Built-In counters

Hadoop maintains some built-in counters for every job, which report various metrics for your job. For example, there are counters for the number of bytes and records processed, which allows us to confirm that the expected amount of input was consumed and the expected amount of output was produced.etc

Built in counters are of three types:

1. Mapreduce Task Counters
2. File system counters
3. Job Counters

1.Defining Task Counters in Mapreduce

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. Etc

Below are the list of important Task counters maintained by Hadoop

2.Defining 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.

Source: Hadoop Definitive Guide

BYTES_READ counter is tracked by File Input Format

Bytes read (BYTES_READ) :The number of bytes read by map tasks via the FileInputFormat.

BYTES_WRITTEN counter is tracked by File Output Format

Bytes written (BYTES_WRITTEN) :The number of bytes written by map tasks (for map-only jobs) or reduce tasks via the FileOutputFormat.

3.Defining 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

Source: Hadoop Definitive Guide

Custom Counters

Introduction:

Apart from this Built-in counters in Mapreduce allows us to create our own set of counters which can be incremented as desired by the user in mapper or reducer for some statistical research.

Counters are defined by a Java enum, which serves to group related counters.

A job may define an N number of enums, each with an N number of fields. The name of the enum is the group name, and the enum's fields are the counter names.

Counters are global: the MapReduce framework aggregates them across all maps and reduces to produce a total at the end of the job

We have created a sample use case to demonstrate the use of custom counters

For this example we have considering Inpatient data set below is the description of data set

This is CSV file with 12 fields about the patient and the hospital.

To demonstrate the use of counters we will create counter to count number of

PULMONARY_DISEASE and

BRONCHITIS_ASTHMA

Case are diogonized in the state CA-(California)

In this example We will just create the Map only job , we are concentrating only on implemntaion of counters , in the same example we can perform our other map related operations

Creating the Custom counters

Step 1 : Create the ENUM for counters.

Since we are counting 2 types of diognaisis ENUM will become the group and fields will become the counters.

```
public enum DIAGNOSIS_COUNTER {

PULMONARY_DISEASE,

BRONCHITIS_ASTHMA

};
```

Step 2 : Find the counters in Mapper and increase the counter as

```
 if(dignosis.contains("PULMONARY") && state.contains("CA")){
context.getCounter(DIAGNOSIS_COUNTER.PULMONARY_DISEASE).increment(1);

}

if(dignosis.contains(" BRONCHITIS") && state.contains("CA") ){
context.getCounter(DIAGNOSIS_COUNTER.BRONCHITIS_ASTHMA).increment(1);

}

public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException {

Configuration conf=new Configuration();

Job job=new Job(conf,"counters");

job.setJarByClass(Counters_demo.class);

job.setMapperClass(Map.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

job.setNumReduceTasks(0);

TextInputFormat.addInputPath(job, new Path(args[0]));

TextOutputFormat.setOutputPath(job, new Path(args[1]));

Path out=new Path(args[1]);

out.getFileSystem(conf).delete(out);
job.waitForCompletion(true);

// get all the job related counters

Counters cn=job.getCounters();

// Find the specific counters that you want to print

Counter c1=cn.findCounter(DIAGNOSIS_COUNTER.PULMONARY_DISEASE);

System.out.println(c1.getDisplayName()+":"+c1.getValue());

Counter c2=cn.findCounter(DIAGNOSIS_COUNTER.BRONCHITIS_ASTHMA);

System.out.println(c2.getDisplayName()+":"+c2.getValue());

/* We can get all the available counters from CounterGroup instance and
print them all in loop*/

for (CounterGroup group : cn) {

System.out.println("* Counter Group: " + group.getDisplayName() + " (" +
group.getName() + ")");

System.out.println(" number of counters in this group: " + group.size());

for (Counter counter : group) {

System.out.println(" - " + counter.getDisplayName() + ": " +
counter.getName() + ": "+counter.getValue());

}

}

}

}
```

Step 3 :

By using the above use case we have got the details of all the types of counters discussed and refer the below screen shot for the same:

File system Counters of the Job:

File input format Counter BYTES_READ:

File output format Counter BYTES_WRITTEN:

Job Counters:

Task Counters:

Custom Counters :

I hope this blog helped you in understanding the concepts of counters and the way it is implemented in Hadoop Framework.