How Map-Reduce works?
Data flow in ***WordCount*** java example

Chun-Chen Tu

timtu@umich.edu

Before we start

- There are also bunch of documents of Mapreduce + Eclipse + Hadoop plugins
  - Google it!
- ***WordCount*** java code comes from http://hadoop.apache.org/docs/r1.2.1

```
package org.myorg;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;
```

mywordcount.java

Class name should consist with file name

```
public class mywordcount
{
    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
    {
                    ... Map Codes ...
    }

    public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>
    {
                    ... Reduce Codes ...
    }

    public static void main(String[] args) throws Exception
    {
                    JobConf conf = new JobConf(mywordcount.class);
                    ... Job Configuration ...
                    JobClient.runJob(conf);
    }

}
```

```
package org.myorg;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;
```

Location
| ->org
| ->myorg
| -> mywordcount

```
public class mywordcount
{
    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
    {
                    ... Map Codes ...
    }

    public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>
    {
                    ... Reduce Codes ...
    }

    public static void main(String[] args) throws Exception
    {
                    JobConf conf = new JobConf(mywordcount.class);
                    ... Job Configuration ...
                    JobClient.runJob(conf);
    }

}
```

```
package org.myorg;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;
```

These are defined in hadoop library.
Should be include when compileing

```
public class mywordcount
{
    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
    {
                    ... Map Codes ...
    }

    public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>
    {
                    ... Reduce Codes ...
    }

    public static void main(String[] args) throws Exception
    {
                    JobConf conf = new JobConf(mywordcount.class);
                    ... Job Configuration ...
                    JobClient.runJob(conf);
    }

}
```

Input key

Output key

```
public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
{
    private final static IntWritable one = new IntWritable(1);

    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter)  \
     throws IOException

    {

                String line = value.toString();

                StringTokenizer tokenizer = new StringTokenizer(line);

                while (tokenizer.hasMoreTokens())
                {
                            word.set(tokenizer.nextToken());

                            output.collect(word, one);

                 }

   }


 }
```

Input value

Output value

```
public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
{
    private final static IntWritable one = new IntWritable(1);

    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter)  \
     throws IOException

    {

            String line = value.toString();

            StringTokenizer tokenizer = new StringTokenizer(line);

            while (tokenizer.hasMoreTokens())
            {
                    word.set(tokenizer.nextToken());

                    output.collect(word, one);

             }

    }


 }
```

xxxWritable : Hadoop defined variable type
one => an IntWritable type object, it's value = 1

Input.txt
Hello, how are you?
I'm fine thank you, and you?

```
public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
{
    private final static IntWritable one = new IntWritable(1);

    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter)  \
     throws IOException

    {
                    String line = value.toString();

                    StringTokenizer tokenizer = new StringTokenizer(line);

                    while (tokenizer.hasMoreTokens())
                    {
                                word.set(tokenizer.nextToken());

                                output.collect(word, one);

                     }

   }


 }
```

value (Text type): Hello, how are you?\nI'm fine thank you, and you?\n (Hadoop type)
line (String type): Hello, how are you?\nI'm fine thank you, and you?\n (Java type)

Input.txt
Hello, how are you?
I'm fine thank you, and you?

```
public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
{
    private final static IntWritable one = new IntWritable(1);

    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter)  \
     throws IOException

    {

            String line = value.toString();

            StringTokenizer tokenizer = new StringTokenizer(line);

            while (tokenizer.hasMoreTokens())
            {
                    word.set(tokenizer.nextToken());

                    output.collect(word, one);

             }

   }


 }
```

tokenizer:
Hello,
how
are
you?
I'm
fine
thank
you,
and
you?

next

next

Input.txt
Hello, how are you?
I'm fine thank you, and you?

```
public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
{
    private final static IntWritable one = new IntWritable(1);

    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter)  \
     throws IOException

    {

            String line = value.toString();

            StringTokenizer tokenizer = new StringTokenizer(line);

            while (tokenizer.hasMoreTokens())
            {
                    word.set(tokenizer.nextToken());

                    output.collect(word, one);

             }

   }


 }
```

tokenizer:

Hello,
how
are
you?
I'm
fine
thank
you,
and
you?

Input.txt
Hello, how are you?
I'm fine thank you, and you?

```
public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
{
    private final static IntWritable one = new IntWritable(1);

    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter)  \
     throws IOException

    {

            String line = value.toString();

            StringTokenizer tokenizer = new StringTokenizer(line);

            while (tokenizer.hasMoreTokens())
            {
                    word.set(tokenizer.nextToken());

                    output.collect(word, one);

             }

   }


 }
```

word: "Hello,"

tokenizer:

Hello,
how
are
you?
I'm
fine
thank
you,
and
you?

Input.txt
Hello, how are you?
I'm fine thank you, and you?

```
public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
{
    private final static IntWritable one = new IntWritable(1);

    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter)  \
     throws IOException

    {

            String line = value.toString();

            StringTokenizer tokenizer = new StringTokenizer(line);

            while (tokenizer.hasMoreTokens())
            {
                    word.set(tokenizer.nextToken());

                    output.collect(word, one);

             }

   }


 }
```

output: <"Hello," , 1>

tokenizer:

Hello,
how
are
you?
I'm
fine
thank
you,
and
you?

Input.txt
Hello, how are you?
I'm fine thank you, and you?

```
public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
{
    private final static IntWritable one = new IntWritable(1);

    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter)  \
     throws IOException

    {

            String line = value.toString();

            StringTokenizer tokenizer = new StringTokenizer(line);

            while (tokenizer.hasMoreTokens())
            {
                    word.set(tokenizer.nextToken());

                    output.collect(word, one);

             }

   }


 }
```

Final output
<"Hello," , 1>
<"how" , 1>
<"are" , 1>
<"you?" , 1>
<"I'm" , 1>
<"fine" , 1>
<"thank" , 1>
<"you," , 1>
<"and" , 1>
<"you?" , 1>

Some operations done by hadoop

output
<“Hello,” , 1>
<“how” , 1>
<“are” , 1>
<“you?” , 1>
<“I’m” , 1>
<“fine” , 1>
<“thank” , 1>
<“you,” , 1>
<“and” , 1>
<“you?” , 1>

output
<“Hello,” , 1>
<“how” , 1>
<“are” , 1>
<“you?” , 1 , 1>
<“I’m” , 1>
<“fine” , 1>
<“thank” , 1>
<“you,” , 1>
<“and” , 1>

```
public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>
{

    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter \
reporter) throws IOException
    {

                int sum = 0;
                while (values.hasNext())
                {

                        sum += values.next().get();

                }

                output.collect(key, new IntWritable(sum));

    }

}
```

<“Hello,” , 1>
<“how” , 1>
<“are” , 1>
<“you?” , 1 , 1>
<“I’m” , 1>
<“fine” , 1>
<“thank” , 1>
<“you,” , 1>
<“and” , 1>

```
public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>
{

    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter \
reporter) throws IOException
    {

                int sum = 0;
                while (values.hasNext())
                {

                        sum += values.next().get();

                }

                output.collect(key, new IntWritable(sum));

    }

}
```

key: you?
value: 1 , 1

```
public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>
{

    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter \
reporter) throws IOException
    {

                int sum = 0;
                while (values.hasNext())
                {

                        sum += values.next().get();

                }

                output.collect(key, new IntWritable(sum));

    }

}
```

sum : 1

key: you?
value: 1 , 1

```
public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>
{

    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter \
reporter) throws IOException
    {

                int sum = 0;
                while (values.hasNext())
                {

                        sum += values.next().get();

                }

                output.collect(key, new IntWritable(sum));

    }

}
```

sum : 1

key: you?
value: 1 , 1

```
public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>
{

    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter \
reporter) throws IOException
    {

                int sum = 0;
                while (values.hasNext())
                {

                        sum += values.next().get();

                }

                output.collect(key, new IntWritable(sum));

    }

}
```

sum : 2

key: you?
value: 1 , 1

```
public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>
{

    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter \
reporter) throws IOException
    {

                int sum = 0;
                while (values.hasNext())
                {

                        sum += values.next().get();

                }

                output.collect(key, new IntWritable(sum));

    }

}
```

sum : 2

key: you?
value: 1 , 1

```
package org.myorg;

public class mywordcount
{
    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
    {
                    … Map Codes …
    }

    public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>
    {
                    … Reduce Codes …
    }

    public static void main(String[] args) throws Exception
    {
                    JobConf conf = new JobConf(mywordcount.class);
                    conf.setJobName("wordcount");
                    conf.setOutputKeyClass(Text.class);
                    conf.setOutputValueClass(IntWritable.class);
                    conf.setMapperClass(Map.class);
                    conf.setCombinerClass(Reduce.class);
                    conf.setReducerClass(Reduce.class);
                    conf.setInputFormat(TextInputFormat.class);
                    conf.setOutputFormat(TextOutputFormat.class);
                    FileInputFormat.setInputPaths(conf, new Path(args[0]));
                    FileOutputFormat.setOutputPath(conf, new Path(args[1]));
                    JobClient.runJob(conf);
    }
}
```

Compile

mkdir wordcount_dir
javac –classpath hadoop-core-1.2.1.jar -d wordcount_dir mywordcount.java

hadoop-core-1.2.1.jar : the hadoop library

```
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;
```

The result directory

```
hadoop@cloud11:~$ tree wordcount_dir
wordcount_dir
└── org
    └── myorg
        ├── mywordcount.class
        ├── mywordcount$Map.class
        └── mywordcount$Reduce.class
```

```
package org.myorg;

public class mywordcount
{
    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
    {
                    … Map Codes …
    }

    public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable>
    {
                    … Reduce Codes …
    }

    public static void main(String[] args) throws Exception
    {
            JobConf conf = new JobConf(mywordcount.class);
            conf.setJobName("wordcount");
            conf.setOutputKeyClass(Text.class);
            conf.setOutputValueClass(IntWritable.class);
            conf.setMapperClass(Map.class);
            conf.setCombinerClass(Reduce.class);
            conf.setReducerClass(Reduce.class);
            conf.setInputFormat(TextInputFormat.class);
            conf.setOutputFormat(TextOutputFormat.class);
            FileInputFormat.setInputPaths(conf, new Path(args[0]));
            FileOutputFormat.setOutputPath(conf, new Path(args[1]));
            JobClient.runJob(conf);
    }
}
```

```
hadoop@cloud11:~$ tree wordcount_dir
wordcount_dir
└── org
    └── myorg
        ├── mywordcount.class
        ├── mywordcount$Map.class
        └── mywordcount$Reduce.class
```

Make jar and execute

Make jar:

jar –cvf mywordcount.jar –C wordcount_dir .

Execute: First you have to put your input file on hdfs

hadoop dfs –put input.txt /input.txt

```
Hello, how are you?
I'm fine thank you, and you?
```

input.txt

Execute the jar file

hadoop jar mywordcount.jar org.myorg.mywordcount /input /output

```
hadoop@cloud11:~$ tree wordcount_dir
wordcount_dir
└── org
    └── myorg
        ├── mywordcount.class
        ├── mywordcount$Map.class
        └── mywordcount$Reduce.class
```

```
Hello, how are you?
I'm fine thank you, and you?
```

input.txt

Check output

Check the output directory:

hadoop dfs –ls /output

```
hadoop@cloud11:~$ hadoop dfs -ls /output
Found 3 items
-rw-r--r--   3 hadoop supergroup          0 2014-01-14 12:56 /output/_SUCCESS
drwxr-xr-x   - hadoop supergroup          0 2014-01-14 12:56 /output/_logs
-rw-r--r--   3 hadoop supergroup         62 2014-01-14 12:56 /output/part-00000
```

hadoop dfs –cat /output/part-00000

```
hadoop@cloud11:~$ hadoop dfs -cat /output/part-00000
Hello,  1
I'm     1
and     1
are     1
fine    1
how     1
thank   1
you,    1
you?    2
```