1. **What is difference between Block and Split?**

Block are HDFS concept, while split is Map reduce concept.

Split is logical split of the data, basically used during data processing using Map/Reduce program or other data processing techniques on Hadoop Ecosystem. Split size is user defined value and you can choose your own split size based on your volume of data (How much data you are processing).

Split is basically used to control number of Mapper in Map/Reduce program. If you have not defined any input split size in Map/Reduce program, then default HDFS block split will be considered as input split.

1. **Compare MapReduce and Spark**

|  |  |  |
| --- | --- | --- |
| **Criteria** | **MapReduce** | **Spark** |
| **Processing Speeds** | Good | Exceptional |
| **Standalone mode** | Needs Hadoop | Can work independently |
| **Ease of use** | Needs extensive Java program | APIs for Python, Java, & Scala |
| **Versatility** | Real-time & machine learning applications | Not optimized for real-time & machine learning applications |

1. **What is Data locality? What is need of Data Locality in Hadoop MapReduce?**

Data locality is the process of moving the computation close to where the actual data resides on the node, instead of moving large data to computation.

1. **How number of Mapper and reducer is decided?**

There are two conditions for no. of mappers.  
(1) No. of Mappers per slave  
(2) No. of Mappers per [MapReduce job](http://data-flair.training/blogs/hadoop-mapreduce-introduction-tutorial-comprehensive-guide/)

**(1) No. of Mappers per slave:** There is no exact formula. It depends on how many cores and how much memory you have on each slave. Generally, one mapper should get 1 to 1.5 cores of processors. So if you have 15 cores then one can run 10 Mappers per Node. So if you have 100 data nodes in [Hadoop Cluster](http://data-flair.training/blogs/install-configure-apache-hadoop-2-7-x-on-ubuntu/) then one can run 1000 Mappers in a Cluster.

**(2) No. of Mappers per MapReduce job:**The number of mappers depends on the amount of [InputSplit](http://data-flair.training/blogs/inputsplit-in-hadoop-mapreduce/) generated by trong>[InputFormat](http://data-flair.training/blogs/hadoop-inputformat-types/) (getInputSplits method). If you have 640MB file and [Data Block size](http://data-flair.training/blogs/data-blocks-hdfs-hadoop-distributed-file-system/) is 128 MB then we need to run 5 Mappers per MapReduce job.

There are two conditions for no. of reducers.  
(1) No. of Reducers per slave  
(2) No. of Reducers per MapReduce job

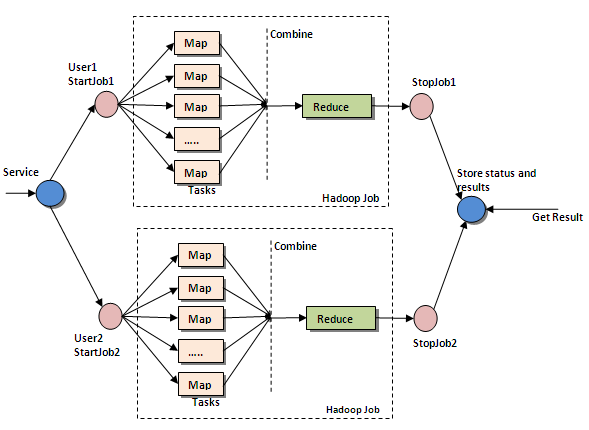
**(1) No. of Reducers per slave:** It is same as No of Mappers per slave

**(2) No. of Reducers per MapReduce job:**  
The right no. reducer we can set with following formula:  
0.95 \* no. of nodes \* mapred.tasktracker.reduce.tasks.maximum  
or  
1.75 \* no. of nodes \* mapred.tasktracker.reduce.tasks.maximum

With 0.95 all of the reducers can launch immediately and start transferring map o/p when map finished.  
With 1.75 faster nodes will finish their first round of reduces and launch the second wave of reduces.

**Follow the links to learn more about**  
[Number of Mappers in Hadoop](http://data-flair.training/blogs/mapper-in-hadoop-mapreduce/)  
[Number of Reducers in Hadoop](http://data-flair.training/blogs/reducer-in-hadoop-mapreduce/)

1. **What is MapReduce Mechanism?**



1. **How to set reuse JVM in Hadoop mapreduce jobs?**

If you have very small tasks that are definitely running after each other, it is useful to set this property to -1 (meaning that a spawned JVM will be reused unlimited times). So you just spawn (number of task in your cluster available to your job)-JVMs instead of (number of tasks)-JVMs.

This is a huge performance improvement. In long running jobs the percentage of the runtime in comparision to setup a new JVM is very low, so it doesn't give you a huge performance boost.

1. **How to skip bad records in hadoop map-reduce?**

Skipping mode is off by default; you enable it independently for map and reduce tasks using the SkipBadRecords class. It’s important to note that skipping mode can detect only one bad record per task attempt, so this mechanism is appropriate only for detecting occasional bad records (a few per task, say). You may need to increase the maximum number of task attempts (via mapred.map.max.attempts and mapred.reduce.max.attempts) to give skipping mode enough attempts to detect and skip all the bad records in an input split. Bad records that have been detected by Hadoop are saved as sequence files in the job’s output directory under the \_logs/skip subdirectory. These can be inspected for diagnostic purposes after the job has completed (using hadoop fs -text, for example).

1. **Who sets number of Mapper and Reducer?**

Number of mappers is intelligently calculating by Hadoop keeping in consideration split size and data locality. Number of reducer is set by user. In case of single reducer all advantage of parallel mapper is gone. So we need to have multiple reducer.

1. **What is MapReduce?**

Referred as the core of Hadoop, MapReduce is a programming framework to process large sets of data or big data across thousands of servers in a Hadoop Cluster. The concept of MapReduce is similar to the cluster scale-out data processing systems. The term MapReduce refers to two important processes of Hadoop program operates.

First is the map() job, which converts a set of data into another breaking down individual elements into key/value pairs (tuples). Then comes reduce() job into play, wherein the output from the map, i.e. the tuples serve as the input and are combined into smaller set of tuples. As the name suggests, the map job every time occurs before the reduce one.

1. **Illustrate a simple example of the working of MapReduce.**

Let’s take a simple example to understand the functioning of MapReduce. However, in real-time projects and applications, this is going to be elaborate and complex as the data we deal with Hadoop and MapReduce is extensive and massive.

Assume you have five files and each file consists of two key/value pairs as in two columns in each file – a city name and its temperature recorded. Here, name of city is the key and the temperature is value.  
San Francisco, 22  
Los Angeles, 15  
Vancouver, 30  
London, 25  
Los Angeles, 16  
Vancouver, 28  
London,12

It is important to note that each file may consist of the data for same city multiple times. Now, out of this data, we need to calculate the maximum temperature for each city across these five files. As explained, the MapReduce framework will divide it into five map tasks and each map task will perform data functions on one of the five files and returns maxim temperature for each city.

(San Francisco, 22)(Los Angeles, 16)(Vancouver, 30)(London, 25)  
Similarly each mapper performs it for the other four files and produce intermediate results, for instance like below.

(San Francisco, 32)(Los Angeles, 2)(Vancouver, 8)(London, 27)  
(San Francisco, 29)(Los Angeles, 19)(Vancouver, 28)(London, 12)  
(San Francisco, 18)(Los Angeles, 24)(Vancouver, 36)(London, 10)  
(San Francisco, 30)(Los Angeles, 11)(Vancouver, 12)(London, 5)

These tasks are then passed to the reduce job, where the input from all files are combined to output a single value. The final results here would be:

(San Francisco, 32)(Los Angeles, 24)(Vancouver, 36)(London, 27)

These calculations are perform instantly and are extremely efficient to calculate outputs on a large dataset.

1. **What are the main components of MapReduce Job?**

**Main Driver Class:** providing job configuration parameters  
**Mapper Class:** must extend org.apache.hadoop.mapreduce.Mapper class and performs execution of map() method  
**Reducer Class:** must extend org.apache.hadoop.mapreduce.Reducer class

1. **What is Shuffling and Sorting in MapReduce?**

Shuffling and Sorting are two major processes operating simultaneously during the working of mapper and reducer.

The process of transferring data from Mapper to reducer is Shuffling. It is a mandatory operation for reducers to proceed their jobs further as the shuffling process serves as input for the reduce tasks.

In MapReduce, the output key-value pairs between the map and reduce phases (after the mapper) are automatically sorted before moving to the Reducer. This feature is helpful in programs where you need sorting at some stages. It also saves the programmer’s overall time.

1. **What is Partitioner and its usage?**

Partitioner is yet another important phase that controls the partitioning of the intermediate map-reduce output keys using a hash function. The process of partitioning determines in what reducer, a key-value pair (of the map output) is sent. The number of partitions is equal to the total number of reduce jobs for the process.

Hash Partitioner is the default class available in Hadoop , which implements the following function.int getPartition(K key, V value, int numReduceTasks)  
The function returns the partition number using the numReduceTasks is the number of fixed reducers.

1. **What is Identity Mapper and Chain Mapper?**

Identity Mapper is the default Mapper class provided by Hadoop. when no other Mapper class is defined, Identity will be executed. It only writes the input data into output and do not perform and computations and calculations on the input data.

The class name is org.apache.hadoop.mapred.lib.IdentityMapper.

Chain Mapper is the implementation of simple Mapper class through chain operations across a set of Mapper classes, within a single map task. In this, the output from the first mapper becomes the input for second mapper and second mapper’s output the input for third mapper and so on until the last mapper.

The class name is org.apache.hadoop.mapreduce.lib.ChainMapper

1. **What main configuration parameters are specified in MapReduce?**

The MapReduce programmers need to specify following configuration parameters to perform the map and reduce jobs:

The input location of the job in HDFs.

The output location of the job in HDFS.

The input’s and output’s format.

The classes containing map and reduce functions, respectively.

The .jar file for mapper, reducer and driver classes

1. **Name Job control options specified by MapReduce.**

Since this framework supports chained operations wherein an input of one map job serves as the output for other, there is a need for job controls to govern these complex operations.

The various job control options are:

**Job.submit() :** to submit the job to the cluster and immediately return

**Job.waitforCompletion(boolean) :** to submit the job to the cluster and wait for its completion

1. **What is InputFormat in Hadoop?**

Another important feature in MapReduce programming, InputFormat defines the input specifications for a job. It performs the following functions:

* Validates the input-specification of job.
* Split the input file(s) into logical instances called InputSplit. Each of these split files are then assigned to individual Mapper.
* Provides implementation of RecordReader to extract input records from the above instances for further Mapper processing

1. **What is the difference between HDFS block and InputSplit?**

An HDFS block splits data into physical divisions while InputSplit in MapReduce splits input files logically.

While InputSplit is used to control number of mappers, the size of splits is user defined. On the contrary, the HDFS block size is fixed to 64 MB, i.e. for 1GB data , it will be 1GB/64MB = 16 splits/blocks. However, if input split size is not defined by user, it takes the HDFS default block size.

1. **What is Text Input Format?**

It is the default InputFormat for plain text files in a given job having input files with .gz extension. In TextInputFormat, files are broken into lines, wherein key is position in the file and value refers to the line of text. Programmers can write their own InputFormat.  
The hierarchy is:

java.lang.Object

org.apache.hadoop.mapreduce.InputFormat<K,V>

org.apache.hadoop.mapreduce.lib.input.FileInputFormat<LongWritable,Text>

org.apache.hadoop.mapreduce.lib.input.TextInputFormat

1. **What is JobTracker?**

JobTracker is a Hadoop service used for the processing of MapReduce jobs  in the cluster. It submits and tracks the jobs to specific nodes having data. Only one JobTracker runs on single Hadoop cluster on its own JVM process. if JobTracker goes down, all the jobs halt.

1. **Explain job scheduling through JobTracker.**

JobTracker communicates with NameNode to identify data location and submits the work to TaskTracker node. The TaskTracker plays a major role as it notifies the JobTracker for any job failure. It actually is referred to the heartbeat reporter reassuring the JobTracker that it is still alive. Later, the JobTracker is responsible for the actions as in it may either resubmit the job or mark a specific record as unreliable or blacklist it.

1. **What is SequenceFileInputFormat?**

A compressed binary output file format to read in sequence files and extends the FileInputFormat.It passes data between output-input (between output of one MapReduce job to input of another MapReduce job)phases of MapReduce jobs.

1. **How to set mappers and reducers for Hadoop jobs?**

Users can configure JobConf variable to set number of mappers and reducers.

job.setNumMaptasks()

job.setNumreduceTasks()

1. **Explain JobConf in MapReduce.**

It is a primary interface to define a map-reduce job in the Hadoop for job execution. JobConf specifies mapper, Combiner, partitioner, Reducer,InputFormat , OutputFormat implementations and other advanced job faets liek Comparators.

1. **What is a MapReduce Combiner?**

Also known as semi-reducer, Combiner is an optional class to combine the map out records using the same key. The main function of a combiner is to accept inputs from Map Class and pass those key-value pairs to Reducer class

1. **What is RecordReader in a Map Reduce?**

RecordReader is used to read key/value pairs form the InputSplit by converting the byte-oriented view and presenting record-oriented view to Mapper.

1. **Define Writable data types in MapReduce.**

Hadoop reads and writes data in a serialized form in writable interface. The Writable interface has several classes like Text (storing String data), IntWritable, LongWriatble, FloatWritable, BooleanWritable. users are free to define their personal Writable classes as well.

[Read this blog](https://intellipaat.com/blog/hadoop-processor-rising-speed-of-big-data-technologies/) to see how the mapping and reducing speeds are increasing in the MapReduce processing engine.

1. **What is OutputCommitter?**

OutPutCommitter describes the commit of MapReduce task. FileOutputCommitter is the default available class available for OutputCommitter in MapReduce. It performs the following operations:

* Create temporary output directory for the job during initialization.
* Then, it cleans the job as in removes temporary output directory post job completion.
* Sets up the task temporary output.
* Identifies whether a task needs commit. The commit is applied if required.
* JobSetup, JobCleanup and TaskCleanup are important tasks during output commit.

1. **What is a “map” in Hadoop?**

In Hadoop, a map is a phase in HDFS query solving. A map reads data from an input location, and outputs a key value pair according to the input type.

1. **What is a “reducer” in Hadoop?**

In Hadoop, a reducer collects the output generated by the mapper, processes it, and creates a final output of its own.

1. **What are the parameters of mappers and reducers?**

The four parameters for mappers are:

* LongWritable (input)
* text (input)
* text (intermediate output)
* IntWritable (intermediate output)

The four parameters for reducers are:

* Text (intermediate output)
* IntWritable (intermediate output)
* Text (final output)
* IntWritable (final output)

1. **What are the key differences between Pig vs MapReduce?**

PIG is a data flow language; the key focus of Pig is to manage the flow of data from input source to output store. As part of managing this data flow it moves data feeding it to

**process 1.** taking output and feeding it to

**process2.** The core features are preventing execution of subsequent stages if previous stage fails, manages temporary storage of data and most importantly compresses and rearranges processing steps for faster processing. While this can be done for any kind of processing tasks Pig is written specifically for managing data flow of Map reduce type of jobs. Most if not all jobs in a Pig are map reduce jobs or data movement jobs. Pig allows for custom functions to be added which can be used for processing in Pig, some default ones are like ordering, grouping, distinct, count etc.

Mapreduce on the other hand is a data processing paradigm, it is a framework for application developers to write code in so that its easily scaled to PB of tasks, this creates a separation between the developer that writes the application vs the developer that scales the application. Not all applications can be migrated to Map reduce but good few can be including complex ones like k-means to simple ones like counting uniques in a dataset.

1. **What is partitioning?**

Partitioning is a process to identify the reducer instance which would be used to supply the mappers output. Before mapper emits the data (Key Value) pair to reducer, mapper identify the reducer as an recipient of mapper output. All the key, no matter which mapper has generated this, must lie with same reducer.

1. **How to set which framework would be used to run mapreduce program?**

mapreduce.framework.name. it can be

* Local
* classic
* Yarn

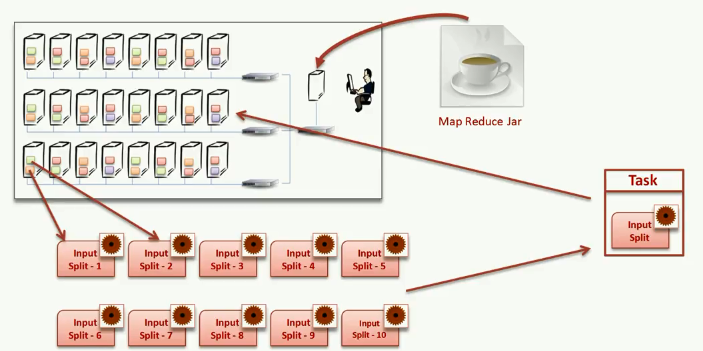
1. **What platform and Java version is required to run Hadoop?**

Java 1.6.x or higher version are good for Hadoop, preferably from Sun. Linux and Windows are the supported operating system for Hadoop, but BSD, Mac OS/X and Solaris are more famous to work.

1. **Can MapReduce program be written in any language other than Java?**

Yes, Mapreduce can be written in many programming languages Java, R, C++, scripting Languages (Python, PHP). Any language able to read from stadin and write to stdout and parse tab and newline characters should work . Hadoop streaming (A Hadoop Utility) allows you to create and run Map/Reduce jobs with any executable or scripts as the mapper and/or the reducer.

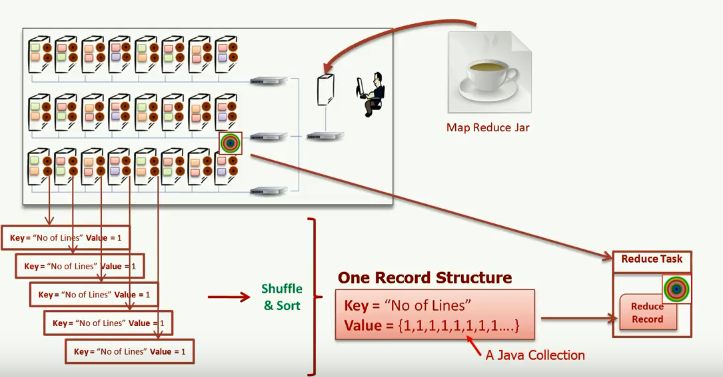
1. **How will a map function read lines from a block?**



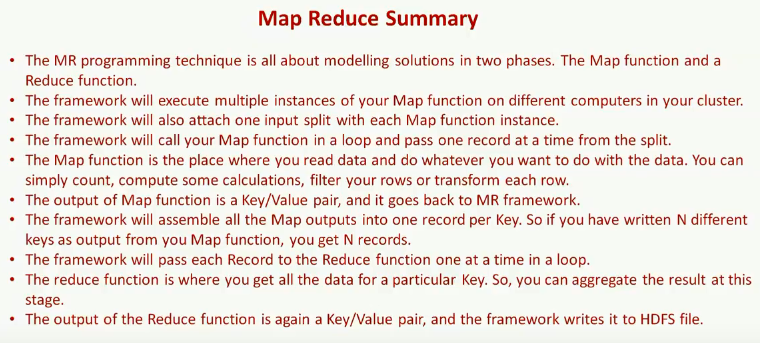
1. **Who will collect data from each map computer and send it to the reduce computer?**

The execution engine takes care of all these things.

To run a MR Job we need to compile and package it in Jar using build tools like maven and sbt. Then we need to submit it to YARN for execution.



1. **What is Map Reduce?**



How and where to use log4j with Map reduce jobs?

We don’t need reducer always.

1. **What is Tool Runner How to use Tool Runner?**

ToolRunner can be used to run classes implementing Tool interface. It works in conjunction with GenericOptionsParser to parse the [generic hadoop command line arguments](https://hadoop.apache.org/docs/r2.6.3/hadoop-project-dist/hadoop-common/CommandsManual.html#Generic_Options) and modifies the Configuration of the Tool. The application-specific options are passed along without being modified.

1. **What are the basic parameters of a Mapper?**

The basic parameters of a Mapper are

* LongWritable and Text
* Text and IntWritable

Hadoop and Spark are the two most popular big data frameworks. But there is a commonly asked question – do we need Hadoop to run Spark? Watch this video to find the answer to this question.

1. **How do Hadoop MapReduce works?**

There are two phases of MapReduce operation.

* Map phase – In this phase, the input data is split by map tasks. The map tasks run in parallel. These split data is used for analysis purpose.
* Reduce phase- In this phase, the similar split data is aggregated from the entire collection and shows the result.

1. **What is MapReduce? What is the syntax you use to run a MapReduce program?**

MapReduce is a programming model in Hadoop for processing large data sets over a cluster of computers, commonly known as HDFS. It is a parallel programming model.

The syntax to run a MapReduce program is – hadoop\_jar\_file.jar /input\_path /output\_path.

1. **What are the configuration parameters in a “MapReduce” program?**

The main configuration parameters in “MapReduce” framework are:

* Input locations of Jobs in the distributed file system
* Output location of Jobs in the distributed file system
* The input format of data
* The output format of data
* The class which contains the map function
* The class which contains the reduce function
* JAR file which contains the mapper, reducer and the driver classes