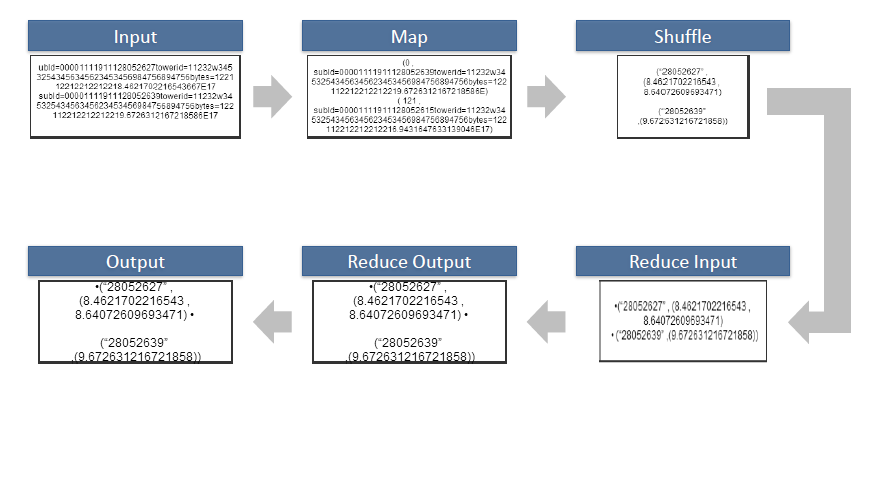
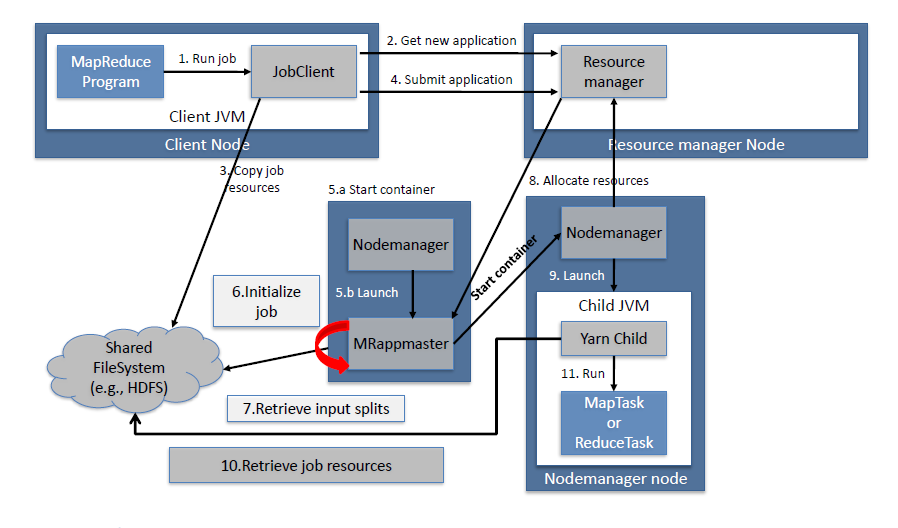
MapReduce

To take the advantage of parallel processing of Hadoop, the query must be in MapReduce form. The MapReduce is a paradigm which has two phases, the mapper phase and the reducer phase. In the Mapper the input is given in the form of key value pair. The output of the mapper is fed to the reducer as input. The reducer runs only after the mapper is over. The reducer too takes input in key value format and the output of reducer is final output.

Steps in Map Reduce

* Map takes a data in the form of pairs and returns a list of <key, value> pairs. The keys will not be unique in this case.
* Using the output of Map, sort and shuffle are applied by the Hadoop architecture. This sort and shuffle acts on these list of <key, value> pairs and sends out unique keys and a list of values associated with this unique key <key, list(values)>.
* Output of sort and shuffle will be sent to reducer phase. Reducer will perform a defined function on list of values for unique keys and Final output will<key, value> will be stored/displayed.

How Many Maps

The size of data to be processed decides the number of maps required. For example, we have 1000 MB data and block size is 64 MB then we need 16 mappers.

Sort and Shuffle

The sort and shuffle occur on the output of mapper and before the reducer.When the mapper task is complete, the results are sorted by key, partitioned if there are multiple reducers, and then written to disk.Using the input from each mapper <k2,v2> , we collect all the values for each unique key k2. This output from the shuffle phase in the form of <k2,list(v2)> is sent as input to reducer phase.

MapReduce Example

**Use Case**

Find the number of occurrences of the word using Map Reduce in a text file

**Solution:**

**Step 1:** Upload the file on HDFS data.txt from /usr/Desktop(local path) to /Hadoop/data (Hadoop folder).

**$hadoop fs**?put /usr/Desktop/data.txt /Hadoop/data

**Step 2:**Write the Map reduce program using eclipse and make the jar of it and name it count.

*File: wc\_mapper.java*

1. import java.io.IOException;
2. import java.util.StringTokenizer;
3. import org.apache.hadoop.io.IntWritable;
4. import org.apache.hadoop.io.LongWritable;
5. import org.apache.hadoop.io.Text;
6. import org.apache.hadoop.mapred.MapReduceBase;
7. import org.apache.hadoop.mapred.Mapper;
8. import org.apache.hadoop.mapred.OutputCollector;
9. import org.apache.hadoop.mapred.Reporter;
10. public class wc\_mapper extends MapReduceBase implements Mapper**<LongWritable**,Text,Text,IntWritable**>**{
11. private final static IntWritable one = new IntWritable(1);
12. private Text word = new Text();
13. public void map(LongWritable key, Text value,OutputCollector**<Text**,IntWritable**>** output,
14. Reporter reporter) throws IOException{
15. String line = value.toString();
16. StringTokenizer  tokenizer = new StringTokenizer(line);
17. while (tokenizer.hasMoreTokens()){
18. word.set(tokenizer.nextToken());
19. output.collect(word, one);
20. }
21. }
23. }

*File: wc\_reducer.java*

1. import java.io.IOException;
2. import java.util.Iterator;
3. import org.apache.hadoop.io.IntWritable;
4. import org.apache.hadoop.io.Text;
5. import org.apache.hadoop.mapred.MapReduceBase;
6. import org.apache.hadoop.mapred.OutputCollector;
7. import org.apache.hadoop.mapred.Reducer;
8. import org.apache.hadoop.mapred.Reporter;
10. public class wc\_reducer extends MapReduceBase implements Reducer**<Text**,IntWritable,Text,IntWritable**>** {
11. public void reduce(Text key, Iterator**<IntWritable>** values,OutputCollector**<Text**,IntWritable**>** output,
12. Reporter reporter) throws IOException {
13. int sum=0;
14. while (values.hasNext()) {
15. sum+=values.next().get();
16. }
17. output.collect(key,new IntWritable(sum));
18. }
19. }

*File: wc\_runner.java*

1. import java.io.IOException;
2. import org.apache.hadoop.fs.Path;
3. import org.apache.hadoop.io.IntWritable;
4. import org.apache.hadoop.io.Text;
5. import org.apache.hadoop.mapred.FileInputFormat;
6. import org.apache.hadoop.mapred.FileOutputFormat;
7. import org.apache.hadoop.mapred.JobClient;
8. import org.apache.hadoop.mapred.JobConf;
9. import org.apache.hadoop.mapred.TextInputFormat;
10. import org.apache.hadoop.mapred.TextOutputFormat;
11. public class wc\_runner {
12. public static void main(String[] args) throws IOException{
13. JobConf conf = new JobConf(wc\_runner.class);
14. conf.setJobName("WordCount");
15. conf.setOutputKeyClass(Text.class);
16. conf.setOutputValueClass(IntWritable.class);
17. conf.setMapperClass(wc\_mapper.class);
18. conf.setCombinerClass(wc\_reducer.class);
19. conf.setReducerClass(wc\_reducer.class);
20. conf.setInputFormat(TextInputFormat.class);
21. conf.setOutputFormat(TextOutputFormat.class);
22. FileInputFormat.setInputPaths(conf,new Path(args[0]));
23. FileOutputFormat.setOutputPath(conf,new Path(args[1]));
24. JobClient.runJob(conf);
25. }
26. }

**Step 3:**Run the jar file

$hadoop jar count.jar WordCount /Hadoop/data.txt/user/root/example\_count

The output is stored in example\_countfolder.

