Q1. Name the most common Input Formats defined in Hadoop? Which one is default?

The two most common Input Formats defined in Hadoop are:

– TextInputFormat

- KeyValueInputFormat

- SequenceFileInputFormat

TextInputFormat is the Hadoop default.

Q2. What is the difference between TextInputFormat and KeyValueInputFormat class?

TextInputFormat: It reads lines of text files and provides the offset of the line as key to the Mapper and actual line as Value to the mapper.

KeyValueInputFormat: Reads text file and parses lines into key, Val pairs. Everything up to the first tab character is sent as key to the Mapper and the remainder of the line is sent as value to the mapper.

Q3. What is InputSplit in Hadoop?

When a Hadoop job is run, it splits input files into chunks and assign each split to a mapper to process. This is called InputSplit.

Q4. How is the splitting of file invoked in Hadoop framework?

It is invoked by the Hadoop framework by running getInputSplit()method of the Input format class (like FileInputFormat) defined by the user.

Q5. Consider case scenario: In M/R system, - HDFS block size is 64 MB

- Input format is FileInputFormat

– We have 3 files of size 64K, 65Mb and 127Mb

How many input splits will be made by Hadoop framework?

Hadoop will make 5 splits as follows:

- 1 split for 64K files

- 2 splits for 65MB files

- 2 splits for 127MB files

Q6. What is the purpose of RecordReader in Hadoop?

The InputSplit has defined a slice of work, but does not describe how to access it. The RecordReader class actually loads the data from its source and converts it into (key, value) pairs suitable for reading by the Mapper. The RecordReader instance is defined by the Input Format.

Q7. After the Map phase finishes, the Hadoop framework does "Partitioning, Shuffle and sort". Explain what happens in this phase?

Partitioning: It is the process of determining which reducer instance will receive which intermediate keys and values. Each mapper must determine for all of its output (key, value) pairs

which reducer will receive them. It is necessary that for any key, regardless of which mapper instance generated it, the destination partition is the same.

Shuffle: After the first map tasks have completed, the nodes may still be performing several more map tasks each. But they also begin exchanging the intermediate outputs from the map tasks to where they are required by the reducers. This process of moving map outputs to the reducers is known as shuffling.

Sort: Each reduce task is responsible for reducing the values associated with several intermediate keys. The set of intermediate keys on a single node is automatically sorted by Hadoop before they are presented to the Reducer.

Q8. If no custom partitioner is defined in Hadoop then how is data partitioned before it is sent to the reducer?

The default partitioner computes a hash value for the key and assigns the partition based on this result.

Q9. What is a Combiner?

The Combiner is a ‘mini-reduce’ process which operates only on data generated by a mapper. The Combiner will receive as input all data emitted by the Mapper instances o

n a given node. The output from the Combiner is then sent to the Reducers, instead of the output from the Mappers.

Q10. What is JobTracker?

JobTracker is the service within Hadoop that runs MapReduce jobs on the cluster.

Q11. What are some typical functions of Job Tracker?

The following are some typical tasks of JobTracker:-

- Accepts jobs from clients

- It talks to the NameNode to determine the location of the data.

- It locates TaskTracker nodes with available slots at or near the data.

- It submits the work to the chosen TaskTracker nodes and monitors progress of each task by receiving heartbeat signals from Task tracker.

Q12. What is TaskTracker?

TaskTracker is a node in the cluster that accepts tasks like MapReduce and Shuffle operations – from a JobTracker.

Q13. What is the relationship between Jobs and Tasks in Hadoop?

One job is broken down into one or many tasks in Hadoop.

Q14. Suppose Hadoop spawned 100 tasks for a job and one of the task failed. What will Hadoop do?

It will restart the task again on some other TaskTracker and only if the task fails more than four

(default setting and can be changed) times will it kill the job.

Q15. Hadoop achieves parallelism by dividing the tasks across many nodes, it is possible for a few slow nodes to rate-limit the rest of the program and slow down the program. What mechanism Hadoop provides to combat this?

Speculative Execution.

Q16. How does speculative execution work in Hadoop?

JobTracker makes different TaskTrackers pr

ocess same input. When tasks complete, they announce this fact to the JobTracker. Whichever copy of a task finishes first becomes the definitive copy. If other copies were executing speculatively, Hadoop tells the TaskTrackers to abandon the tasks and discard their outputs. The Reducers then receive their inputs from whichever Mapper completed successfully, first.

Q17. Using command line in Linux, how will you

- See all jobs running in the Hadoop cluster

- Kill a job?

Hadoop job – list

Hadoop job – kill jobID

Q18. What is Hadoop Streaming?

Streaming is a generic API that allows programs written in virtually any language to be used as Hadoop Mapper and Reducer implementations.

Q19. What is the characteristic of streaming API that makes it flexible run MapReduce jobs in languages like Perl, Ruby, Awk etc.?

Hadoop Streaming allows to use arbitrary programs for the Mapper and Reducer phases of a MapReduce job by having both Mappers and Reducers receive their input on stdin and emit output (key, value) pairs on stdout.

Q20. What is Distributed Cache in Hadoop?

Distributed Cache is a facility provided by the MapReduce framework to cache files (text, archives, jars and so on) needed by applications during execution of the job. The framework will copy the necessary files to the slave node before any tasks for the job are executed on that node.

Q21. What is the benefit of Distributed cache? Why can we just have the file in HDFS and have the application read it?

This is because distributed cache is much faster. It copies the file to all trackers at the start of the job. Now if the task tracker runs 10 or 100 Mappers or Reducer, it will use the same copy of distributed cache. On the other hand, if you put code in file to read it from HDFS in the MR Job then every Mapper will try to access it from HDFS hence if a TaskTracker run 100 map jobs then it will try to read this file 100 times from HDFS. Also HDFS is not very efficient when used like this.

Q.22 What mechanism does Hadoop framework provide to synchronise changes made in

Distribution Cache during runtime of the application?

This is a tricky question. There is no such mechanism. Distributed Cache by design is read only during the time of Job execution.

Q23. Have you ever used Counters in Hadoop. Give us an example scenario?

Anybody who claims to have worked on a Hadoop project is expected to use counters.

Q24. Is it possible to provide multiple input to Hadoop? If yes then how can you give multiple directories as input to the Hadoop job?

Yes, the input format class provides methods to add multiple directories as input to a Hadoop job.

Q25. Is it possible to have Hadoop job output in multiple directories? If yes, how?

Yes, by using Multiple Outputs class.

Q26. What will a Hadoop job do if you try to run it with an output directory that is already present? Will it

- Overwrite it

- Warn you and continue

- Throw an exception and exit

The Hadoop job will throw an exception and exit.

Q27. How can you set an arbitrary number of mappers to be created for a job in Hadoop?

You cannot set it.

Q28. How can you set an arbitrary number of Reducers to be created for a job in Hadoop?

You can either do it programmatically by using method setNumReduceTasks in the Jobconf Class or set it up as a configuration setting.

Q29. How will you write a custom partitioner for a Hadoop job?

To have Hadoop use a custom partitioner you will have to do minimum the following three:

- Create a new class that extends Partitioner Class

- Override method getPartition

- In the wrapper that runs the Mapreduce, either

- Add the custom partitioner to the job programmatically using method set Partitioner Class or – add the custom partitioner to the job as a config file (if your wrapper reads from config file or oozie)

Q30. How did you debug your Hadoop code?

There can be several ways of doing this but most common ways are:-

- By using counters.

- The web interface provided by Hadoop framework.

Q31. Did you ever built a production process in Hadoop? If yes, what was the process when your Hadoop job fails due to any reason?

It is an open-ended question but most candidates if they have written a production job, should talk about some type of alert mechanism like email is sent or there monitoring system sends an alert. Since Hadoop works on unstructured data, it is very important to have a good alerting system for errors since unexpected data can very easily break the job.

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 in 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

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>
```

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

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 then split and then replicated to each reducer that contains that key

Illustration

jnd = join cinfo by city, users by city using ‘skewed’;

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.

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;
```

What is Speculative execution?

A job running on a Hadoop cluster could be divided in to many tasks. In a big cluster some of these tasks could be running slow for various reasons, hardware degradation or software miconfiguration etc. Hadoop initiates a replica of a task when it sees a tasks which is running for sometime and failed to make any progress, on average, as the other tasks from the job. This replica or duplicate exeuction 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.

What is the benefit of using counters in Hadoop?

Counters are a useful for gathering statistics about the job. Assume you have a 100 node cluster and a job with 100 mappers is running in the cluster on 100 different nodes. Lets say you would like to know each time you see a 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 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.

What is the difference between an InputSplit and a Block?

Block is a physical division of data and does not take in to account the logical boundary of records. Meaning you could have a record that started in one block and ends in another block. Where as InputSplit considers the logical boundaries of records as well.

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.

How do you do a file system check in HDFS?

FSCK command is used to do a file system check in HDFS. It is a very useful command to check the health of the file, block names and block locations.

Illustration

hdfs fsck /dir/hadoop-test -files -blocks -locations

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)

How do you overwrite replication factor?

There are few ways to do this. Look at the below illustration.

Illustration

hadoop fs -setrep -w 5 -R hadoop-test

hadoop fs -Ddfs.replication=5 -cp hadoop-test/test.csv hadoop-test/test_with_rep5.csv

What are the functions of InputFormat?

Validate input data is present and check input configuration
Create InputSplits 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.

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.

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.

Qs. What is Hadoop framework?

Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment.

- Hadoop is part of the Apache project sponsored by the Apache Software Foundation.
- Hadoop makes it possible to run applications on systems with thousands of nodes involving thousands of terabytes. Its distributed file system facilitates rapid data transfer rates among nodes and allows the system to continue operating uninterrupted in case of a node failure. This approach lowers the risk of catastrophic system failure, even if a significant number of nodes become inoperative.
- Hadoop framework is used by major players including Google, Yahoo and IBM, largely for applications involving search engines and advertising.
- The preferred operating systems are Windows and Linux but Hadoop can also work with BSD and OS X

Qs. What is MapReduce?
MapReduce is a parallel programming model which is used to process large data sets across hundreds or thousands of servers in a Hadoop cluster.

- Map/reduce brings compute to the data at data location in contrast to traditional parallelism, which brings data to the compute location.
- The Term MapReduce is composed of Map and Reduce phase. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job. he programming language for MapReduce is Java.All data emitted in the flow of a MapReduce program is in the form of Key/Value pairs.

Qs. Explain a MapReduce program
A MapReduce program consists of the following 3 parts :

- Driver
- Mapper
- Reducer

1. **The Driver** code runs on the client machine and is responsible for building the configuration of the job and submitting it to the Hadoop Cluster. The Driver code will contain the main() method that accepts arguments from the command line.
2. **The Mapper** code reads the input files as <Key,Value> pairs and emits key value pairs. The Mapper class extends MapReduceBase and implements the Mapper interface. The Mapper interface expects four generics, which define the types of the input and output key/value pairs. The first two parameters define the input key and value types, the second two define the output key and value types.
3. **The Reducer** code reads the outputs generated by the different mappers as <Key,Value> pairs and emits key value pairs. The Reducer class extends MapReduceBase and implements the Reducer interface. The Reducer interface expects four generics, which define the types of the input and output key/value pairs. The first two parameters define the intermediate key and value types, the second two define the final output key and value types. The keys are WritableComparables, the values are Writables

Qs. Which interface needs to be implemented to create Mapper and Reducer for the Hadoop?

1. org.apache.hadoop.mapreduce.Mapper
2. org.apache.hadoop.mapreduce.Reducer

Qs. What Mapper does?
Mapper is the first phase of Map phase which process map task.Mapper reads key/value pairs and emit key/value pair.

- Maps are the individual tasks that transform input records into intermediate records.
- The transformed intermediate records do not need to be of the same type as the input records. A given input pair may map to zero or many output pairs.

Qs. How many daemon processes run on a Hadoop cluster?
Hadoop is comprised of five separate daemons. Each of these daemons runs in its own JVM. Following 3 Daemons run on Master nodes.
NameNode - This daemon stores and maintains the metadata for HDFS.The namenode is the master server in Hadoop and manages the file system namespace and access to the files stored in the cluster.
Secondary NameNode -Secondary namenode, isn't a redundant daemon for the namenode but instead provides period checkpointing and housekeeping taskse.
JobTracker - Each cluster will have a single jobtracker that manages MapReduce jobs, distributes

individual tasks to machines running the Task Tracker.
Following 2 Daemons run on each Slave nodes
DataNode – Stores actual HDFS data blocks.The datanode manages the storage attached to a node, of which there can be multiple nodes in a cluster. Each node storing data will have a datanode daemon running.
TaskTracker – It is Responsible for instantiating and monitoring individual Map and Reduce tasks i.e.Tasktracker per datanode performs the actual work

Qs. What is InputSplit in Hadoop?
Input Split: It is part of input processed by a single map. Each split is processed by a single map. In other words InputSplit represents the data to be processed by an individual Mapper. Each split is divided into records , and the map processes each record, which is a key value pair. Split is basically a number of rows and record is that number.

- The length of the InputSplit is measured in bytes.
- Every InputSplit has a storage locations (hostname strings). The storage locations are used by the MapReduce system to place map tasks as close to split's data as possible.

For example if data is 128 MB and block size if 64 MB(default)

- Case 1 - Input split size [64 MB] = Block size [64 MB], # of Map task 2
- Case 2 - Input split size [32 MB] = Block size [64 MB], # of Map task 4
- Case 2 - Input split size [128 MB] = Block size [64 MB], # of Map task 1

Qs. What is the InputFormat?
The InputFormat class is one of the fundamental classes in the Hadoop Map Reduce framework. This class is responsible for defining two main things:

- Data splits
- Record reader

1. Data split is a fundamental concept in Hadoop Map Reduce framework which defines both the size of individual Map tasks and its potential execution server.
2. The Record Reader is responsible for actual reading records from the input file and submitting them (as key/value pairs) to the mapper.

Qs. Consider case scenario: In M/R system, - HDFS block size is 64 MB. Now Input format is FileInputFormat and we have 3 files of size 64K, 65Mb and 127Mb. How many input splits will be made by Hadoop framework?
Hadoop will make 5 splits as follows:

- 1 split for 64K files
- 2 splits for 65MB files
- 2 splits for 127MB files

Qs. What is JobTracker?
JobTracker is the service within Hadoop that runs MapReduce jobs on the cluster.

Qs. What if job tracker machine is down?
In Hadoop 1.0, Job Tracker is single Point of availability means if JobTracker fails, all jobs must restart.Overall Execution flow will be interupted. Due to this limitation, In hadoop 2.0 Job Tracker concept is replaced by YARN.

In YARN, the term JobTracker and TaskTracker has totally disappeared. YARN splits the two major functionalities of the JobTracker i.e. resource management and job scheduling/monitoring into 2 separate daemons (components).

- Resource Manager
- Node Manager(node specific)

Qs. What happens when a datanode fails ?
When a datanode fails:

- Jobtracker and namenode detect the failure
- On the failed node all tasks are re-scheduled
- Namenode replicates the users data to another node

Qs. What are some typical functions of Job Tracker?
The following are some typical tasks of JobTracker:-

When Client applications submit map reduce jobs to the Job tracker.

- The JobTracker talks to the Name node to determine the location of the data.
- The JobTracker locates Tasktracker nodes with available slots at or near the data
- The JobTracker submits the work to the chosen Tasktracker nodes.
- 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.
- When the work is completed, the JobTracker updates its status.
- Client applications can poll the JobTracker for information.

Qs. Mention what are the main configuration parameters that user need to specify to run MapReduce Job?
The user of Mapreduce framework needs to specify

- Job’s input locations in the distributed file system
- Job’s output location in the distributed file system
- Input format
- Output format
- Class containing the map function
- Class containing the reduce function

- JAR file containing the mapper, reducer and driver classes

Qs. What is TaskTracker?
TaskTracker is a node in the cluster that accepts tasks like MapReduce and Shuffle operations – from a JobTracker.

1. Each Task Tracker is responsible to execute and manage the individual tasks assigned by Job Tracker
2. Task Tracker also handles the data motion between the map and reduce phases.
3. One Prime responsibility of Task Tracker is to constantly communicate with the Job Tracker the status of the Task.
4. If the JobTracker fails to receive a heartbeat from a TaskTracker within a specified amount of time, it will assume the TaskTracker has crashed and will resubmit the corresponding tasks to other nodes in the cluster

Qs.Explain what is heartbeat in HDFS?
Heartbeat is referred to a signal used between a data node and Name node, and between task tracker and job tracker, if the Name node or job tracker does not respond to the signal, then it is considered there is some issues with data node or task tracker.
Qs. Explain what is sqoop in Hadoop?

Sqoop is a connectivity tool which transfers data in both directions between relational databases (MySQL, Oracle, Teradata) ,data warehouses and Hadoop HDFS and other Hadoop data source like Hive, HBase.

- Sqoop allows easy import and export of data from structured data stores.
- Sqoop integrates with Oozie, allowing you to schedule and automate import and export tasks.

Example: A simple use case will be an organization that runs a nightly sqoop import to load the day's data from a production DB into a Hive data ware house for analysis.
Qs. Suppose Hadoop spawned 100 tasks for a job and one of the task failed. What will Hadoop do?
It will restart the task again on some other TaskTracker and only if the task fails more than four (default setting and can be changed) times will it kill the job.

Qs. What is speculative execution (also called backup tasks)? What problem does it solve?
In Hadoop during **Speculative Execution** a certain number of duplicate tasks are launched.
On different slave node, multiple copies of same map or reduce task can be executed using Speculative Execution.
In simple words, if a particular drive is taking long time to complete a task, Hadoop will create a duplicate task on another disk.
Disk that finish the task first are retained and disks that do not finish first are killed.

Qs. Explain the use of TaskTracker in the Hadoop cluster?

1. A TaskTracker is a slave node in the cluster which that accepts the tasks from JobTracker like Map, Reduce or shuffle operation. TaskTracker also runs in its own JVM Process.

2. Every TaskTracker is configured with a set of slots; these indicate the number of tasks that it can accept. The TaskTracker starts a separate JVM processes to do the actual work (called as Task Instance) this is to ensure that process failure does not take down the task tracker.
3. The TaskTracker monitors these task instances, capturing the output and exit codes. When the Task instances finish, successfully or not, the task tracker notifies the JobTracker.
4. The TaskTracker also send out heartbeat messages to the JobTracker, usually every few minutes, to reassure the JobTracker that it is still alive. These messages also inform the JobTracker of the number of available slots, so the JobTracker can stay up to date with where in the cluster work can be delegated.

Qs. Explain what are the basic parameters of a Mapper?
The basic parameters of a Mapper are

1. LongWritable and Text
2. Text and IntWritable

Qs. What is Distributed Cache in Hadoop?
Distributed Cache is a facility provided by the MapReduce framework to cache files (text, archives, jars and so on) needed by applications during execution of the job. The framework will copy the necessary files to the slave node before any tasks for the job are executed on that node.

Qs. What is the benefit of Distributed cache? Why can we just have the file in HDFS and have the application read it?
Distributed cache is much faster. It copies the file to all trackers at the start of the job. Now if the task tracker runs 10 or 100 Mappers or Reducer, it will use the same copy of distributed cache. On the other hand, if you put code in file to read it from HDFS in the MR Job then every Mapper will try to access it from HDFS hence if a TaskTracker run 100 map jobs then it will try to read this file 100 times from HDFS. Also HDFS is not very efficient when used like this.

Qs. How can you set an arbitrary number of Reducers to be created for a job in Hadoop?
You can either do it programmatically by using method setNumReduceTasks in the Jobconf Class or set it up as a configuration setting.

Qs. How will you write a custom partitioner for a Hadoop job?
To have Hadoop use a custom partitioner you will have to do minimum the following three:

- Create a new class that extends Partitioner Class
- Override method getPartition
- In the wrapper that runs the Mapreduce, either
- Add the custom partitioner to the job programmatically using method set Partitioner Class or – add the custom partitioner to the job as a config file (if your wrapper reads from config file or oozie)

Qs. How HDFA differs with NAS?
Following are differences between HDFS and NAS

- In HDFS Data Blocks are distributed across local drives of all machines in a cluster, whereas in NAS data is stored on dedicated hardware.
- HDFS is designed to work with MapReduce System, since computation is moved to data. NAS is not suitable for MapReduce since data is stored separately from the computations.
- HDFS runs on a cluster of machines and provides redundancy using replication protocol. Whereas NAS is provided by a single machine therefore does not provide data redundancy.