**Answers**

**1. What is the difference between TextInputFormat and KeyValueInputFormat class?**

Ans:

The TextInputFormat class converts every row of the source file into key/value types where the BytesWritable key represents the offset of the record and the Text value represents the entire record itself.

TextInputFormat 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. TextInputFormat is the default file format in Hadoop.

The KeyValueTextInputFormat is an extended version of TextInputFormat, which is useful when we have to fetch every source record as Text/Text pair where the key/value is populated from the record by splitting the record with a fixed delimiter.

KeyValueTextInputFormat 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.

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

Ans:  It is invoked by the Hadoop framework by running getInputSplit() method of the Input format class.

**3. 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 for each file?**

**Ans:** Hadoop will make 5 splits as follows:

- 1 split for 64K files

- 2 splits for 65MB files

- 2 splits for 127MB files

**4. After the Map phase finishes, the Hadoop framework performs “Partitioning, Shuffle and sort”. Explain each event in brief.**

**Ans:**

**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.

**5. What is a Combiner?**

**Ans:** 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 on a given node. The output from the Combiner is then sent to the Reducers, instead of the output from the Mappers.

**6. What is Hadoop streaming?**

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

**7. What are the most commonly defined input formats in Hadoop and explain each in brief.**

**Ans:** Hadoop can process many different types of data formats, from flat text files to databases.

Some commonly used Input formats are:-

**Input Split**

An input split is a chunk of the input that is processed by a single map. Each map processes a single split. Each split is divided into records, and the map processes each record — a key-value pair — in turn. Splits and records are logical: there is nothing that requires them to be tied to files, In a database context, a split might correspond to a range of rows from a table and a record to a row in that range (this is precisely the case with DBInputFormat, which is an input format for reading data from a relational database).

Input splits are represented by the Java class InputSplit is in the org.apache.hadoop.mapreduce package.

An InputSplit has a length in bytes and a set of storage locations, which are just hostname strings. A split doesn’t contain the input data instead it is just a reference to the data. The storage locations are used by the MapReduce system to place map tasks as close to the split’s data as possible, and the size is used to order the splits so that the largest get processed first, in an attempt to minimize the job runtime (this is an instance of a greedy approximation algorithm).

**Record Reader**

The client running the job calculates the splits for the job by calling getSplits(), then sends them to the application master, which uses their storage locations to schedule map tasks that will process them on the cluster. The map task passes the split to the createRecordReader() method on InputFormat to obtain a RecordReader for that split. A RecordReader is little more than an iterator over records, and the map task uses one to generate record key-value pairs, which it passes to the map function. After running setup(), the nextKeyValue() is called repeatedly on the Context (which delegates to the identically named method on the RecordReader) to populate the key and value objects for the mapper. The key and value are retrieved from the RecordReader by way of the Context and are passed to the map() method for it to do its work. When the reader gets to the end of the stream, the nextKeyValue() method returns false, and the map task runs its cleanup() method and then completes.

**FileInputFormat**

FileInputFormat is the base class for all implementations of InputFormat that use files as their data source .It provides two things: a place to define which files are included as the input to a job, and an implementation for generating splits for the input files. The job of dividing splits into records is performed by subclasses.

**8. Explain what is distributed Cache in MapReduce Framework?**

**Ans:** DistributedCache is a facility provided by the Map-Reduce framework to cache files (text, archives, jars etc.) needed by applications. Applications specify the files, via urls (hdfs:// or http://) to be cached via the [JobConf](https://hadoop.apache.org/docs/r2.4.1/api/org/apache/hadoop/mapred/JobConf.html). The DistributedCache assumes that the files specified via urls are already present on the [FileSystem](https://hadoop.apache.org/docs/r2.4.1/api/org/apache/hadoop/fs/FileSystem.html) at the path specified by the url and are accessible by every machine in the cluster. The framework will copy the necessary files on to the slave node before any tasks for the job are executed on that node. Its efficiency stems from the fact that the files are only copied once per job and the ability to cache archives which are un-archived on the slaves.

DistributedCache can be used to distribute simple, read-only data/text files and/or more complex types such as archives, jars etc. Archives (zip, tar and tgz/tar.gz files) are un-archived at the slave nodes. Jars may be optionally added to the classpath of the tasks, a rudimentary software distribution mechanism. Files have execution permissions. In older version of Hadoop Map/Reduce users could optionally ask for symlinks to be created in the working directory of the child task. In the current version symlinks are always created. If the URL does not have a fragment the name of the file or directory will be used. If multiple files or directories map to the same link name, the last one added, will be used. All others will not even be downloaded. DistributedCache tracks modification timestamps of the cache files. Clearly the cache files should not be modified by the application or externally while the job is executing.

**9. Explain what happens in textinputformat?**

Ans: In textinputformat, each line in the text file is a record. [Value](https://plus.google.com/s/%23Value) is the content of the line while [Key](https://plus.google.com/s/%23Key) is the byte offset of the line. For Example: Key: [longWritable](https://plus.google.com/s/%23longWritable), Value: text

**10. Explain what is Sequencefileinputformat?**

**Ans:** Sequencefileinputformat is an input format for reading in sequence files. It is a specific compressed binary file format which is optimized for passing the data between the output of one MapReduce job to the input of some other MapReduce job.