Developing Map-Reduce Program for Hadoop

The University of Texas at Dallas

Big Data Course CS6350

Professor: Dr. Latifur Khan

TA: Gbadebo Ayoade(gga110020@utdallas.edu)

Release Date: Spring 2015

```
 //Driver code
      public static void main(String[] args) throws Exception {
            Configuration conf = new Configuration();
            String[] otherArgs = new GenericOptionsParser(conf,
args).getRemainingArgs();
            // get all args
            if (otherArgs.length != 4) {
                  System.err.println("Usage: JoinExample <in> <in2> <out>
<anymovieid>");
                  System.exit(2);
            }


            conf.set("movieid", otherArgs[3]); //setting global data variable for
hadoop

            // create a job with name "joinexc"
            Job job = new Job(conf, "joinexc");
job.setJarByClass(JoinExample.class);


            job.setReducerClass(Reduce.class);

            // OPTIONAL :: uncomment the following line to add the Combiner
            // job.setCombinerClass(Reduce.class);


            MultipleInputs.addInputPath(job, new Path(otherArgs[0]),
TextInputFormat.class ,
                        Map1.class );

            MultipleInputs.addInputPath(job, new
Path(otherArgs[1]),TextInputFormat.class,Map2.class );

            job.setOutputKeyClass(Text.class);
            // set output value type
            job.setOutputValueClass(Text.class);

            //set the HDFS path of the input data
            // set the HDFS path for the output
            FileOutputFormat.setOutputPath(job, new Path(otherArgs[2]));

            job.waitForCompletion(true);


      }
```

I have attached the source file to this lecture.

//The Mapper classes and reducer code

```
//The Mapper classes and  reducer  code
```

```
	public static class Map1 extends Mapper<LongWritable, Text, Text, Text>{
		String mymovieid;

		@Override
		protected void setup(Context context)
				throws IOException, InterruptedException {
			// TODO Auto-generated method stu
			super.setup(context);

			Configuration conf = context.getConfiguration();
			mymovieid = conf.get("movieid"); // to retrieve movieid set in
main method

		}


		private Text rating;
		private Text movieid = new Text();  // type of output key
		public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
			String[] mydata = value.toString().split("::");
			///   System.out.println(value.toString());
			String intrating = mydata[2];
			rating = new Text("rat~" + intrating);
			movieid.set(mydata[1].trim());
			context.write(movieid, rating);


		}


	}


	public static class Map2 extends Mapper<LongWritable, Text, Text, Text>{
		private Text myTitle = new Text();
		private Text movieid = new Text();  // type of output key
		public void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
			String[] mydata = value.toString().split("::");
			System.out.println(value.toString());
			String title = mydata[1];
			myTitle.set("mov~" + title);
			movieid.set(mydata[0].trim());
			context.write(movieid, myTitle);


		}


	}
	//The reducer class
	public static class Reduce extends Reducer<Text,Text,Text,Text> {
		private Text result = new Text();
		private Text myKey = new Text();
		//note you can create a list here to store the values

		public void reduce(Text key, Iterable<Text> values,Context context )
```

```
throws IOException, InterruptedException {


                  for (Text val : values) {
                        result.set(val.toString());
                        myKey.set(key.toString());
                        if(key.toString().trim().compareTo("1")== 0){
                        context.write(myKey,result ); //output only if movied is 1
                        }

                  }


            }
      }
```

Other APIs you will need.
To add file to distributed cache for map side join

This should be added to you driver code.

```
final String NAME_NODE = "hdfs://sandbox.hortonworks.com:8020";
               job.addCacheFile(new URI(NAME_NODE
                       + "/user/hue/users/users.dat"));
```

//added to your mapper class for map side join

```
@Override
            protected void setup(Context context)
                        throws IOException, InterruptedException {
                  // TODO Auto-generated method stu
                  super.setup(context);
                  myMap = new HashMap<String, String >();
                  Configuration conf = context.getConfiguration();
                  movieid = conf.get("movieid"); //for retrieving data you set in
driver code
                  Path[] localPaths = context.getLocalCacheFiles();

                  for(Path myfile:localPaths)
                {
                    String line=null;
                    String nameofFile=myfile.getName();
                    File file =new File(nameofFile+"");
                    FileReader fr= new FileReader(file);
                    BufferedReader br= new BufferedReader(fr);
                    line=br.readLine();
                    while(line!=null)
                    {
                        String[] arr=line.split("::");
                        myMap.put(arr[0], arr[1]); //userid and gender
                    line=br.readLine();
                    }
                }

            }
```

Hadoop also provides setup and cleanup to perform preprocessing and post processing on your data.

Below is a pseudocode
class mapper:
setup():
initialize top ten sorted list
map(key, record):
insert record into top ten sorted list
if length of array is greater-than 10 then
truncate list to a length of 10

cleanup():
for record in top sorted ten list:
emit null,record

class reducer:

```
setup():
        initialize top ten sorted list
        reduce(key, records):
        sort records
        truncate records to top 10
        for record in records:
                emit record

cleanup():
```

You can check page 81 in MapReduce Design Patterns by Donald Miner and Adam Shook for the java code.