with concatenated non-NULL value from a group. What is a similar function in Hive?

MySQL has a useful function known as GROUP_CONCAT, which combines all the elements of a group into a single string using a user-specified delimiter.

Hive already has a UDAF called COLLECT_SET that adds all input but it removes duplicates because it internally uses *java.util.Set* collection to store the collected values. Set automatically remove duplicate entries.

If we want to keep the duplicate values we need to create a UDAF and replace the instances of Set with instances of *ArrayList*. The result of the aggregate will be a single array of all values.

105. What are the different types of functions in Hive?

- User Defined Function (UDF)
- User Defined Aggregate Function (UDAF). UDAF takes in multiple values, perform aggregate operations and can return arrays or structures. For e.g. SUM (..)
- User-Defined Table Generating Functions (UDTF). UDFs cannot return multiple columns or multiple rows. UDTF address this issue and return multiple columns and even multiple rows.

106. How can you do compression at the field level?

In Hive compression can be enabled at intermediate Map Output and Output files. However in HBASE you can enable compression at the column family level.

Illustration

```
hbase> disable 'test'
hbase> alter 'test', {NAME => 'cf', COMPRESSION => 'GZ'}
hbase> enable 'test'
```

107. How do you delete records in Hive?

Data in Hive is stored as files behind the scenes so there is no concept of delete in Hive. Instead the dataset behind the table or partitions can be overwritten using INSERT OVERWRITE

108. Since compressed data involves decompression during processing, how does this increase performance?

Compressed files in HDFS help in reducing the amount of needed disk space. But compression does not allow splitting which is a bad thing for MapReduce jobs. However this can be overcome by compression using Sequence files.

But enabling intermediate map output compression can be of huge benefit and will certainly overcome the overhead of decompressing the data at the reduce side.

Assume you have 100 mappers and 1 reducer. Now reducer is trying to bring the data from 100 mappers as they are getting completed. If the size of the data is small the data transfer rate over the network would be higher and will result in a performance boost.

Also, enabling map output compression benefits not only the individual job's performance but also the overall cluster performance in terms of network utilization when compressed data is transferred between maps and reduce.

109. Let's say you are deploying a jar and you have a 100 node cluster. Do you have to deploy the jar to all 100 nodes?

No. When you execute a MapReduce program using *hadoop jar* command the client will upload the jar and needed files into HDFS. Then the jars will be used across all the nodes since all the needed files are on HDFS.

110. If you have only 10 map slots in your cluster, how does a job with 15 mappers run in the cluster?

If all 10 map slots are available the job will execute in 2 waves. In the first wave 10 mappers will run and in the next wave the remaining 5 mappers will run.

Sqoop

111. How is data ingested into your Hadoop cluster?

Data is brought in to the cluster from several data points - Database and files from other systems and files from outside vendors (in some cases).

Sqoop is used if the data is ingested from the database into HDFS In case of files, Shell Scripts can be used to copy the files into HDFS Flume will be used if you are bringing in Application log files into HDFS

112. From where (which node or server) the data gets ingested into the Hadoop cluster?

The question is if you are uploading a file into HDFS from which node you would perform the *copyFromLocal* or *put* operation. More specifically will you perform the *copyFromLocal* or *put* operation from one of your nodes in the cluster or from a node outside of the cluster.

It is also safe to perform the data ingestion process from a node outside of the cluster. This node is called the EDGE node. This is a good practice for couple of reasons.

- When uploading a file you can make sure the file is good and clean (not a virus etc) before you bring the file in to the cluster
- Using a EDGE node will avoid any unnecessary load on the cluster that may incur during the ingestion process

113. How do you control the number of mappers in Sqoop Import?

Using –m <n> or --num-mappers <n> in your Sqoop import command. The default number is 4.
Illustration
sqoop import --connect jdbc:mysql://mysql.address.internal/hadoop --table stocks **-m 2** --target-dir /user/temp/stocks

114. How do you import compressed data using Sqoop?

Sqoop import command takes compression arguments to enable the compression and specify the codec.
Use -z,--compress to enable compression and --compression-codec <c> to specify a codec (default gzip)
Illustration
sqoop import --connect jdbc:mysql://mysql.address.internal/hadoop --table stocks **--compress**
--target-dir /user/temp/stocks

115. How do you bring in only the recent data using Sqoop?

Sqoop incremental import option can be used for this. Incremental import works in 2 modes - *append* and *lastmodified* and both relies on the column mentioned in --check-column to be examine when determining which rows to import.

- append - use this when importing a table where new rows are continually being added with increasing row id values.
- lastmodified - use this when rows of the source table may be updated, and each such update will set the value of a last-modified column to the current timestamp.

Illustration
sqoop import --connect jdbc:mysql:// mysql.address.internal/hadoop --table stocks --target-dir /user/temp/stcoks_increment --incremental append --check-column id

116. How does the state of the last import gets stored for subsequent imports when using incremental import?

--last-value should be mentioned with the value from the previous import so that the current import will pull only the records greater than the value mentioned in --last-value
At the end of an incremental import, the value which should be specified as --last-value for a subsequent import is printed to the screen. When running a subsequent import, you should specify --last-value in this way to ensure you import only the new or updated data. This is handled automatically by creating an incremental import as a saved job, which is the preferred mechanism for performing a recurring incremental import.
Illustration
--To create sqoop job
sqoop job --create incrementalImportJob -- import --connect jdbc:mysql://mysql.address.internal /hadoop --table stocks --target-dir /user/temp/widgets_job --incremental append --check-column id
--To list sqoop jobs
sqoop job –list
--To show details of sqoop job
sqoop job --show incrementalImportJob
--To execute sqoop job
sqoop job --exec incrementalImportJob

117. How do you only keep the most recent updated data when using incremental load using lastmodified mode?

When you use the lastmodified option in your incremental import, sqoop will import records if the record is updated between imports. This means that sqoop will import the same row during different imports. So how can we make sure we only keep the latest row and ignore the rest.
sqoop-merge allows you to combine two datasets where entries in one dataset should overwrite entries of an older dataset. For Illustration, an incremental import run in last-modified mode will generate multiple datasets in HDFS where successively newer data appears in each dataset. The merge tool will "flatten" two datasets into one, taking the newest available records for each primary key.

118. How can you override the split column used by Sqoop import operation?

You can also explicitly choose a different column with the *--split-by* argument. If split-by is not mentioned Sqoop will use the primary key (if present) to split data between mappers.
Illustration
sqoop import --query 'SELECT a.*, b.* FROM a JOIN b on (a.id == b.id) WHERE $CONDITIONS'
--split-by a.id --target-dir /user/foo/joinresults

119. How you imported data from AS400 to Sqoop?

The question is to ask what databases are supported by Sqoop. You can use Sqoop with any other JDBC-compliant database. First, download the appropriate JDBC driver for the type of database you want to import, and install the .jar file in the $SQOOP_HOME/lib directory on your client machine.

Flume

120. What are the key components involved in a Flume agent?

- Source - A Flume source consumes events delivered to it by an external source like a web server.
- Channel - When a Flume source receives an event, it stores it into one or more channels. The channel is a passive store that keeps the event until it's consumed by a Flume sink.
- Sink - The sink removes the event from the channel and puts it into an external repository like HDFS (via Flume HDFS sink) or forwards it to the Flume source of the next Flume agent (next hop) in the flow.

Illustration

Below Flume agent configuration is to define a source (log file), channel and sink (file in local file system)

```
# Flume Components
agent.channels = memory-channel
agent.sources = tail-source
agent.sinks = local-file-sink
# Define a memory channel on agent called memory-channel.
agent.channels.memory-channel.type = memory
# Define a source on agent and connect to channel memory-channel.
agent.sources.tail-source.type = exec
agent.sources.tail-source.command = tail -F /home/temp/logfile.log
agent.sources.tail-source.channels = memory-channel
# Define a sink that outputs to local file system.
agent.sinks.local-file-sink.type = file_roll
agent.sinks.local-file-sink.sink.directory = flume/local-file
agent.sinks.local-file-sink.sink.rollInterval = 60
agent.sinks.local-file-sink.channel = memory-channel
```

121. What is replication in Flume?

In a replication setup, there are multiple channels between the source and the sink. Each Flume event from the source is sent to all the channels. The sink then picks up the events from the channel it is paired with.

Illustration

Below Flume agent configuration is to define a source, 2 channels and 2 sinks (file in local file system and file in HDFS). All the events from the source will be sent to both the channels.

```
# Flume Components
agent.channels = memory-channel-local memory-channel-hdfs
agent.sources = tail-source
agent.sinks = local-sink hdfs-sink
# Define a memory channel on agent called memory-channel.
agent.channels.memory-channel-local.type = memory
agent.channels.memory-channel-hdfs.type = memory
# Define a source on agent and connect to channel memory-channel.
agent.sources.tail-source.type = exec
agent.sources.tail-source.command = tail -F /home/temp/logfile.log
agent.sources.tail-source.channels = memory-channel-local memory-channel-hdfs
# Define a sink that outputs to local file.
agent.sinks.local-sink.type = file_roll
agent.sinks.local-sink.sink.directory = flume/local-file
agent.sinks.local-sink.sink.rollInterval = 60
agent.sinks.local-sink.channel = memory-channel-local
# Define a sink that outputs to hdfs.
agent.sinks.hdfs-sink.type = hdfs
agent.sinks.hdfs-sink.hdfs.path = flume/replicate
agent.sinks.hdfs-sink.hdfs.fileType = DataStream
agent.sinks.hdfs-sink.hdfs.rollCount = 5
agent.sinks.hdfs-sink.hdfs.inUseSuffix = .tmp
agent.sinks.hdfs-sink.channel = memory-channel-hdfs
```

122. What is multiplexing in Flume?

A flume event can be sent to a subset of available channels when an event's attribute matches a preconfigured value.
Illustration
In this illustration, if an event attribute called *location* is set to *newyork*, then it should go to *memory-channel-newyork* and, if it's *chicago* then it should go to *memory-channel-chicago* and *memory-channel-default*
Below excerpt from the flume configuration file demonstrate that.
agent.sources.avro-collection-source.selector.type = multiplexing
agent.sources.avro-collection-source.selector.header = location
agent.sources.avro-collection-source.selector.mapping.newyork = memory-channel-newyork
agent.sources.avro-collection-source.selector.mapping.chicago = memory-channel-chicago memory-channel-default

THE BASICS

123. What is Big Data?

The term Big Data refers to extremely large data sets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions. It is used to describe a massive volume of both structured and unstructured data that is so large it is difficult to process using traditional database and software techniques. In most enterprise scenarios the volume of data is too big or it moves too fast or it exceeds current processing capacity.
What is considered "huge" – 10 GB, 100 GB or 100 TB?
Well, there is no straight hard number that defines "huge" or Big Data. There are 2 reasons why there is no straight forward answer - First, what is considered Big or huge today in terms is data size or volume need not be considered as Big a year from now. It is very much a moving target.
Second, it is all relative. What you and I consider to be "Big" may not be the case for companies like Google and Facebook.
Hence for the above 2 reasons, it is very hard to put a number to define big data volume. Let's say if we are defining Big Data problem in terms of pure volume alone then in our opinion –

- 100 GB – Not a chance. We all have hard disk greater than 10 GB. Clearly not Big Data.
- 1 TB – Still no Because a well defined traditional database can handle 1 TB or even more without any issues.
- 100 TB – Likely. Some would claim 100 TB to be a Big Data problem and others might disagree. Again, its all relative.
- 1000 TB – Big Data problem.

Recommended Reading - https://en.wikipedia.org/wiki/Big_data

124. What are the characteristics of Big Data?

Gartner, and now much of the industry, continue to use this "3Vs" model for describing big data.

- **Volume** – refers to the sheer quantity of data that an organization or enterprise is expected to manage to perform quality analysis to derive business decisions.
- **Velocity** - refers to the speed of generation of data or how fast the data is generated and expected to be processed and analyzed.
- **Variety** - refers to the structure and content in which the data is presented. For eg. text, binary, images, video etc. Additionally, a new V "Veracity" is added by some organizations to describe Big Data.
- **Veracity** - The quality of the data being captured can vary greatly. Accuracy of analysis depends on the veracity of the source data.

125. What is Hadoop framework?

Apache Hadoop is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware.

126. What are the core components of Hadoop?

The base Apache Hadoop framework is composed of the following modules: ➢ Hadoop Common – contains libraries and utilities needed by other Hadoop modules; ➢ Hadoop Distributed File System (HDFS) – a distributed file-system that stores data on commodity servers ➢ Hadoop YARN – a resource-management platform responsible for managing computing resources Hadoop ➢ MapReduce – a distributed programming model for large scale data processing.

127. What is MapReduce ?

MapReduce is a framework for processing parallelizable problems across huge datasets using a large number of computers. The name MapReduce originally referred to the proprietary Google technology, but has since been generalized. Apache Hadoop offers an open source implementation of MapReduce.

128. What are the phases in MapReduce?

- **Map Phase -** Worker node execute the map() function to all the records in a subset of the whole dataset and emit a key value pair
- **Shuffle Phase -** Worker nodes redistribute key value pairs produced by the map() functions, sort and copy them to the Reduce phase
- **Reduce Phase -**Worker nodes now execute the reduce() on each group of key and its values, per key, in parallel.

129. Explain Shuffle process

The process in which the Map outputs are sorted and transferred to the corresponding reducers as inputs is known as the shuffle.

- Map Side

Map output is divided into partitions corresponding to the reducers that they will ultimately be sent to. Within each partition, the map output is sorted by key, and if there is a combiner function, it is run on the output of the sort.

- Reduce Side

In the Copy phase, the reduce task starts copying their corresponding Map outputs as soon as each completes. When all the map outputs have been copied, the reduce task merges the map outputs, maintaining their sort ordering and this is referred to as the sort phase.

During the reduce phase, the reduce function is invoked for each key in the sorted output.

130. What is a Combiner?

The primary goal of combiners is to optimize/minimize the number of key value pairs that will be shuffled accross the network between mappers and reducers and thus to save as most bandwidth as possible. Usage of the Combiner is optional. If this pass is suitable for your job, instances of the Combiner class are run on every node that has run map tasks. The Combiner will receive as input all data emitted by the Mapper instances on a given node. The output from the Combiner is then sent to the Reducers, instead of the output from the Mappers. The Combiner is a "mini-reduce" process which operates only on data generated by one machine.

131. What is JobTracker?

Job Tracker service is responsible for the coordination of MapReduce tasks in a Hadoop cluster. Following are the actions performed by the Job Tracker during a MapReduce execution.

1. Client applications submit jobs to the Job tracker.
2. The JobTracker talks to the NameNode to determine the location of the data
3. The JobTracker locates TaskTracker nodes with available slots at or near the data
4. The JobTracker submits the work to the chosen TaskTracker nodes.
5. The TaskTracker nodes are monitored. If they do not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different TaskTracker.
6. A TaskTracker will notify the JobTracker when a task fails. The JobTracker decides what to do then: it may resubmit the job elsewhere, it may mark that specific record as something to avoid, and it may may even blacklist the TaskTracker as unreliable.
7. When the work is completed, the JobTracker updates its status.
8. Client applications can poll the JobTracker for information.

In Hadoop version 2 Job Tracker is replaced by Resource Manager and Application Master Reference:
http://wiki.apache.org/hadoop/JobTracker

132. What is TaskTracker?

A TaskTracker is a node in the cluster that accepts tasks - Map, Reduce and Shuffle operations - from a JobTracker.

In Hadoop version 2, Task Tracker is replaced by Node Manager

133. Explain HDFS write operation

- Client will request NameNode to create a file. NameNode performs various checks to make sure the file doesn't already exist, and that the client has the right permissions to create the file.
- Client will ask the NameNode to allocate new blocks by picking a list of suitable DataNodes to store the replicas.
- Data packets are sent to the first DataNode which stores the packet and forwards it to the second DataNode and then to the 3rd DataNode.

134. How are failures handled during HDFS write operation?

- During write data packets are sent to the first DataNode which stores the packet and forwards it to the second DataNode and then to the 3rd DataNode.
- If the write to the 2nd DataNode failed, the data will still be written to the 3rd DataNode.
- The current block on the good DataNodes is given a new identity. So that the partial block on the failed 2nd DataNode will be deleted if the failed DataNode recovers later on.
- NameNode notices that the block is under-replicated, and it arranges for a further replica to be created on another node.

135. How do you copy a file from local file system to Hadoop cluster?

copyFromLocal command can be used to copy a file from local file system to HDFS

Illustration

$ hdfs dfs -copyFromLocal <localsrc> <HDFS URI>

136. What are the different modes you can run Hadoop?

- Local (Standalone) Mode Hadoop is configured to run in a non-distributed mode, as a single Java process. This is useful for debugging.
- Pseudo-Distributed Mode Hadoop can also be run on a single-node in a pseudo-distributed mode where each Hadoop daemon runs in a separate Java process.
- Fully-Distributed Mode This is a fully distributed set up in which typically one machine in the cluster is designated as the NameNode and another machine the as JobTracker (ResourceManager in Hadoop 2) exclusively. These nodes are referred as master nodes. The remaining nodes in the cluster act as both DataNode *and* TaskTracker (NodeManager in Hadoop 2) and they are referred to as slave nodes.

137. What are the parameters of mappers and reducers function?

Map and Reduce method signature tells you a lot about the type of input and ouput your Job will deal with. Assuming you are using TextInputFormat, Map function's parameters could look like -
LongWritable (Input Key)
Text (Input Value)
Text (Intermediate Key Output)
IntWritable (Intermediate Output)
The four parameters for reduce function could be -
Text (Intermediate Key Output)
IntWritable (Intermediate Value Output)
Text (Final Key Output)
IntWritable (Final Value Output)

138. What is InputSplit and how it is different from a Block?

Input splits are a logical division of your records whereas HDFS blocks are a physical division of the input data. You could have a record that started in one block and ends in another block. Block is a physical division of data and does not take in to account the logical boundary of records whereas InputSplit considers the logical boundaries of records as well.

139. What is Speculative execution?

A job running on a Hadoop cluster could be divided into many tasks. In a big cluster some of these tasks could be running slow for various reasons, hardware degradation or software misconfiguration etc. Hadoop initiates a replica of a task when it sees a task which is running for some time and failed to make any progress, on average, as the other tasks from the job. This replica or duplicate execution of task is referred to as Speculative Execution.
When a task completes successfully all the duplicate tasks that are running will be killed. So if the original task completes before the speculative task, then the speculative task is killed; on the other hand, if the speculative task finishes first, then the original is killed.

140. What is distributed cache?

Distributed Cache is a facility provided by the Map-Reduce framework to cache files (text, archives, jars etc.) needed by applications. Applications specify the files with location to be cached via the *JobConf*. The *DistributedCache* assumes that the files specified via URLs are already present on the *FileSystem* at the path specified by the URL and are accessible by every machine in the cluster. Hadoop's MapReduce framework will copy the necessary files on to the slave node before any tasks for the job are executed on that node.
Illustration
In driver program
JobConf job = new JobConf();
DistributedCache.addCacheFile(new URI("hdfs://namenode:port/dir/smalldataset.txt"), job);
In Mapper or Reducer
File f = new File("./smalldataset.txt");

141. What is the benefit of using counters in Hadoop?

Counters are a useful for gathering details about the job in a centralized fashion. Assume you have a 100 node cluster and a job with 100 mappers is running in the cluster on 100 different nodes. Let's say you would like to know each time you see an invalid record in your Map phase. You could add a log message in your Mapper so that each time you see an invalid line you can make an entry in the log. But consolidating all the log messages from 100 different nodes will be time consuming. You can use a counter instead in your Map program and increment the value of the counter every time you see an invalid record. The nice thing about using counters is that is gives you a consolidate value for the whole job rather than showing 100 separate outputs. In this specific example you would see the number of input lines printed at the end of job.

142. What is Streaming?

Streaming allows you to create and run Map/Reduce jobs with any executable or script as the mapper and/or the reducer.
Illustration
$HADOOP_HOME/bin/hadoop jar $HADOOP_HOME/hadoop-streaming.jar \ -input myInputDirs \
-output myOutputDir \ -mapper /bin/cat \ -reducer /bin/wc
In the above example, both the mapper and the reducer are executables that read the input from stdin (line by line) and emit the output to stdout. The utility will create a Map/Reduce job, submit the job to an appropriate cluster, and monitor the progress of the job until it completes. Suggested Reading: https://hadoop.apache.org/docs/r1.2.1/streaming.html#Hadoop+Streaming

143. What is the basic difference between traditional RDBMS and Hadoop?

RDBMS
MapReduce
Volume
Gigabytes
Petabytes
Access
Interactive and Batch
Batch
Updates
Read & Write many times
Write once, read many times
Schema
Static
Dynamic
Scaling
Nonlinear
Linear

144. What is a ‘block’ in HDFS?

When a file or dataset is uploaded to HDFS, the dataset is divided in to fixed size chunks or blocks and placed across different nodes in the cluster.
The size of block is configurable. In older versions of Hadoop the size of the block was set to 64 MB by default in the later version it is moved up to 128 MB.
Storing the data in blocks help with parallel and distributed access of data and replication for fault tolerance.

145. What is a Replica Placement Strategy?

Replica placement defines how the blocks are stored in the cluster. By default the 1st replica is placed on the local machine, otherwise a random data node. The 2nd replica is placed on a data node that is on a different rack. The 3rd replica is placed on a data node which is on the same rack as the first replica.

146. What is the difference between Pig and Hive?

Both Pig and Hive scripts gets transformed into a one or more MapReduce jobs behind the scene.
Pig tends to create a flow of data - small steps where in each you do some processing. Hence it is ideal for data transformation tasks and a good fit for ETL kind of processing. Pig can be easier for someone who had not earlier experience with SQL.
Hive gives you SQL-like language to operate on your data, so transformation from RDBMS is much easier. Hive also gives you metastore for information about your data (tables, schema, etc) which is useful to store the stricter of your data. Storing the data in a structured way using Hive partitions and buckets can be very helpful in terms of performance.

147. What is Hive Metastore?

All the metadata for Hive tables and partitions are stored and accessed through the Hive Metastore.