Group B assignment 2

Design a distributed application using map reduce which processes a log file of a system

Here's an example code in Java using MapReduce to process a log file of a system. This code will count the number of occurrences of each unique log level (e.g., INFO, WARNING, ERROR) in the log file.

First, you need to define the mapper and reducer functions:

```
import java.io.IOException;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;


public class LogAnalyzer {


   // Mapper function
   public static class LogMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
      private final static IntWritable one = new IntWritable(1);
      private Text logLevel = new Text();


      public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
         String line = value.toString();
         String[] parts = line.split("\\s+"); // Split log entry by whitespace


         // Check if log entry is valid (contains a log level)
         if (parts.length > 2 && parts[2].matches(".*\\[.*\\].*")) {
            String level = parts[2].replaceAll("\\[|\\]", ""); // Extract log level from log entry
            logLevel.set(level);
            context.write(logLevel, one); // Emit intermediate key-value pair
         }
      }
```

```
    }

    // Reducer function
    public static class LogReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
        public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get(); // Aggregate occurrences of each log level
            }
            context.write(key, new IntWritable(sum)); // Emit final output
        }
    }
}
```

Next, you need to set up the MapReduce job and run it:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.*;
import org.apache.hadoop.mapreduce.lib.output.*;

public class LogAnalyzerJob {
    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Log Analyzer");
        job.setJarByClass(LogAnalyzerJob.class);

        // Set Mapper and Reducer classes
```

```
        job.setMapperClass(LogAnalyzer.LogMapper.class);
        job.setReducerClass(LogAnalyzer.LogReducer.class);

        // Set intermediate key-value classes
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);

        // Set final output key-value classes
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        // Set input and output paths
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        // Run the job and wait for completion
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

To run the code, you need to provide the input log file and output directory as command line arguments:

```
$ hadoop jar LogAnalyzer.jar LogAnalyzerJob /path/to/input/log/file /path/to/output/directory
```

This code will process the log file and output the number of occurrences of each unique log level to the specified output directory. You can modify the mapper and reducer functions to perform different analyses on the log data as per your requirement.