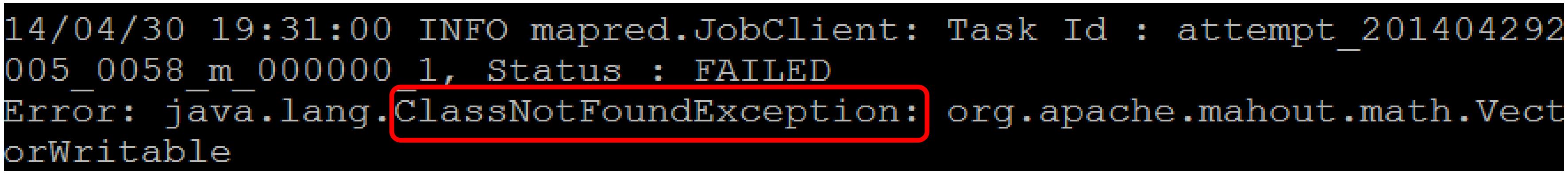
# How to use *libjars* options in Hadoop

Chun-Chen Tu
timtu@umich.edu

# Problems occur when …

- You want to use third-party jar library like mahout-core-0.9-job.jar

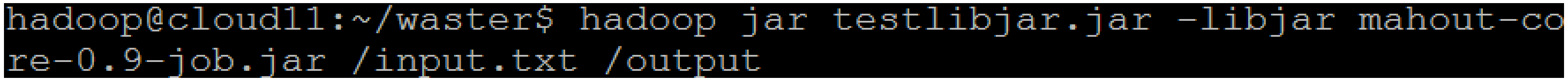


```
14/04/30 19:31:00 INFO mapred.JobClient: Task Id : attempt_201404292
005_0058_m_000000_1, Status : FAILED
Error: java.lang.ClassNotFoundException: org.apache.mahout.math.Vect
orWritable
```

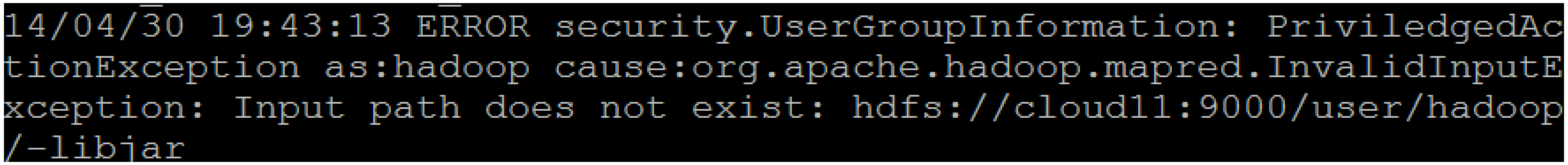
- Here's your solutions:
  - Setup mahout on all the nodes including CLASSPATH … etc.
  - Use –*libjars* options.

# -libjars

- Pass the jar libraries with jobs. Other options like –files, -archives need the same modifications as below.
- When using *-libjars, -files, -archives* option, we need to add GenericOptionsParser in the main funciton.



```
hadoop@cloud11:~/waster$ hadoop jar testlibjar.jar -libjar mahout-co
re-0.9-job.jar /input.txt /output
```



```
14/04/30 19:43:13 ERROR security.UserGroupInformation: PriviledgedAc
tionException as:hadoop cause:org.apache.hadoop.mapred.InvalidInputE
xception: Input path does not exist: hdfs://cloud11:9000/user/hadoop
/-libjar
```

-libjar is misrecognized as an input

# Notes

- We need to use a new Map/Reduce API
  - Old
    - class Map extends MapReduceBase implements Mapper<>
    - class Reduce extends MapReduceBase implements Reducer<>
  - New
    - class Map extends Mapper<>
    - class Reduce extends Reducer<>
- And then add the *GenericOptionsParser* in main
- Our java is called wordcount_libjar.java. This is modified from the wordcount.java example. Just to show how -libjar works.

```
package org.myorg;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.util.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat
import org.apache.mahout.math.*;
```

Class VectorWritable comes from mahout jar library. We add this line to test if the third-party jar is successfully used.

```
public class wordcount_libjar
{
    public static class Map extends Mapper<LongWritable, Text, Text, IntWritable>
    {
                    ... Map Codes ...
                    VectorWritable vec = new VectorWritable();
    }
    public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>
    {
                    ... Reduce Codes ...
    }
    public static void main(String[] args) throws Exception
    {
                    ... GenericOptionsParser...
                    ... Job Configuration ...
                    job.waitForCompletion(true);
    }
}
```

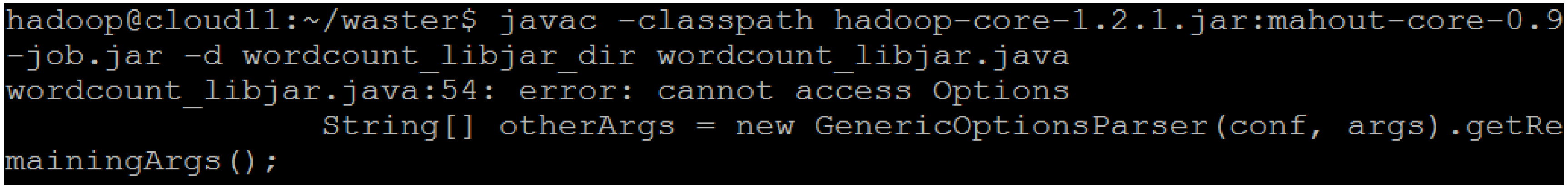
```
package org.myorg;

public class mywordcount
{
  public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>

  public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>

  public static void main(String[] args) throws Exception
  {
       Configuration conf = new Configuration();
        String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
        if (otherArgs.length != 2)
        {
           System.err.println("Usage: wordcount <in> <out>");
           System.exit(2);
        }

        Job job = new Job(conf);
        job.setJarByClass(wordcount_libjar.class);
        job.setJobName("Job to test lib jar");
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setMapperClass(Map.class);
        job.setCombinerClass(Reduce.class);
        job.setReducerClass(Reduce.class);
        FileInputFormat.addInputPath(job, new Path(otherArgs[0]));
        FileOutputFormat.setOutputPath(job, new Path(otherArgs[1]));
        job.waitForCompletion(true);
  }
}
```

Use the parser to deal with input arguments.
This time, -libjar won't be misrecognized.

# Compile

The library consists of VectorWritable.class

*mkdir wordcount_libjar_dir. *
*javac –classpath hadoop-core-1.2.1.jar :mahout-core-0.9-job.jar *
*-d wordcount_libjar_dir wordcount_libjar.java*



```
hadoop@cloud11:~/waster$ javac -classpath hadoop-core-1.2.1.jar:mahout-core-0.9
-job.jar -d wordcount_libjar_dir wordcount_libjar.java
wordcount_libjar.java:54: error: cannot access Options
                String[] otherArgs = new GenericOptionsParser(conf, args).getRe
mainingArgs();
```

The error message basically results from missing library jars. When using the parser we need to include commons-cli jar to the classpath. This jar exists under hadoop/lib.

*javac –classpath hadoop-core-1.2.1.jar :mahout-core-0.9-job.jar:commons-cli-1.2.jar *
*-d wordcount_libjar_dir wordcount_libjar.java*

This time, we can successfully compile the code. And then pack it to generate jar file.
*jar -cvf wordcount_libjar.jar -C wordcount_libjar_dir .*

# Execute with -libjar option

```
hadoop jar wordcount_libjar.jar org.myorg.wordcount_libjar -libjars mahout-core-0.9-
job.jar /input.txt /output
```