MapReduce Program

Class Text

org.apache.hadoop.io

Class Text

java.lang.Object
  └ org.apache.hadoop.io.BinaryComparable
    └ org.apache.hadoop.io.Text

All Implemented Interfaces:

Comparable<BinaryComparable>, Writable, WritableComparable<BinaryComparable>

- This class stores text using standard UTF8 encoding.
- It provides methods to serialize, deserialize, and compare texts at byte level.
- They have the "Writable" interface -they know how to write to a DataOutput stream and read from a DataInput stream - explicitly.
- The type of length is integer and is serialized using zero-compressed format.
- In addition, it provides methods for string traversal without converting the byte array to a string.

StubMapper

```
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class StubMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

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

          String [] words =value.toString().split(" ");
          for(String word: words)
          {
                  context.write(new Text(word), new IntWritable(1));
          }
    /*
     * TODO implement
     */

  }
}
```

StubReducer

```
import java.io.IOException;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class StubReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

  @Override
  public void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {

    /*
     * TODO implement
     */

          Integer count = 0;
          for(IntWritable val:values)
          {
                  count += val.get();
          }
          context.write(key, new IntWritable(count));
  }
}
```

StubDriver

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class StubDriver {

  public static void main(String[] args) throws Exception {

    /*
     * Validate that two arguments were passed from the command line.
     */
    if (args.length != 2) {
      System.out.printf("Usage: StubDriver <input dir> <output dir>\n");
      System.exit(-1);
    }

    /*
     * Instantiate a Job object for your job's configuration.
     */
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");


    /*
     * Specify the jar file that contains your driver, mapper, and reducer.
     * Hadoop will transfer this jar file to nodes in your cluster running
     * mapper and reducer tasks.
     */
    job.setJarByClass(StubDriver.class);
```

StubDriver

```
job.setJarByClass(StubDriver.class);

/*
 * Specify an easily-decipherable name for the job.
 * This job name will appear in reports and logs.
 */
job.setJobName("Stub Driver");

/*
 * TODO implement
 */

job.setMapperClass(StubMapper.class);
job.setReducerClass(StubReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
/*
 * Start the MapReduce job and wait for it to finish.
 * If it finishes successfully, return 0. If not, return 1.
 */
boolean success = job.waitForCompletion(true);
System.exit(success ? 0 : 1);
```

StubTester

```
MapDriver<LongWritable, Text, Text, IntWritable> mapDriver;
ReduceDriver<Text, IntWritable, Text, IntWritable> reduceDriver;
MapReduceDriver<LongWritable, Text, Text, IntWritable, Text, IntWritable> mapReduceDriver;

/*
 * Set up the test. This method will be called before every test.
 */
@Before
public void setUp() {

  /*
   * Set up the mapper test harness.
   */
  StubMapper mapper = new StubMapper();
  mapDriver = new MapDriver<LongWritable, Text, Text, IntWritable>();
  mapDriver.setMapper(mapper);

  /*
   * Set up the reducer test harness.
   */
  StubReducer reducer = new StubReducer();
  reduceDriver = new ReduceDriver<Text, IntWritable, Text, IntWritable>();
  reduceDriver.setReducer(reducer);

  /*
   * Set up the mapper/reducer test harness.
   */
  mapReduceDriver = new MapReduceDriver<LongWritable, Text, Text, IntWritable, Text, IntWritable>();
  mapReduceDriver.setMapper(mapper);
```

Execution

- Copy input file “pg001.txt” inside the project
- Right click the project and “Run as Application”. Select “Run configurations”. Click on arguments tab and type “pg001.txt output/wordcount” in program arguments.
- Right click the project and “Run as Application”. Select “Java applications”
- After execution is complete, Right click the project and click on refresh to see the new output folder. This will have the results. Delete this folder before executing again.