1. Data is replicated in multiple node, 3 by default. How does hadoop decide on which node to run the map job?
2. **What HDFS is not meant for?**

HDFS is not good at:

* 1. Applications that requires low latency access to data (in terms of milliseconds)
  2. Lot of small files
  3. Multiple writers and file modifications

1. **What different type of schedulers and type of scheduler did you use?**
   1. **Capacity Scheduler**  
      It is designed to run Hadoop applications as a shared, multi-tenant cluster while maximizing the throughput and the utilization of the cluster.
   2. **Fair Scheduler**  
      Fair scheduling is a method of assigning resources to applications such that all apps get, on average, an equal share of resources over time.
2. **Steps involved in decommissioning (removing) the nodes in the Hadoop cluster?**

* Update the network addresses in the dfs.exclude and mapred.exclude
* $ hadoop dfsadmin -refreshNodes and hadoop mradmin -refreshNodes
* Check Web UI it will show “Decommissioning in Progress”
* Remove the Nodes from include file and then run again the step 2 refreshNodes.
* Remove the Nodes from slave file.

1. **Steps involved in commissioning (adding) the nodes in the Hadoop cluster?**
   1. Update the network addresses in the dfs.include and mapred.include
   2. $ hadoop dfsadmin -refreshNodes and hadoop mradmin -refreshNodes
   3. Update the slave file.
   4. Start the DataNode and NodeManager on the added Node.
2. **How to keep HDFS cluster balanced?**

Balancer is a tool that tries to provide a balance to a certain threshold among data nodes by copying block data distribution across the cluster.

1. **What is distcp?**

distcp is the program comes with Hadoop for copying large amount of data to and from Hadoop file systems in parallel.

It is implemented as MapReduce job where copying is done through maps that run in parallel across the cluster.

There are no reducers.

1. **What are the daemons of HDFS? – Name Node, Data node,** **Secondary NameNode.**
2. **What are the functions of NameNode?**

The NameNode is mainly responsible for:

* **Namespace**  
  Maintain metadata about the data
* **Block Management**  
  Processes block reports and maintain location of blocks.  
  Supports block related operations  
  Manages replica placement

1. **What is HDFS Federation?**

HDFS federation allows scaling the name service horizontally; it uses multiple independent NameNodes for different namespaces. All the NameNodes use the DataNodes as common storage for blocks. Each DataNode registers with all the NameNodes in the cluster.

DataNodes send periodic heartbeats and block reports and handles commands from the NameNodes.

1. What **is HDFS High Availability?**

In HDFS High Availability (HA) cluster; two separate machines are configured as NameNodes.

But one of the NameNodes is in an **Active** state; 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

They shared the same storage and all DataNodes connects to both the NameNodes.

1. **How client application interacts with the NameNode?**

Client applications interact using Hadoop HDFS API with the NameNode when it has to locate/add/copy/move/delete a file.

The NameNode responds the successful requests by returning a list of relevant DataNode servers where the data is residing.

Client can talk directly to a DataNode after the NameNode has given the location of the data

1. **What is rack-aware replica placement policy?**

Rack-awareness is used to take a node’s physical location into account while scheduling tasks and allocating storage. Default replication factor is 3 for a data blocks on HDFS. The first two copies are stored on DataNodes located on the same rack while the third copy is stored on a different rack.

1. **What is the main purpose of HDFS fsck command?**

fsck is a utility to check health of the file system, to find missing files, over-replicated, under-replicated and corrupted blocks.

**$ hadoop fsck -files -blocks –racks**

1. **What is the purpose of DataNode block scanner?**

Block scanner runs on every DataNode, which periodically verifies all the blocks stored on the DataNode. If bad blocks are detected it will be fixed before any client reads.

1. **What is the purpose of dfsadmin tool?**

It is used to find information about the state of HDFS

It performs administrative tasks on HDFS

Invoked by hadoop dfsadmin command as superuser

1. **What is the command for printing the topology?**

It displays a tree of racks and DataNodes attached to the tracks as viewed by the hdfs.

*dfsadmin -printTopology*

1. **What is RAID?**

RAID is a way of combining multiple disk drives into a single entity to improve performance and/or reliability. There are a variety of different levels in RAID  
For example, In RAID level 1 copy of the same data on two disks increases the read performance by reading alternately from each disk in the mirror.

1. **Does Hadoop requires RAID?**

In DataNodes storage is not using RAID as redundancy can be achieved by replication between the Nodes. In NameNode’s disk RAID is recommended.

1. **Assuming default configurations, how is a file of the size 1 GB (uncompressed) stored in HDFS?**

Default block size is 64MB. So, file of 1GB will be stored as 16 blocks. MapReduce job will create 16 input splits; each will be processed with separate map task i.e. 16 mappers.

1. **What are Hadoop Writables?**

Hadoop Writables allows Hadoop to read and write the data in a serialized form for transmission as compact binary files. This helps in straightforward random access and higher performance. Hadoop provides in built classes, which implement Writable: Text, IntWritable, LongWritable, FloatWritable, and BooleanWritable.

1. **Why comparison of types is important for MapReduce?**

A comparison is important as in the sorting phase the keys are compared with one another. For comparison, the WritableComparable interface is implemented.

1. **What is the purpose of RawComparator interface?**

RawComparator allows the implementors to compare records read from a stream without deserialization them into objects, so it will be optimized, as there is not overhead of object creation.

1. **What is a NullWritable?**

It is a special type of Writable that has zero-length serialization. In MapReduce, a key or a value can be declared as NullWritable if we don’t need that position, storing constant empty value.

1. **What is Avro Serialization System?**

Avro is a language-neutral data serialization system. It has data formats that work with different languages. Avro data is described using a language-independent schema (usually written in JSON). Avro data files support compression and are **splittable**.

Avro provides AvroMapper and AvroReducer to run MapReduce programs.

1. **Explain use cases where SequenceFile class can be a good fit?**

When the data is of type binary then SequenceFile will provide a persistent structure for binary key-value pairs. SequenceFiles also work well as containers for smaller files as HDFS and MapReduce are optimized for large files.

1. **What is MapFile?**

A MapFile is an indexed SequenceFile and it is used for look-ups by key.

1. **What are the steps involved in MapReduce framework?**

Firstly, the mapper input key/value pairs maps to a set of intermediate key/value pairs.

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 maps to zero or many output pairs.

The Hadoop MapReduce framework creates one map task for each InputSplit generated by the InputFormat for the job.

It then calls map(WritableComparable, Writable, Context) for each key/value pair in the InputSplit for that task.

All intermediate values associated with a given output key are grouped passed to the Reducers.

1. Where **is the Mapper Output stored?**

The mapper output is stored on the Local file system of each individual mapper nodes. The intermediate data is cleaned up after the Hadoop Job completes.

1. **What is a partitioner and how the user can control which key will go to which reducer?**

Partitioner controls the partitioning of the keys of the intermediate map-outputs by the default. The key to decide the partition uses hash function. Default partitioner is HashPartitioner. A custom partitioner is implemented to control, which keys go to which Reducer.

public class SamplePartitioner extends Partitioner {

@Override

public int getPartition(Text key, Text value, int numReduceTasks) {

}}

1. **What are combiners and its purpose?**

Combiners are used to increase the efficiency of a MapReduce program. It can be used to aggregate intermediate map output locally on individual mapper outputs.

Combiners can help reduce the amount of data that needs to be transferred across to the reducers. Reducer code as a combiner if the operation performed is commutative and associative.

1. **How a number of partitioners and reducers are related?**

The total numbers of partitions are the same as the number of reduce tasks for the job.

1. **What is IdentityMapper?**

IdentityMapper implements the mapping inputs directly to output. IdentityMapper.class is used as a default value when JobConf.setMapperClass is not set.

1. **What is IdentityReducer?**

In IdentityReducer no reduction is performed, writing all input values directly to the output. IdentityReducer.class is used as a default value when JobConf.setReducerClass is not set.

1. **What is the reducer and its phases?**

Reducer reduces a set of intermediate values, which has same key to a smaller set of values. The framework then calls reduce().  
**Syntax:**  
reduce(WritableComparable, Iterable, Context) method for each pair in the grouped inputs.  
Reducer has three primary phases:

Shuffle

Sort

Reduce

**36. Detail description of the Reducer phases?**

**Shuffle:**  
Sorted output (Mapper) à Input (Reducer). Framework then fetches the relevant partition of the output of all the mappers.

**Sort:**  
The framework groups Reducer inputs by keys. The shuffle and sort phases occur simultaneously; while map-outputs are being fetched they are merged.

**Secondary Sort:**  
Grouping the intermediate keys are required to be different from those for grouping keys before reduction, then Job.setSortComparatorClass(Class).

**Reduce:**  
reduce(WritableComparable, Iterable, Context) method is called for each pair in the grouped inputs.  
The output of the reduce task is typically written using Context.write(WritableComparable, Writable).

1. **Can there be no Reducer?**

Yes, the number of reducer can be zero if no reduction of values is required.

1. **What can be optimum value for Reducer?**

Value of Reducers can be: 0.95

1. 1.75 multiplied by ( \* < number of maximum containers per node>)

Increasing number of reducers

1. Increases the framework overhead
2. Increases load balancing
3. Lowers the cost of failures
4. **What are a Counter and its purpose?**

The counter is a facility for MapReduce applications to report its statistics. They can be used to track job progress in a very easy and flexible manner. It is defined by MapReduce framework or by applications. Each Counter can be of any Enum type. Applications can define counters of type Enum and update them via counters.incrCounter in the map and/or reduce methods.

1. **Define different types of Counters?**

**Built in Counters:**

* Map Reduce Task Counters
* Job Counters

**Custom Java Counters:**

* MapReduce allows users to specify their own counters (using Java enums) for performing their own counting operation.

1. **Why Counter values are shared by all map and reduce tasks across the MapReduce framework?**

Counters are global so shared across the MapReduce framework and aggregated at the end of the job across all the tasks.

1. **Explain speculative execution.**

Speculative execution is a way of dealing with individual machine’s performance. As there are lots of machines in the cluster, some machines can have low performance, which affects the performance of the whole job.

Speculative execution in Hadoop can run multiple copies of the same map or reduce task on different task tracker nodes and the results from first node to finish are used.

1. **What is DistributedCache and its purpose?**

DistributedCache is a facility provided by the MapReduce framework to cache files (text, archives, jars etc.) needed by applications. It distributes application-specific, large, read-only files efficiently. The user needs to use DistributedCache to distribute and symlink the script file.

1. **What is the Job interface in MapReduce framework?**

Job is the primary interface for a user to describe a MapReduce job to the Hadoop framework for execution. Some basic parameters are configured for example:

Job.setNumReduceTasks(int)

Configuration.set(JobContext.NUM\_MAPS, int)

Mapper

Combiner (if any)

Partitioner

Reducer

InputFormat

OutputFormat implementations

setMapSpeculativeExecution(boolean))/ setReduceSpeculativeExecution(boolean))

Maximum number of attempts per task (setMaxMapAttempts(int)/ setMaxReduceAttempts(int)) etc.

DistributedCache for large amounts of (read-only) data.

1. **What is the default value of map and reduce max attempts?**

The framework will try to execute a map task or reduce task by **default 4 times** before giving up on it.

1. **Explain InputFormat?**