(CS4480 / CS5488) How to debug in Hadoop in a simple way (Optional)

By TA team

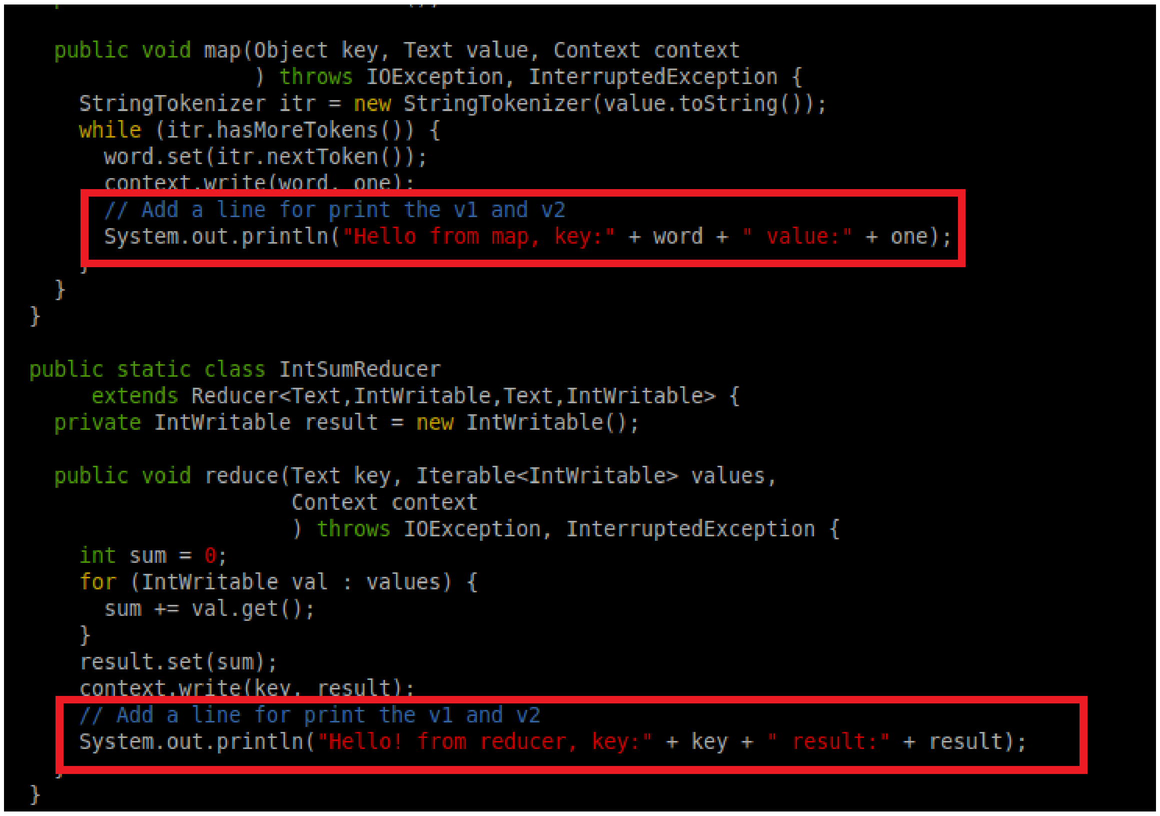
We notice some of you already know the idea of MaxWordCount and TopKWordCount and have troubles on implementation. If you don't know how to debug it, a solution maybe to print out some intermediate results.

The method is very simple: print some variables and check the logs.

For example: WordCount.java

Step 1: Add *System.out.println()* to print the intermediate results

See the figure below: we add two lines in the red rectangle for WordCount.java



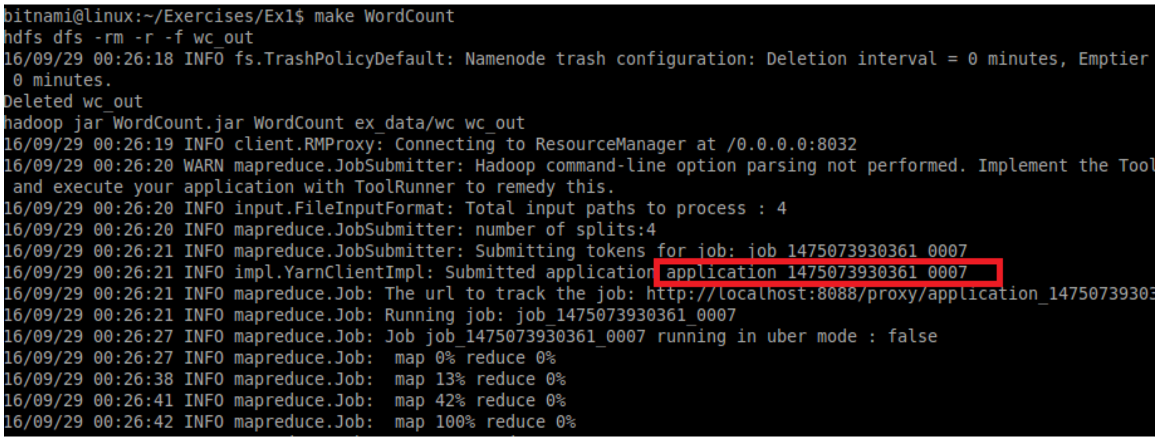
```
  public void map(Object key, Text value, Context context
                  ) throws IOException, InterruptedException {
    StringTokenizer itr = new StringTokenizer(value.toString());
    while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      context.write(word, one);
      // Add a line for print the v1 and v2
      System.out.println("Hello from map, key:" + word + " value:" + one);
    }
  }
}

public static class IntSumReducer
     extends Reducer<Text,IntWritable,Text,IntWritable> {
  private IntWritable result = new IntWritable();

  public void reduce(Text key, Iterable<IntWritable> values,
                     Context context
                     ) throws IOException, InterruptedException {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write(key, result);
    // Add a line for print the v1 and v2
    System.out.println("Hello! from reducer, key:" + key + " result:" + result);
  }
}
```

Step 2: Run the program by command – *make WordCount* and record the application ID.

See the figure below, application ID is application_1475073930361_0007 in this example. Use your own one in your experiment.



```
bitnami@linux:~/Exercises/Ex1$ make WordCount
hdfs dfs -rm -r -f wc_out
16/09/29 00:26:18 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minutes, Emptier
 0 minutes.
Deleted wc_out
hadoop jar WordCount.jar WordCount ex_data/wc wc_out
16/09/29 00:26:19 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
16/09/29 00:26:20 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Implement the Tool
 and execute your application with ToolRunner to remedy this.
16/09/29 00:26:20 INFO input.FileInputFormat: Total input paths to process : 4
16/09/29 00:26:20 INFO mapreduce.JobSubmitter: number of splits:4
16/09/29 00:26:21 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1475073930361_0007
16/09/29 00:26:21 INFO impl.YarnClientImpl: Submitted application application_1475073930361_0007
16/09/29 00:26:21 INFO mapreduce.Job: The url to track the job: http://localhost:8088/proxy/application_147507393036
16/09/29 00:26:21 INFO mapreduce.Job: Running job: job_1475073930361_0007
16/09/29 00:26:27 INFO mapreduce.Job: Job job_1475073930361_0007 running in uber mode : false
16/09/29 00:26:27 INFO mapreduce.Job:  map 0% reduce 0%
16/09/29 00:26:38 INFO mapreduce.Job:  map 13% reduce 0%
16/09/29 00:26:41 INFO mapreduce.Job:  map 42% reduce 0%
16/09/29 00:26:42 INFO mapreduce.Job:  map 100% reduce 0%
```

Step 3: Check every stdout files in */usr/local/hadoop-2.6.0/logs/userlogs/your application ID/container***/*

In this example, we do

cd /usr/local/hadoop-2.6.0/logs/userlogs/application_1475073930361_0007

ls

There are 6 folders which are: **(If you have more than 6, you may check ALL of them**)

container_1475073930361_0007_01_000001/

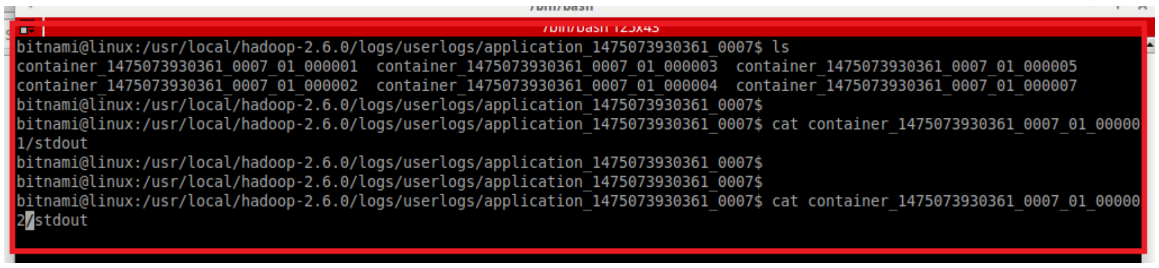
container_1475073930361_0007_01_000002/

container_1475073930361_0007_01_000003/

container_1475073930361_0007_01_000004/

container_1475073930361_0007_01_000005/

container_1475073930361_0007_01_000007/



```
bitnami@linux:/usr/local/hadoop-2.6.0/logs/userlogs/application_1475073930361_0007$ ls
container_1475073930361_0007_01_000001  container_1475073930361_0007_01_000003  container_1475073930361_0007_01_000005
container_1475073930361_0007_01_000002  container_1475073930361_0007_01_000004  container_1475073930361_0007_01_000007
bitnami@linux:/usr/local/hadoop-2.6.0/logs/userlogs/application_1475073930361_0007$
bitnami@linux:/usr/local/hadoop-2.6.0/logs/userlogs/application_1475073930361_0007$ cat container_1475073930361_0007_01_00000
1/stdout
bitnami@linux:/usr/local/hadoop-2.6.0/logs/userlogs/application_1475073930361_0007$
bitnami@linux:/usr/local/hadoop-2.6.0/logs/userlogs/application_1475073930361_0007$
bitnami@linux:/usr/local/hadoop-2.6.0/logs/userlogs/application_1475073930361_0007$ cat container_1475073930361_0007_01_00000
2/stdout
```

Therefore, check all stdout in all of the container* folders

cat container_1475073930361_0007_01_000001/stdout

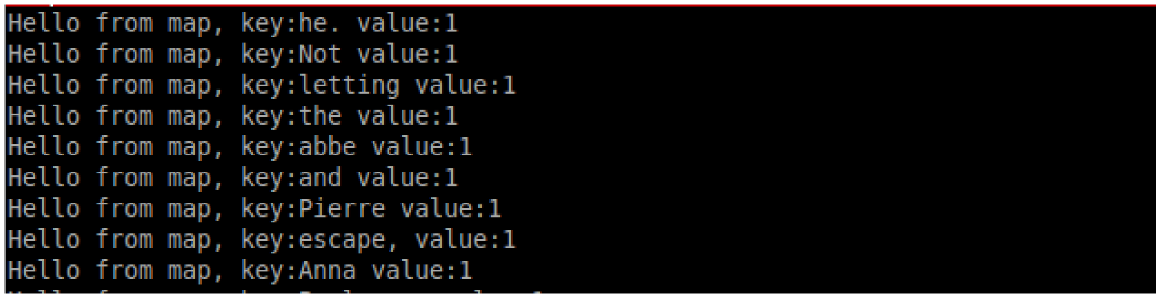
cat container_1475073930361_0007_01_000002/stdout *cat container_1475073930361_0007_01_000003/stdout*

cat container_1475073930361_0007_01_000004/stdout

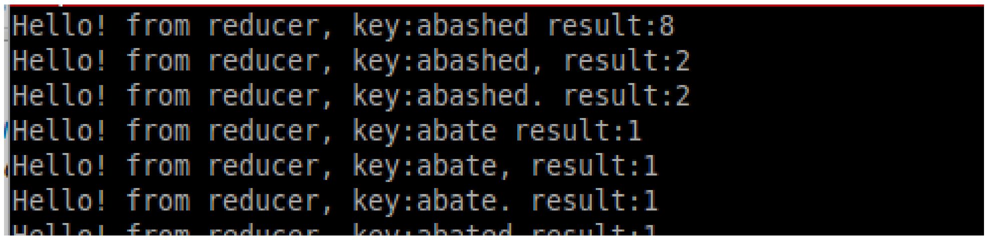
cat container_1475073930361_0007_01_000005/stdout

cat container_1475073930361_0007_01_000007/stdout

We can found map and reduce results we print finally after checking **ALL** folders:



```
Hello from map, key:he. value:1
Hello from map, key:Not value:1
Hello from map, key:letting value:1
Hello from map, key:the value:1
Hello from map, key:abbe value:1
Hello from map, key:and value:1
Hello from map, key:Pierre value:1
Hello from map, key:escape, value:1
Hello from map, key:Anna value:1
```



```
Hello! from reducer, key:abashed result:8
Hello! from reducer, key:abashed, result:2
Hello! from reducer, key:abashed. result:2
Hello! from reducer, key:abate result:1
Hello! from reducer, key:abate, result:1
Hello! from reducer, key:abate. result:1
```

(END)