**Q1-What are the uses of counters?**

Answer:

1. The MapReduce framework provides Counters as an efficient mechanism for tracking the occurrences of global events within the map and reduces the phases of jobs.

2. For example, a typical MapReduce job will kick off several mapper instances, one for each block of the input data, all running the same code. These instances are part of the same job, but run independent of one another.

3. Counters allow a developer to track aggregated events from all of those separate instances.

4. A more concrete use of Counters can be found in the MapReduce framework itself. Each MapReduce job defines several standard Counters. The output of these Counters can be found in the job details of the Job Tracker web UI.

**Q2-MR Unit testing is based on?**

Answer:

MRUnit testing framework is based on **JUnit** and it can test Map Reduce programs written on 0.20 0.23.x , 1.0.x , 2.x version of Hadoop.

**Q3-How testing is useful in industry?**

Answer:

1. In production, when you are working with huge amount of data, it is advisable not to run the job against the complete data.

2. The Hadoop job might run for hours and then fail. In such case, you need to debug the code, re-compile the logic and re-run it again.

3. This creates unnecessary delay in data processing. Also, chances are there that other jobs might depend/start on completion of your job.

4. If your Hadoop job doesn’t finish on time, it might hamper overall analysis. So, it always advisable to test the logic before putting it into production.

**Q4-Mapreduce Task Counters,File system counters,Job Counter.**

Answer:

**1. Task Counters**

Task counters gather information about tasks over the course of their execution, and the results are aggregated over all the tasks in a job. For example, the MAP\_INPUT\_RECORDS counter counts the input records read by each map task and aggregates over all map tasks in a job, so that the final figure is the total number of input records for the whole job.

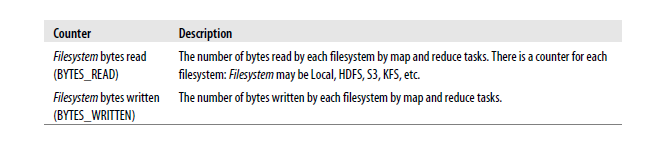
Below are the list of important Task counters maintained by Hadoop:



**2. File system counters**

File system countres track 2 main details, number of bytes read by the file system and number of bytes written.

Below are the name and description of the file system counters:

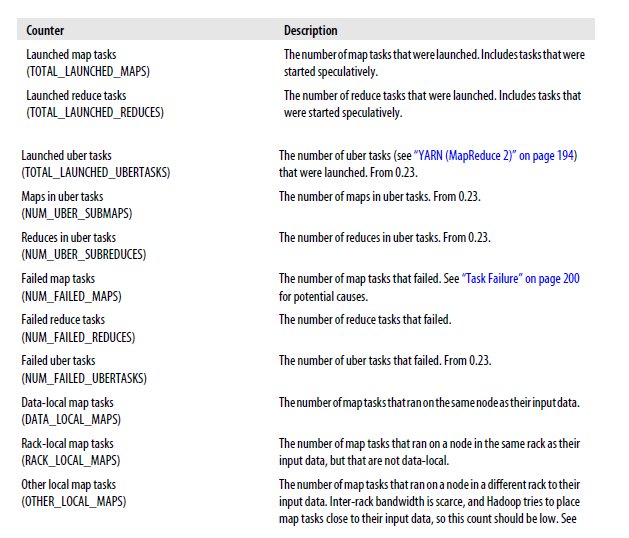


**3. Job Counters**

Job counters are maintained by the jobtracker (or application master in

YARN), so they don’t need to be sent across the network, unlike all other counters, including user-defined ones. They measure job-level statistics, not values that change while a task is running. For example, TOTAL\_LAUNCHED\_MAPS counts the number of map tasks that were launched over the course of a job (including ones that failed).

Below are the list of important Job counters maintained by Hadoop:



**Q5-Raw comparator VS Writable Comparator**

Answer:

|  |  |
| --- | --- |
| Raw comparator | Writable Comparable |
| Keys are compared using their corresponding raw bytes. | Keys are compared using you will deserialized objects. |
| It is faster as compared to Writable Comparable | It is slower as compared to Raw comparator. |

**Q6- Partitioner, Sort comparator, Group comparator.**

Answer:

**PARTITIONER:**

1. A partitioner works like a condition in processing an input dataset. The partition phase takes place after the Map phase and before the Reduce phase.

2. The number of partitioners is equal to the number of reducers. That means a partitioner will divide the data according to the number of reducers. Therefore, the data passed from a single partitioner is processed by a single Reducer.

3. HashPartitioner is the default partitioner in Hadoop, which creates one Reduce task for each unique “key”.  All the values with the same key goes to the same instance of your reducer, in a single call to the reduce function.

4. If user is interested to store a particular group of results in different reducers, then the user can write his own partitioner implementation. It can be general purpose or custom made to the specific data types or values that you expect to use in user application.

5. Custom Partitioner is a process that allows you to store the results in different reducers, based on the user condition. By setting a partitioner to partition by the key, we can guarantee that, records for the same key will go to the same reducer. A partitioner ensures that only one reducer receives all the records for that particular key.

**SORT COMPARATOR:**

SortComparator decides how keys will be sorted in input of reduce. By default it uses natural ordering.

**GROUP COMPARATOR:**

Group Comparator decides which map output keys will be united (grouped) into one key, and of course all collections of values will be grouped too. Usually it takes a first key as the only one for summary collection