MapReduce:

1] MapReduce in Hadoop to find the number of times each IP accessed the website.

Mapper:

```
import java.io.IOException;
import java.util.List;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import com.google.common.base.Splitter;
import com.google.common.collect.Lists; public class accessMapper extends Mapper<LongWritable,
Text, Text, IntWritable>
{ @Override public void map(LongWritable key, Text value, Context context)throws IOException,
InterruptedException { IntWritable one = new IntWritable(1);
String line = value.toString();
List<String> items = Lists.newArrayList(Splitter.on(',').split(line));
String ipaddress = items.get(0);
String website=items.get(2); Text compositeKey = new Text(ipaddress+ " " + website);
context.write(compositeKey,one); } }
```

Reducer:

```
import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class accessReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException
{
int sum = 0;
for (IntWritable v : values)
{
sum += v.get();
} IntWritable count= new IntWritable(sum);
context.write(key, count);
} }
```

Main Class:

```
import java.io.IOException;
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.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
public class App
{
public static void main( String[] args ) throws IOException
{
Configuration conf =new Configuration();
// Create a new Job
Job job = Job.getInstance(conf, "Word count example");
job.setJarByClass(App.class);
// Specify various job-specific parameters
job.setJobName("myjob");
//set the mapper and reducer
job.setMapperClass(accessMapper.class);
job.setReducerClass(accessReducer.class);
// set the format of mapper and reducer
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
//set the key nd value format of the output
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
//set the output and input format
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
// Submit the job, then poll for progress until the job is complete
try { job.waitForCompletion(true);
} catch (ClassNotFoundException e) {
e.printStackTrace();
} catch (InterruptedException e) {
e.printStackTrace();
}
} }
```

2] Top 10 most visited IP addresses

Mapper:

```
import java.io.IOException;
import java.util.List;
import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import com.google.common.base.Splitter;
import com.google.common.collect.Lists;
public class top10IpAddressMapper extends Mapper<Object, Text, Text, LongWritable>
{
@Override
public void map(Object key, Text value, Context context)throws IOException, InterruptedException {
LongWritable one = new LongWritable(1);
String line = value.toString();
List<String> items = Lists.newArrayList(Splitter.on(',').split(line));
String ipaddress = items.get(0);
Text Key = new Text(ipaddress);
context.write(Key,one);
}
}
```

Reducer:

```
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
public class top10IpAddressReducer extends Reducer<Text, LongWritable, Text, LongWritable> {
public void reduce(Text key, Iterable<LongWritable> values, Context context) throws IOException,
InterruptedException
{
int sum = 0;
for (LongWritable v : values)
{
sum += v.get();
}
LongWritable count= new LongWritable(sum);
context.write(key, count);
}
}
```

Key and Value Swapper class:

```
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
public class KeyValueSwappingMapper extends Mapper<Text, LongWritable, LongWritable,
Text> {
public void map(Text key, LongWritable value, Context context) throws IOException,
InterruptedException {
context.write(value, key);
}
}
```

Main Class:

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
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.input.SequenceFileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.SequenceFileOutputFormat;
public class top10IpAddress
{
public static void main( String[] args ) throws IOException, ClassNotFoundException,
InterruptedException
{
Configuration conf =new Configuration();
Path out = new Path(args[1]);
// Create a new Job
Job job = Job.getInstance(conf, "Top10Ipaddress");
job.setJarByClass(top10IpAddress.class);
job.setJobName("myjob");
job.setMapperClass(top10IpAddressMapper.class);
job.setCombinerClass(top10IpAddressReducer.class);
job.setReducerClass(top10IpAddressReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(LongWritable.class);
job.setOutputFormatClass(SequenceFileOutputFormat.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(out,"Out1"));
if (!job.waitForCompletion(true)) {
System.exit(1);
}
Job job2 = Job.getInstance(conf, "sort by frequency");
job2.setJarByClass(top10IpAddress.class);
job2.setMapperClass(KeyValueSwappingMapper.class);
job2.setNumReduceTasks(1);
job2.setSortComparatorClass(LongWritable.DecreasingComparator.class);
job2.setOutputKeyClass(LongWritable.class);
job2.setOutputValueClass(Text.class);
job2.setInputFormatClass(SequenceFileInputFormat.class);
FileInputFormat.addInputPath(job2, new Path(out, "Out1"));
FileOutputFormat.setOutputPath(job2, new Path(out, "Out2"));
if (!job2.waitForCompletion(true)) {
System.exit(1);
}
}
}
```