**ACD\_BDD\_Session\_12\_Assignment\_1\_Main**

**Problem Statement**

**Give a brief answers to the questions below:**

**1. What is the purpose of RecordReader in Hadoop?**

-> RecordReader reads <key,value> pairs from an InputSplit.

RecordReader, typically, converts the byte-oriented view of the input, provided by the InputSplit, and presents a record-oriented view for the Mapper & Reducer tasks for processing. It thus assumes the responsibility of processing record boundaries and presenting the tasks with keys and values.

**2. What happens if the number of reducers is 0?**

-> The output of the Map task goes directly to the FileSystem, into the output path set by setOutputPath(path). The framework does not sort the Map output before writing them to the FileSystem.

**3. What is meant by Map-side and Reduce-side join in Hadoop?**

-> Map-Side join:

A map-side join between large inputs works by performing the join before the data reaches the map function. For this to work, though, the inputs to each map ,must be partitioned and sorted in a particular way. Each input dataset must be divided into the same numbers of partitions, and it must be sorted by the same key(the join key) in each source. All the records for a particular key must reside in the same partition. This may sound like a strict requirement (and it is), but it actually fits the description of the output of a MapReduce job.

A map-side join can be used to join the outputs of several jobs that had the same number of reducers, the same keys, and output files that are not splitable (by virtue of being smaller than an HDFS block or being gzip compressed).

CompositeInputFormat from the org.apache.hadoop.mapreduce.join package is used to run a map-side join.

Reduce-Side join:

A reduce-side join is more general than a map-side join, in that the input dataset's don't have to be structured in any particular way, but it is less efficient because both dataset's have to go through the MapReduce shuffle. The basic idea is that the mapper tags each record with its source and uses the join key as the map output key, so the records with the same key are brought together in the reducer.

**4. What is the significance of conf.setMapper class?**

-> Conf.setMapper class sets the mapper class and everything related to map job such as reading a data and generating a key-value pair out of the mapper.

**5. Give an example scenario on the usage of counters.**

-> There are often things that you would like to know about the data you are analyzing but that are peripherals to the analysis you are performing. For example if you were counting invalid records and discovered that the proportion of invalid records in the whole dataset was very high, you might be prompted to check why so many records were being marked as invalid - perhaps there is a bug in the part of the program that detects invalid records? Or if the data was of poor quality and genuinely did have very many records, after discovering this, you might decide to increase the size of the dataset so that the number of good records was large enough for meaningful analysis.

Counters are a useful channel for gathering statistics about the job: for quality control or for application level statistics. They are also useful for problem diagnosis. If you are tempted to put a log message into your map or reduce task, it is often better to see whether you can use a counter instead to record that a particular condition occurred. In addition to counter values being much easier to retrieve than log output for large distributed jobs, you get a record of the number of times that condition occurred, which is more work to obtain from a set of logfiles.

**6. Elaborate some problems which can only be solved by MapReduce and cannot be solved by PIG?**

**7. In what kind of scenarios, MR jobs will be more useful than PIG?**

-> Let us take a scenario where we want to count the population in two cities. I have a data set and sensor list of different cities. I want to count the population by using one mapreduce for two cities. Let us assume that one is Bangalore and the other is Noida. So I need to consider key of Bangalore city similar to Noida through which I can bring the population data of these two cities to one reducer. The idea behind this is somehow I have to instruct map reducer program – whenever you find city with the name ‘Bangalore‘ and city with the name ‘Noida’, you create the alias name which will be the common name for these two cities so that you create a common key for both the cities and it get passed to the same reducer. For this, we have to write custom partitioner.

In mapreduce when you create a ‘key’ for city, you have to consider ‘city’ as the key. So, whenever the framework comes across a different city, it considers it as a different key. Hence, we need to use customized partitioner. There is a provision in mapreduce only, where you can write your custom partitioner and mention if city = bangalore or noida then pass similar hashcode. However, we cannot create custom partitioner in Pig. As Pig is not a framework, we cannot direct execution engine to customize the partitioner. In such scenarios, MapReduce works better than Pig.

**8. What are combiners and when are these used in a MapReduce job?**

-> Many MapReduce jobs are limited by the bandwidth available on the cluster, so it pays to minimize the data transferred between map and reduce tasks. Hadoop allows the user to specify a combiner function to be run on the map output, and the combiner function's output forms the input to the reducer function. Because the combiner function is an optimization, Hadoop does not provide a guarantee of how many times it will call it for a particular map output record, if at all. In other words, calling the combiner function zero, one, or many times should produce the same output from the reducer.