Chain Mappers

Use Case

edureka!

Version 2.0

Chain Mappers

Table of Contents

Chain Mappers

Problem Statement:

In this, we shall discuss about how to implement Chain Mapper in Map Reduce.

Let us assume we have two mappers mapper1, mapper2. If we want to give Mapper1 output as Mapper2 input or to invoke chained fashion in mappers i.e. the output of the first becomes the input of the second, and so on until the last Mapper, the output of the last Mapper will be written to the task's output then we need to implement ChainMappers.

Important Links:

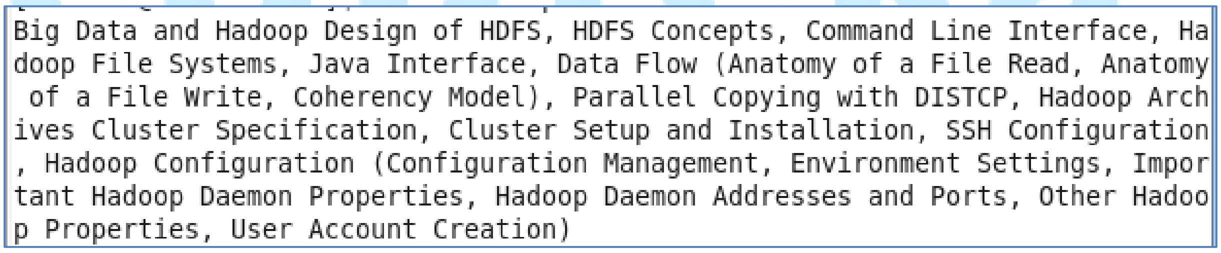
Edureka VM Installation Guide:

Please refer to Installation guide section present in the LMS for accessing the Edureka VM Installation Guide.

Codes:

https://edureka.wistia.com/medias/l6ly6bflhx/download?media_file_id=67295612

Dataset:



```
Big Data and Hadoop Design of HDFS, HDFS Concepts, Command Line Interface, Ha
doop File Systems, Java Interface, Data Flow (Anatomy of a File Read, Anatomy
 of a File Write, Coherency Model), Parallel Copying with DISTCP, Hadoop Arch
ives Cluster Specification, Cluster Setup and Installation, SSH Configuration
, Hadoop Configuration (Configuration Management, Environment Settings, Impor
tant Hadoop Daemon Properties, Hadoop Daemon Addresses and Ports, Other Hadoo
p Properties, User Account Creation)
```

Implementation:

To implement the chain mapper let us consider our simple word count example.

Here we have two Mappers.

Tokenizer – In this mapper we split the input file into words and taking their initial count as 1.

UpperCase – In this mapper we will take the Tokenizer Mapper output as input and then convert those words to upper case.

To chain these two mappers, we use chain Mapper. Finally in the reducer part, we will perform the word count and will return the output.

Tokenizer:

Splitting the input file into words and considering initial count as 1

```
public static class Tokenizer extends Mapper<LongWritable, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {

        String line = value.toString();
        System.out.println("Line:" + line);
        StringTokenizer itr = new StringTokenizer(line);
        while (itr.hasMoreTokens()) {
            word.set(itr.nextToken());
            context.write(word, one);
        }
    }

}
```

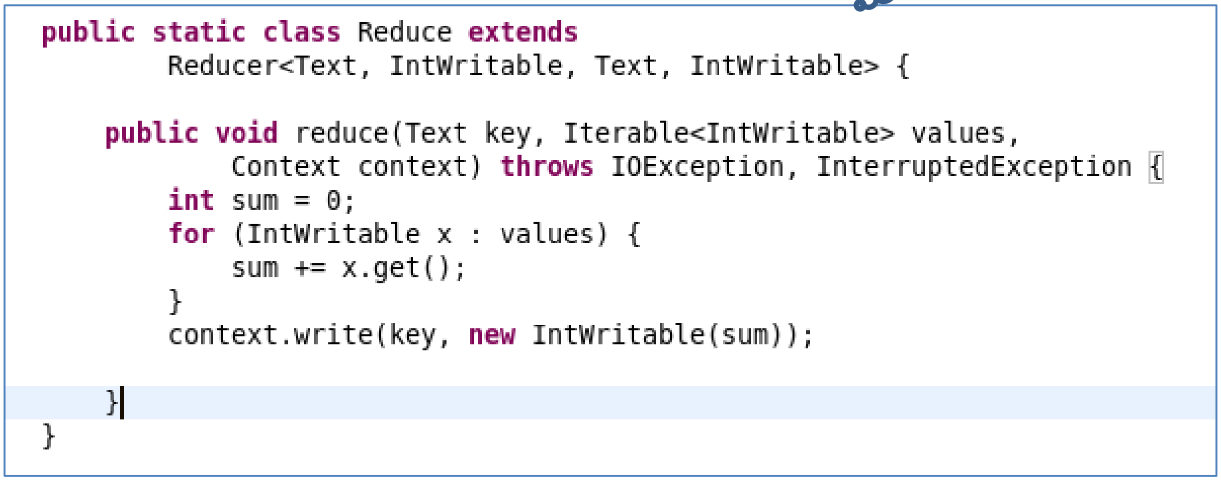
UpperCase:

Converting words to upper case.

```
public static class UpperCase extends Mapper<Text, IntWritable, Text, IntWritable> {

    public void map(Text key, IntWritable value, Context context)
            throws IOException, InterruptedException {

        String word = key.toString().toUpperCase();
        System.out.println("Upper Case:" + word);
        context.write(new Text(word), value);
    }

}
```

Reducer:

Performing word count

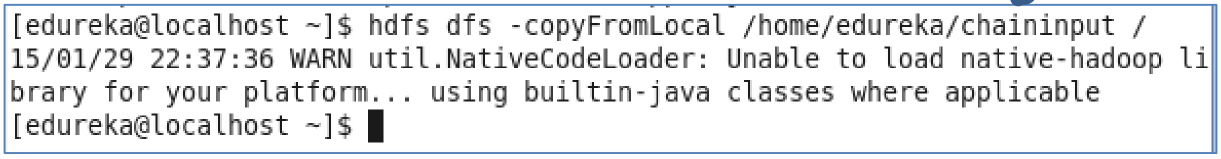


```
public static class Reduce extends
        Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values,
            Context context) throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable x : values) {
            sum += x.get();
        }
        context.write(key, new IntWritable(sum));

    }
}
```

First let us transfer our input file to hdfs.

Command: hdfs dfs -copyFromLocal /home/edureka/chaininput /

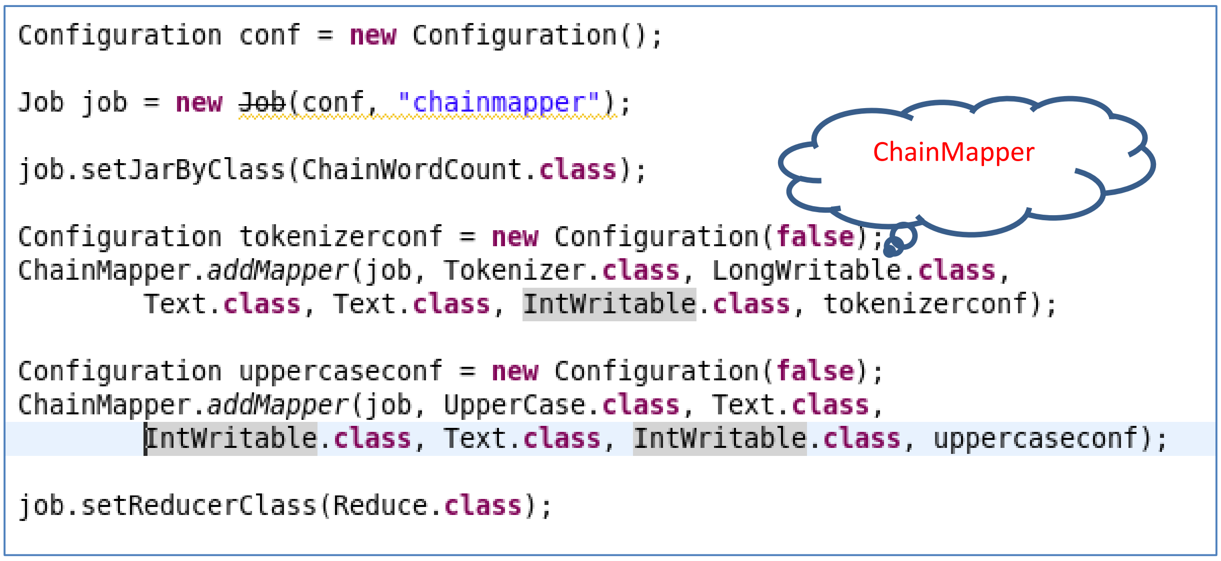
Copying input file to hdfs



```
[edureka@localhost ~]$ hdfs dfs -copyFromLocal /home/edureka/chaininput /
15/01/29 22:37:36 WARN util.NativeCodeLoader: Unable to load native-hadoop li
brary for your platform... using builtin-java classes where applicable
[edureka@localhost ~]$
```

To implement Chain mappers we have to use addMapper() of ChainMapper which is a static class. Below are the arguments for addMapper()

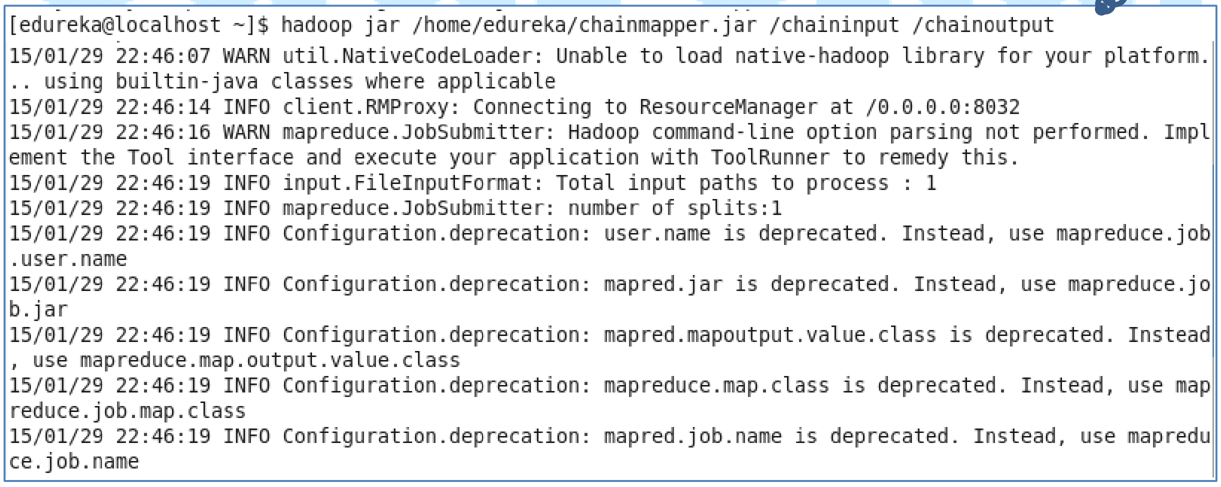
- Object of JobConf ,
- Mapper class,
- inputKeyClass of the Mapper,
- inputValueClass of the Mapper,
- outputKeyClass of the Mapper,
- outputValueClass of the Mapper,
- Boolean byValue,
- JobConf Object for Mapper



```
Configuration conf = new Configuration();

Job job = new ~~Job~~(conf, "chainmapper");

job.setJarByClass(ChainWordCount.class);

Configuration tokenizerconf = new Configuration(false);
ChainMapper.addMapper(job, Tokenizer.class, LongWritable.class,
        Text.class, Text.class, IntWritable.class, tokenizerconf);

Configuration uppercaseconf = new Configuration(false);
ChainMapper.addMapper(job, UpperCase.class, Text.class,
        IntWritable.class, Text.class, IntWritable.class, uppercaseconf);

job.setReducerClass(Reduce.class);
```

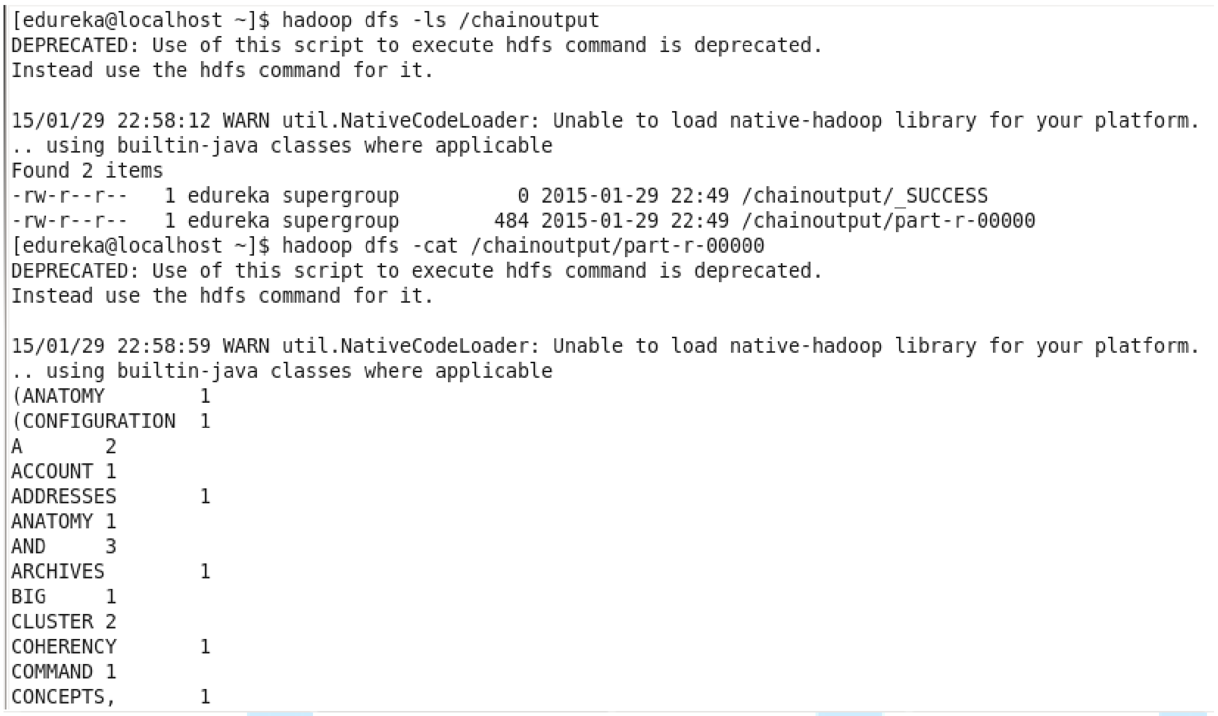
Run the Map reduce program

Command: hadoop jar /home/edureka/chainmapper.jar /chaininput /chainoutput



```
[edureka@localhost ~]$ hadoop jar /home/edureka/chainmapper.jar /chaininput /chainoutput
15/01/29 22:46:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform.
.. using builtin-java classes where applicable
15/01/29 22:46:14 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
15/01/29 22:46:16 WARN mapreduce.JobSubmitter: Hadoop command-line option parsing not performed. Impl
ement the Tool interface and execute your application with ToolRunner to remedy this.
15/01/29 22:46:19 INFO input.FileInputFormat: Total input paths to process : 1
15/01/29 22:46:19 INFO mapreduce.JobSubmitter: number of splits:1
15/01/29 22:46:19 INFO Configuration.deprecation: user.name is deprecated. Instead, use mapreduce.job
.user.name
15/01/29 22:46:19 INFO Configuration.deprecation: mapred.jar is deprecated. Instead, use mapreduce.jo
b.jar
15/01/29 22:46:19 INFO Configuration.deprecation: mapred.mapoutput.value.class is deprecated. Instead
, use mapreduce.map.output.value.class
15/01/29 22:46:19 INFO Configuration.deprecation: mapreduce.map.class is deprecated. Instead, use map
reduce.job.map.class
15/01/29 22:46:19 INFO Configuration.deprecation: mapred.job.name is deprecated. Instead, use mapredu
ce.job.name
```

Let us check the output of MR.



```
[edureka@localhost ~]$ hadoop dfs -ls /chainoutput
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

15/01/29 22:58:12 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform.
.. using builtin-java classes where applicable
Found 2 items
-rw-r--r--   1 edureka supergroup          0 2015-01-29 22:49 /chainoutput/_SUCCESS
-rw-r--r--   1 edureka supergroup        484 2015-01-29 22:49 /chainoutput/part-r-00000
[edureka@localhost ~]$ hadoop dfs -cat /chainoutput/part-r-00000
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

15/01/29 22:58:59 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform.
.. using builtin-java classes where applicable
(ANATOMY        1
(CONFIGURATION  1
A       2
ACCOUNT 1
ADDRESSES       1
ANATOMY 1
AND     3
ARCHIVES        1
BIG     1
CLUSTER 2
COHERENCY       1
COMMAND 1
CONCEPTS,       1
```

We have successfully implemented the Chain Mappers!!!

Reference links:

https://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/lib/ChainMapper.html