What is HDFS?

HDFS, the Hadoop Distributed File System, is a distributed file system designed to hold very large amounts of data (terabytes or even petabytes), and provide high-throughput access to this information. Files are stored in a redundant fashion across multiple machines to ensure their durability to failure and high availability to very parallel applications

What does the statement "HDFS is block structured file system" means?

It means that in HDFS individual files are broken into blocks of a fixed size. These blocks are stored across a cluster of one or more machines with data storage capacity

What does the term "Replication factor" mean?

Replication factor is the number of times a file needs to be replicated in HDFS

What is the default replication factor in HDFS?

3

What is the typical block size of an HDFS block?

64Mb to 128Mb

What is the benefit of having such big block size (when compared to block size of linux file system like ext)?

It allows HDFS to decrease the amount of metadata storage required per file (the list of blocks per file will be smaller as the size of individual blocks increases). Furthermore, it allows for fast streaming reads of data, by keeping large amounts of data sequentially laid out on the disk.

Why is it recommended to have few very large files instead of a lot of small files in HDFS?

This is because the Name node contains the meta data of each and every file in HDFS and more files means more metadata and since namenode loads all the metadata in memory for speed hence having a lot of files may make the metadata information big enough to exceed the size of the memory on the Name node.

True/false question. What is the lowest granularity at which you can apply replication factor in HDSF?

You can choose replication factor per directory.

You can choose replication factor per file in a directory.

You can choose replication factor per block of a file.

True

True

False

What is a datanode in HDFS?

ndividual machines in the HDFS cluster that hold blocks of data are called datanodes.

What is a Namenode in HDSF?

The Namenode stores all the metadata for the file system

What alternate way does HDFS provides to recover data in case a Namenode, without backup, fails and cannot be recovered?

There is no way. If Namenode dies and there is no backup then there is no way to recover data

Describe how a HDFS client will read a file in HDFS,like will it talk to data node or namenode ... how will data flow etc?

To open a file, a client contacts the Name Node and retrieves a list of locations for the blocks that comprise the file. These locations identify the Data Nodes which hold each block. Clients then read file data directly from the Data Node servers, possibly in parallel. The Name Node is not directly involved in this bulk data transfer, keeping its overhead to a minimum.

Using linux command line. how will you List the the number of files in a HDFS directory, Create a directory in HDFS, Copy file from your local directory to HDSF?

```
hadoop fs -ls

hadoop fs -mkdir

hadoop fs -put localfile hdfsfile
```

What are the default configuration files that are used in Hadoop?

As of 0.20 release, Hadoop supported the following read-only default configurations

src/core/core-default.xml

src/hdfs/hdfs-default.xml

src/mapred/mapred-default.xml

How will you make changes to the default configuration files?

Hadoop does not recommends changing the default configuration files, instead it recommends making all site specific changes in the following files

conf/core-site.xml

conf/hdfs-site.xml

conf/mapred-site.xml

Unless explicitly turned off, Hadoop by default specifies two resources, loaded in-order from the classpath:

core-default.xml : Read-only defaults for hadoop.

core-site.xml: Site-specific configuration for a given hadoop installation.

Hence if same configuration is defined in file core-default.xml and src/core/core-default.xml then

the values in file core-default.xml (same is true for other 2 file pairs) is used.

Consider case scenario where you have set property mapred.output.compress to true to ensure that all output files are compressed for efficient space usage on the cluster. If a cluster user does not want to compress data for a specific job then what will you recommend him to do ?

Ask him to create his own configuration file and specify configuration mapred.output.compress to false and load this file as a resource in his job.

In the above case scenario, how can ensure that user cannot override the configuration mapred.output.compress to false in any of his jobs ?

This can be done by setting the property final to true in the core-site.xml file.

What of the following is the only required variable that needs to be set in file conf/hadoop-env.sh for hadoop to work?

HADOOP_LOG_DIR

JAVA_HOME

HADOOP_CLASSPATH

The only required variable to set is JAVA_HOME that needs to point to <java installation> directory

List all the daemons required to run the Hadoop cluster?

NameNode

DataNode

JobTracker

TaskTracker

Whats the default port that jobtrackers listens to?

50030

Whats the default port where the dfs namenode web ui will listen on?

50070

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 get Partition In the wrapper that runs the Map Reducer, either add the custom partitioner to the job programtically using method setPartitionerClass or add the custom partitioner to the job as a config file (if your wrapper reads from config file or oozie)

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

Did you ever built a production process in Hadoop ?

If yes then what was the process when your hadoop job fails due to any reason Its an open ended question but most candidates, if they have written a production job, should talk about some type of alert mechanisn like email is sent or there monitoring system sends an alert. Since Hadoop works on unstructured data, its very important to have a good alerting system for errors since unexpected data can very easily break the job.

Did you ever ran into a lop sided job that resulted in out of memory error, if yes then how did you handled it?

This is an open ended question but a candidate who claims to be an intermediate developer and has worked on large data set (10-20GB min) should have run into this problem. There can be many ways to handle this problem but most common way is to alter your algorithm and break down the job into more map reduce phase or use a combiner if possible.

Whats is Distributed Cache in Hadoop?

Distributed Cache is a facility provided by the Map/Reduce 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.

What is the benifit 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 task tracker 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.

What mechanism does Hadoop framework provides to synchronize changes made in Distribution Cache during runtime of the application?

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

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

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

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

Yes, by using Multiple Outputs class

What will a hadoop job do if you try to run it with an output directory that is already present? Will it overwrite it or or warn you and continue or throw an exception and exit?

The hadoop job will throw an exception and exit.

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

This is a trick question. You cannot set it

How can you set an arbitary number of reducers to be created for a job in Hadoop?

You can either do it progamatically by using method setNumReduceTasksin the JobConfclass or set it up as a configuration setting

How the HDFS Blocks are replicated?

HDFS is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks; all blocks in a file except the last block are the same size. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. An application can specify the number of replicas of a file. The replication factor can be specified at file creation time and can be changed later. Files in HDFS are write-once and have strictly one writer at any time. The NameNode makes all decisionsregarding replication of blocks. HDFS uses rack-aware replica placement policy. In default configuration there are total 3 copies of a datablock on HDFS, 2 copies are stored on datanodes on same rack and 3rd copy on a different rack.

What is a JobTracker in Hadoop? How many instances of JobTracker run on a Hadoop Cluster?

JobTracker is the daemon service for submitting and tracking MapReduce jobs in Hadoop. There is only One Job Tracker process run on any hadoop cluster. Job Tracker runs on its own JVM process. In a typical production cluster its run on a separate machine. Each slave node is configured with job tracker node location. The JobTracker is single point of failure for the Hadoop MapReduce service. If it goes down, all running jobs are halted. JobTracker in Hadoop performs following actions(from Hadoop Wiki:)

Client applications submit jobs to the Job tracker.

- The JobTracker talks to the NameNode 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.
- 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.
- When the work is completed, the JobTracker updates its status.
- Client applications can poll the JobTracker for information.

How JobTracker schedules a task?

The TaskTrackers send out heartbeat messages to the JobTracker, usually every few minutes, to reassure the JobTracker that it is still alive. These message also inform the JobTracker of the number of available slots, so the JobTracker can stay up to date with where in the cluster work can be delegated. When the JobTracker tries to find somewhere to schedule a task within the MapReduce operations, it first looks for an empty slot on the same server that hosts the DataNode containing the data, and if not, it looks for an empty slot on a machine in the same rack.

What is a Task Tracker in Hadoop? How many instances of TaskTracker run on a Hadoop Cluster?

A TaskTracker is a slave node daemon in the cluster that accepts tasks (Map, Reduce and Shuffle operations) from a JobTracker. There is only One Task Tracker process run on any hadoop slave node. Task Tracker runs on its own JVM process. 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. 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. The TaskTrackers also send out heartbeat messages to the JobTracker, usually every few minutes, to reassure the JobTracker that it is still alive. These message 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.

What is a Task instance in Hadoop? Where does it run?

Task instances are the actual MapReduce jobs which are run on each slave node. 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. Each Task Instance runs on its own JVM process. There can be multiple processes of task instance running on a slave node. This is based on the number of slots configured on task tracker. By default a new task instance JVM process is spawned for a task.

How many Daemon processes run on a Hadoop system?

Hadoop is comprised of five separate daemons. Each of these daemon run in its own JVM. Following 3 Daemons run on Master nodes NameNode – This daemon stores and maintains the metadata for HDFS. Secondary NameNode – Performs housekeeping functions for the NameNode. JobTracker – 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. TaskTracker - Responsible for instantiating and monitoring individual Map and Reduce tasks.

What is configuration of a typical slave node on Hadoop cluster? How many JVMs run on a slave node?

Single instance of a Task Tracker is run on each Slave node. Task tracker is run as a separate JVM process. Single instance of a DataNode daemon is run on each Slave node. DataNode daemon is run as a separate JVM process. One or Multiple instances of Task Instance is run on each slave node. Each task instance is run as aseparate JVM process. The number of Task instances can be controlled by configuration. Typically a high end machine is configured to run more task instances.

What is the difference between HDFS and NAS ?

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. 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 are moved to data.NAS is not suitable for MapReduce since data is stored seperately from the computations. HDFS runs on a cluster of machines and provides redundancy usinga replication protocal. Whereas NAS is provided by a single machine therefore does not provide data redundancy.

How NameNode Handles data node failures?

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

Does MapReduce programming model provide a way for reducers to communicate with each other? In a MapReduce job can a reducer communicate with another reducer?

Nope, MapReduce programming model does not allow reducers to communicate with each other. Reducers run in isolation.

Can I set the number of reducers to zero?

Yes, Setting the number of reducers to zero is a valid configuration in Hadoop. When you set the reducers to zero no reducers will be executed, and the output of each mapper will be stored to a separate file on HDFS. [This is different from the condition when reducers are set to a number greater than zero and the Mappers output (intermediate data) is written to the Local file system(NOT HDFS) of each mappter slave node.]

Where is the Mapper Output (intermediate kay-value data) stored ?

The mapper output (intermediate data) is stored on the Local file system (NOT HDFS) of each individual mapper nodes. This is typically a temporary directory location which can be setup in config by the hadoop administrator. The intermediate data is cleaned up after the Hadoop Job completes.

What are combiners? When should I use a combiner in my MapReduce Job?

Combiners are used to increase the efficiency of a MapReduce program. They are used to aggregate intermediate map output locally on individual mapper outputs. Combiners can help you reduce the amount of data that needs to be transferred across to the reducers. You can use your reducer code as a combiner if the operation performed is commutative and associative. The execution of combiner is not guaranteed, Hadoop may or may not execute a combiner. Also, if required it may execute it more then 1 times. Therefore your MapReduce jobs should not depend on the combiners execution.

What is Writable & WritableComparable interface?

- org.apache.hadoop.io.Writable is a Java interface. Any key or value type in the Hadoop Map-Reduce framework implements this interface. Implementations typically implement a static read(DataInput) method which constructs a new instance, calls readFields(DataInput) and returns the instance.
- org.apache.hadoop.io.WritableComparable is a Java interface. Any type which is to be used as a key in the Hadoop Map-Reduce framework should implement this interface. WritableComparable objects can be compared to each other using Comparators.

What is the Hadoop MapReduce API contract for a key and value Class?

- The Key must implement the org.apache.hadoop.io.WritableComparable interface.
- The value must implement the org.apache.hadoop.io.Writable interface.

What is a IdentityMapper and IdentityReducer in MapReduce ?

- org.apache.hadoop.mapred.lib.IdentityMapper Implements the identity function, mapping inputs directly to outputs. If MapReduce programmer do not set the Mapper Class using JobConf.setMapperClass then IdentityMapper.class is used as a default value.
- org.apache.hadoop.mapred.lib.IdentityReducer Performs no reduction, writing all input values directly to the output. If MapReduce programmer do not set the Reducer Class using JobConf.setReducerClass then IdentityReducer.class is used as a default value.

What is the meaning of speculative execution in Hadoop? Why is it important?

Speculative execution is a way of coping withc individual Machine performance. In large clusters where hundreds or thousands of machines are involved there may be machines which are not performing as fast as others. This may result in delays in a full job due to only one machine not performaing well. To avoid this, speculative execution in hadoop can run multiple copies of same map or reduce task on different slave nodes.The results from first node to finish are used.

When is the reducers are started in a MapReduce job?

In a MapReduce job reducers do not start executing the reduce method until the all Map jobs have completed. Reducers start copying intermediate key-value pairs from the mappers as soon as they are available. The programmer defined reduce method is called only after all the mappers have finished.

If reducers do not start before all mappers finish then why does the progress on MapReduce job shows something like Map(50%) Reduce(10%)? Why reducers progress percentage is displayed when mapper is not finished yet?

Reducers start copying intermediate key-value pairs from the mappers as soon as they are available. The progress calculation also takes in account the processing of data transfer which is done by reduce process, therefore the reduce progress starts showing up as soon as any intermediate key-value pair for a mapper isavailable to be transferred to reducer. Though the reducer progress is updated still the programmer defined reduce method is called only after all the mappers have finished.

What is HDFS ? How it is different from traditional file systems?

HDFS, the Hadoop Distributed File System, is responsible for storing huge data on the cluster. This is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS is designed to support very large files. Applications that are compatible with HDFS are those that deal with large data sets. These applications write their data only once but they read it one or more times and require these reads to be satisfied at streaming speeds. HDFS supports write-once-read-many semantics on files.

What is HDFS Block size? How is it different from traditional file system block size?

In HDFS data is split into blocks and distributed across multiple nodes in the cluster. Each block is typically 64Mb or 128Mb in size. Each block is replicated multiple times. Default is to replicate each block three times. Replicas are stored on different nodes. HDFS utilizes the local file system to store each HDFS block as a separate file. HDFS Block size can not be compared with the traditional file system block size.

What is a NameNode? How many instances of NameNode run on a Hadoop Cluster?

The NameNode is the centerpiece of an HDFS file system. It keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these files itself. There is only One NameNode process run on any hadoop cluster. NameNode runs on its own JVM process. In a typical production cluster its run on a separate machine. The NameNode is a Single Point of Failure for the HDFS Cluster. When the NameNode goes down, the file system goes offline. Client applications talk to the NameNode whenever they wish to locate a file, or when they want to add/copy/move/delete a file. The NameNode responds the successful requests by returning a list of relevant DataNode servers where the data lives.

What is a DataNode? How many instances of DataNode run on a Hadoop Cluster?

A DataNode stores data in the Hadoop File System HDFS. There is only One DataNode process run on any hadoop slave node. DataNode runs on its own JVM process. On startup, a DataNode connects to the NameNode. DataNode instances can talk to each other, this is mostly during replicating data.

How the Client communicates with HDFS?

The Client communication to HDFS happens using Hadoop HDFS API. Client applications talk to the NameNode whenever they wish to locate a file, or when they want to add/copy/move/delete a file on HDFS. The NameNode responds the successful requests by returning a list of relevant DataNode servers where the data lives. Client applications can talk directly to a DataNode, once the NameNode has provided the location of the data.