**Map Reduce Interview Questions**

**Definition:**

MapReduce is a programming framework that allows us to perform distributed and parallel processing on large data sets in a distributed environment.

* **MapReduce 2.0 or YARN Architecture:**
  + MapReduce framework also follows Master/Slave Topology where the master node (Resource Manager) manages and tracks various MapReduce jobs being executed on the slave nodes (Node Mangers).
  + Resource Manager consists of two main components:
    - **Application Manager:** It accepts job-submissions, negotiates the container for ApplicationMaster and handles failures while executing MapReduce jobs.
    - **Scheduler:** Scheduler allocates resources that is required by various MapReduce application running on the Hadoop cluster.
* **How MapReduce job works:**
  + As the name MapReduce suggests, reducer phase takes place after the mapper phase has been completed.
  + So, the first is the map job, where a block of data is read and processed to produce key-value pairs as intermediate outputs.
  + The reducer receives the key-value pair from multiple map jobs.
  + Then, the reducer aggregates those intermediate data tuples (intermediate key-value pair) into a smaller set of tuples or key-value pairs which is the final output.
* **The syntax for running the MapReduce program is**

|  |  |
| --- | --- |
|  | hadoop\_jar\_file.jar /input\_path /output\_path. |
|  |  |

### ****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](https://intellipaat.com/blog/hadoop-mapreduce-what-it-refers-to/) 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.

### 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

### ****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.

### ****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.  
SanFrancisco,22  
LosAngeles,15  
Vancouver,30  
London,25  
LosAngeles,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.

### What are the roles of Mapper, Combiner, Partitioner, Shuffle & Sort and Reducer classes?

Entire Map reduce works on the Key-Value pairs  
Mapper phase takes input key-value pairs from a file or from any source and produces output in the form of key-value pairs after performing transformations. Those key-value pairs can be of simple types like IntWritable, LongWritable, text etc., or it can also be some composite types if you can create your custom writable or custom writable comparable. The key value of the mapper come from different machines, all the values of one key are collected together, so the shuffle makes sure that the values having the same key are collected together and then that is passed to the reducer.  
The reducer takes the group of values for a key, in map reduce and is known as the iterable of all the values and can perform operations like aggregation etc.,  
Combiner is like a mini reducer, if the operation is associative like if the operation is finding the maximum or minimum then you can have the aggregation before the shuffle phase, this reduces the work of shuffle and also it reduces the load on the reducer. So, based on the requirement, if you can use combiner in the application. But you cannot use combiner all the times. You can use only if your operations are associative and ordering does not matter. Whenever a mapper produces the output, there is a technique to decide, which key should go into which reducer. Suppose, if there are multiple reducers running parallel, partitioner controls which key should go into which reducer. By default, there is hash partitioner that decides which should go into which reducer  
along with its values. Hash petitioner works on the hash code and it decides the reducer by performing the modulus of the key by dividing it with a number of reducers.  
The reducer output is sorted according to the key. The output key of the mapper is taken care by the shuffle and sort. As a developer, you can override the properties of Mapper, Combiner, Partitioner, Reducers, but shuffle and sort is internal to the Hadoop framework.

### How to control the number of reducers in a map reduce program?

By default, for every 1 GB of input data, 1 reducer will be spawned/created. But you can also override this property by using the below property job.setNumReduceTasks(int n)  
The above property will set the number of reducers based on the integer number you provide to the function as parameter.

### How does Hadoop know how many mappers has to be started?

Number of mappers equals the number of input splits Number of input splits(for a single file) = Ceil(Size of file)/(Size of input split)  
For example, if you have 1GB of data and the input split size is 128MB then 1024/128 gives you 8 so 8 mappers will be started.  
In default situations, input split size equals to the block size so number of input splits is equal to the number of blocks. So, you can say that number of mappers is equal to the number of blocks.

### ****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.

**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

**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.

**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

**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.

**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.

**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.

**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()

**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.

### ****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

### ****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.

### ****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.

**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.

**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.

**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.

**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)

**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.

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

mapreduce.framework.name. it can be

1. Local
2. classic
3. Yarn

**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.

**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.

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

Identity Mapper is the default Mapper class provided by Hadoop. when no other Mapper class is defined, Identify 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.

**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

**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

## **What do you mean by data locality?**

* [***Data locality***](https://www.edureka.co/blog/mapreduce-tutorial/#data_locality) talks about moving computation unit to data rather data to the computation unit.
* MapReduce framework achieves data locality by processing data locally
* Which means processing of the data happens in the very node by Node Manager where data blocks are present.

## **Is it mandatory to set input and output type/format in MapReduce?**

No, it is not mandatory to set the input and output type/format in MapReduce. By default, the cluster takes the input and the output type as ‘text’.

## **Can we rename the output file?**

Yes, we can rename the output file by implementing multiple format output class.

## **What are the advantages of using MapReduce with Hadoop?**

|  |  |
| --- | --- |
| **Advantages of MapReduce** | |
| **Advantage** | **Description** |
| ***Flexible*** | Hadoop MapReduce programming can access and operate on different types of structured and unstructured |
| ***Parallel Processing*** | MapReduce programming divides tasks for execution in parallel |
| ***Resilient*** | Is fault tolerant that quickly recognizes the faults & then apply a quick recovery solution implicitly |
| ***Scalable*** | Hadoop is a highly scalable platform that can store as well as distribute large data sets across plenty of servers |
| ***Cost-effective*** | High scalability of Hadoop also makes it a cost-effective solution for ever-growing data storage needs |
| ***Simple*** | It is based on a simple programming model |
| ***Secure*** | Hadoop MapReduce aligns with HDFS and HBase security for security measures |
| ***Speed*** | It uses the distributed file system for storage that processes even the large sets of unstructured data in minutes |

## **Explain the process of spilling in MapReduce?**

The output of a map task is written into a circular memory buffer (RAM). The default size of buffer is set to 100 MB  which can be tuned by using mapreduce.task.io.sort.mb property. Now, spilling is a process of copying the data from memory buffer to disc when the content of the buffer reaches a certain threshold size. By default, a background thread starts spilling the contents from memory to disc after 80% of the buffer size is filled. Therefore, for a 100 MB size buffer the spilling will start after the content of the buffer reach a size of 80 MB.

## **What is a distributed cache in MapReduce Framework?**

Distributed Cache can be explained as, a facility provided by the MapReduce framework to cache files needed by applications. Once you have cached a file for your job, Hadoop framework will make it available on each and every data nodes where your map/reduce tasks are running. Therefore, one can access the cache file as a local file in your Mapper or Reducer job.

## **What is a combiner and where you should use it?**

Combiner is like a mini reducer function that allow us to perform a local aggregation of map output before it is transferred to reducer phase. Basically, it is used to optimize the network bandwidth usage during a MapReduce task by cutting down the amount of data that is transferred from a mapper to the reducer.

## **Why the output of map tasks are stored (spilled ) into local disc and not in HDFS?**

The outputs of map task are the intermediate key-value pairs which is then processed by reducer to produce the final aggregated result. Once a MapReduce job is completed, there is no need of the intermediate output produced by map tasks. Therefore, storing these intermediate output into HDFS and replicate it will create unnecessary overhead.

## **What happens when the node running the map task fails before the map output has been sent to the reducer?**

In this case, map task will be assigned a new node and whole task will be run again to re-create the map output.

## **What is the role of a MapReduce Partitioner?**

A partitioner divides the intermediate key-value pairs produced by map tasks into partition. The total number of partition is equal to the number of reducers where each partition is processed by the corresponding reducer. The partitioning is done using the hash function based on a single key or group of keys. The default partitioner available in Hadoop is HashPartitioner.

## **How can we assure that the values regarding a particular key goes to the same reducer?**

By using a partitioner we can control that a particular key – value goes to the same reducer for processing.

## **What is the difference between Input Split and HDFS block?**

HDFS block defines how the data is physically divided in HDFS whereas input split defines the logical boundary of the records required for processing it.

## **What is the purpose of TextInputFormat?**

TextInputFormat is the default input format present in the MapReduce framework. In TextInputFormat, an input file is produced as keys of type LongWritable (byte offset of the beginning of the line in the file) and values of type Text (content of the line).

## **What is the role of RecordReader in Hadoop MapReduce?**

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

## **When should you use SequenceFileInputFormat?**

SequenceFileInputFormat is an input format for reading within sequence files. It is a specific compressed binary file format which is optimized for passing the data between the outputs of one “MapReduce” job to the input of some other “MapReduce” job.

## **What is an identity Mapper and Identity Reducer?**

Identity mapper is the default mapper provided by the Hadoop framework. It runs when no mapper class has been defined in the MapReduce program where it simply passes the input key – value pair for the reducer phase.

Like Identity Mapper, Identity Reducer is also the default reducer class provided by the Hadoop, which is automatically executed if no reducer class has been defined. It also performs no computation or process, rather it just simply write the input key – value pair into the specified output directory.

## **What is a map side join?**

Map side join is a process where two data sets are joined by the mapper.

## **What are the advantages of using map side join in MapReduce?**

* Map-side join helps in minimizing the cost that is incurred for sorting and merging in the shuffle and reduce stages.
* Map-side join also helps in improving the performance of the task by decreasing the time to finish the task.

## **What is reduce side join in MapReduce?**

As the name suggests, in the reduce side join, the reducer is responsible for performing the join operation. It is comparatively simple and easier to implement than the map side join as the sorting and shuffling phase sends the values having identical keys to the same reducer and therefore, by default, the data is organized for us.

♣**Tip**: I would suggest you to go through a dedicated blog on [***reduce side join***](https://www.edureka.co/blog/mapreduce-example-reduce-side-join/) in MapReduce where the whole process of reduce side join is explained in detail with an example.

## **What do you know about NLineInputFormat?**

NLineInputFormat splits ‘n’ lines of input as one split.

## **Is it legal to set the number of reducer task to zero? Where the output will be stored in this case?**

Yes, It is legal to set the number of reduce-tasks to zero if there is no need for a reducer. In this case the outputs of the map task is directly stored into the HDFS which is specified in the setOutputPath(Path).

## **Is it necessary to write a MapReduce job in Java?**

No, MapReduce framework supports multiple languages like Python, Ruby etc.

## **How do you stop a running job gracefully?**

One can gracefully stop a MapReduce job by using the command: hadoop job -kill JOBID

## **How will you submit extra files or data ( like jars, static files, etc. ) for a MapReduce job during runtime?**

The distributed cache is used to distribute large read-only files that are needed by map/reduce jobs to the cluster. The framework will copy the necessary files from a URL on to the slave node before any tasks for the job are executed on that node. The files are only copied once per job and so should not be modified by the application.

## **How does inputsplit in MapReduce determines the record boundaries correctly?**

RecordReader is responsible for providing the information regarding record boundaries in an input split.

## **How do reducers communicate with each other?**

This is a tricky question. The “MapReduce” programming model does not allow “reducers” to communicate with each other. “Reducers” run in isolation.

## **Define Speculative Execution**

If a node appears to be executing a task slower than expected, the master node can redundantly execute another instance of the same task on another node. Then, the task which finishes first will be accepted whereas other tasks will be killed. This process is called speculative execution.

#### **What do MapReduce framework consists of?**

MapReduce framework is used to write applications for processing large data in parallel on large clusters of commodity hardware.

**It consists of:**

**ResourceManager (RM)**

* + - Global resource scheduler
    - One master RM

**NodeManager (NM)**

* + - One slave NM per cluster-node.

**Container**

* + - RM creates Containers upon request by AM
    - The application runs in one or more containers

**ApplicationMaster (AM)**

* + - One AM per application
    - Runs in Container

#### **What are the two main components of ResourceManager?**

* + - **Scheduler**

It allocates the resources (containers) to various running applications based on resource availability and configured shared policy.

* + - **ApplicationManager**

It is mainly responsible for managing a collection of submitted applications

#### **What is a Hadoop counter?**

Hadoop Counters measures the progress or tracks the number of operations that occur within a MapReduce job. Counters are useful for collecting statistics about MapReduce job for application-level or quality control.

#### **What are the steps involved to submit a Hadoop job?**

Steps involved in Hadoop job submission:

* + - Hadoop job client submits the job jar/executable and configuration to the ResourceManager.
    - ResourceManager then distributes the software/configuration to the slaves.
    - ResourceManager then scheduling tasks and monitoring them.
    - Finally, job status and diagnostic information are provided to the client.

#### **How does MapReduce framework view its input internally?**

It views the input data set as a set of pairs and processes the map tasks in a completely parallel manner.

#### **What are the basic parameters of Mapper?**

The basic parameters of Mapper are listed below:

1. LongWritable and Text
2. Text and IntWritable

#### **What are Writables and explain its importance in Hadoop?**

* + - Writables are interfaces in Hadoop. They act as a wrapper class to almost all the primitive data types of Java.
    - A serializable object which executes a simple and efficient serialization protocol, based on DataInput and DataOutput.
    - Writables are used for creating serialized data types in Hadoop.

#### **Why comparison of types is important for MapReduce?**

* + - It is important for MapReduce as in the sorting phase the keys are compared with one another.
    - For Comparison of types, WritableComparable interface is implemented.

#### **What are the methods used for restarting the NameNode in Hadoop?**

The methods used for restarting the NameNodes are the following:

* + - You can use **/sbin/hadoop-daemon.sh stop namenode** command for stopping the NameNode individually and then start the NameNode using **/sbin/hadoop-daemon.sh start namenode.**
    - Use **/sbin/stop-all.sh** and then use**/sbin/start-all.sh** command for stopping all the demons first and then start all the daemons.

These script files are stored in the sbin directory inside the Hadoop directory store.

#### **What are the different modes in which Hadoop can run?**

* + - **Standalone Mode(local mode) -** This is the default mode where Hadoop is configured to run. In this mode, all the components of Hadoop such as DataNode, NameNode, etc., run as a single Java process and useful for debugging.
    - **Pseudo Distributed Mode(Single-Node Cluster) -** Hadoop runs on a single node in a pseudo-distributed mode. Each Hadoop daemon works in a separate Java process in Pseudo-Distributed Mode, while in Local mode, each Hadoop daemon operates as a single Java process.
    - **Fully distributed mode (or multiple node cluster) -** All the daemons are executed in separate nodes building into a multi-node cluster in the fully-distributed mode.

#### **Why aggregation cannot be performed in Mapperside?**

* + - We cannot perform Aggregation in mapping because it requires sorting of data, which occurs only at Reducer side.
    - For aggregation, we need the output from all the mapper functions, which is not possible during the map phase as map tasks will be running in different nodes, where data blocks are present.

#### **What is Identity Mapper?**

* + - Identity Mapper is a default Mapper class which automatically works when no Mapper is specified in the MapReduce driver class.
    - It implements mapping inputs directly into the output.
    - IdentityMapper.class is used as a default value when JobConf.setMapperClass is not set.

#### **What are the phases of MapReduce Reducer?**

The MapReduce reducer has three phases:

* + - **Shuffle phase -** In this phase, the sorted output from a mapper is an input to the Reducer. This framework will fetch the relevant partition of the output of all the mappers by using HTTP.
    - **Sort phase -** In this phase, the input from various mappers is sorted based on related keys. This framework groups reducer inputs by keys. Shuffle and sort phases occur concurrently.
    - **Reduce phase -** In this phase, reduce task aggregates the key-value pairs after shuffling and sorting phases. The OutputCollector.collect() method, writes the output of the reduce task to the Filesystem.

#### **What is the purpose of MapReduce Partitioner in Hadoop?**

The MapReduce Partitioner manages the partitioning of the key of the intermediate mapper output. It makes sure that all the values of a single key pass to same reducers by allowing the even distribution over the reducers.

#### **How will you write a custom partitioner for a Hadoop MapReduce job?**

* + - Build a new class that extends Partitioner Class
    - Override the get partition method in the wrapper.
    - Add the custom partitioner to the job as a config file or by using the method set Partitioner